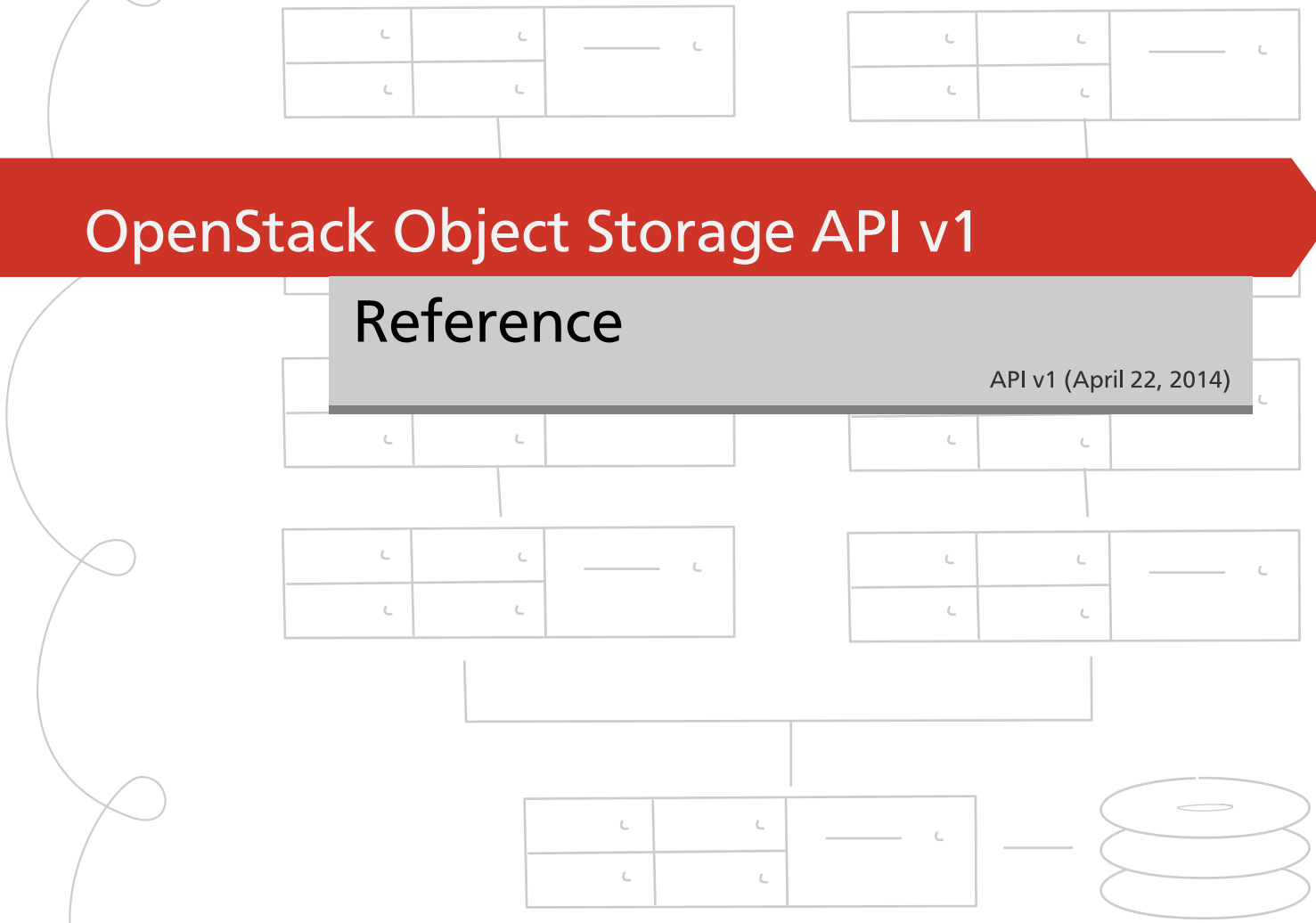
OpenStack Object Storage API v1

Reference

API v1 (April 22, 2014)



BUILT FOR

openstack™

CLOUD SOFTWARE

openstack™

docs.openstack.org

OpenStack Object Storage API v1 Reference

API v1 (2014-04-22)
Copyright © 2010-2014 OpenStack Foundation All rights reserved.

This reference is for software developers who develop applications by using the OpenStack™ Object Storage Application Programming Interface (API) v1.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Table of Contents

Preface 4
Intended Audience 4
Document change history 4
1. Object Storage API concepts and features 1
Object Storage API overview 1
Environment variables required to run examples 4
Discoverability 4
Authentication 5
Serialized response formats 6
Object versioning 8
Large objects 11
Assign CORS headers to requests 15
Use Content-Encoding metadata 16
Use the Content-Disposition metadata 16
Schedule objects for deletion 16
Pseudo-hierarchical folders and directories 17
Page through large lists of containers or objects 19
Auto-extract archive files 19
Bulk delete 21
Container synchronization 22
Container quotas 24
Temporary URL middleware 24
Form POST middleware 27
Create static website 30
2. Object Storage API operations 32
Accounts 32
Containers 42
Objects 56
3. Object Storage API examples 76
cURL commands 76
Authenticate 78
Account services 79
Container services 81
Object services 86

Preface

Intended Audience 4
Document change history 4

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

OpenStack Object Storage enables you to store and get files and content through the Representational State Transfer (REST) interface. There are also language-specific APIs that utilize the RESTful API but make it much easier for developers to integrate into their applications.

For more details about the OpenStack Object Storage service, see http://docs.openstack.org/developer/swift/.

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

Intended Audience

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

- RESTful web services
- HTTP/1.1

You can also find language-specific APIs in several popular programming languages such as C#/.NET, Java, PHP, Python, and Ruby. These APIs utilize the REST API and are provided to help developers rapidly integrate OpenStack Object Storage support into their applications without needing to write at the REST interface. Each API includes its own documentation in its native format. For example, the Java API includes Javadoc documentation.

Document change history

This version of the document replaces and obsoletes all previous versions. The following table describes the latest changes:

Revision Date	Summary of Changes
April 16, 2014	• Added `X-Newest` header to show container details and list objects call.
February 3, 2014	• Rewrote introduction and validated all code examples.
January 24, 2014	• Added information about form **POST** and temporary URL middleware.
October 16, 2013	• Updated the description of the delete account metadata method.

Revision Date	Summary of Changes
September 28, 2013	• Clarify UTF-8 and URL-encoding of container and object names in headers.
June 10, 2013	• Corrected delete container to be delete object.
May 29, 2013	• Updates for static large object support.
February 2, 2013	• Moved code samples into separate files for content sharing.
September 12, 2012	• Fixed bug 1049362 - Adds back metadata deletion information.
September 5, 2012	• Fixed bugs 1009706 - Added Object Versioning and Static Web.
March 29, 2012	• Fixed bugs 890435 and 907563; Add/Update Container Metadata and Expiring Objects. Changed to Maven 1.0.10.
February 10, 2011	• Revised to change first to last in the first range example for fetching a portion of an object.
January 25, 2011	• Revised for OpenStack Object Storage use by removing CDN references, Rackspace Cloud references, and revised account examples and URLs for generic implementations. Clarified that container and object names must be UTF-8 encoded.
January 12, 2011	• Removed references to Access Control List (ACL). Fixed error in examples referring to `X-Auth-Key` where it should be `X-Auth-Token`. Added section numbers.
January 4, 2011	• Expanded authentication information for UK release. Added `delimiter` as a query parameter and server-side object copy example.
May 5, 2008	• Initial release.

1. Object Storage API concepts and features

Object Storage API overview 1
Environment variables required to run examples 4
Discoverability 4
Authentication 5
Serialized response formats 6
Object versioning 8
Large objects 11
Assign CORS headers to requests 15
Use Content-Encoding metadata 16
Use the Content-Disposition metadata 16
Schedule objects for deletion 16
Pseudo-hierarchical folders and directories 17
Page through large lists of containers or objects 19
Auto-extract archive files 19
Bulk delete 21
Container synchronization 22
Container quotas 24
Temporary URL middleware 24
Form POST middleware 27
Create static website 30

Object Storage API overview

OpenStack Object Storage is an object-based storage system that stores content and metadata as objects. You create, modify, and get objects and metadata by using the Object Storage API, which is implemented as a set of Representational State Transfer (REST) web services.

For an introduction to OpenStack Object Storage, see Object Storage in the *OpenStack Cloud Administrator Guide*.

You use the HTTPS (SSL) protocol to interact with Object Storage, and you use standard HTTP calls to perform API operations. You can also use language-specific APIs, which use the RESTful API, that make it easier for you to integrate into your applications.

To assert your right to access and change data in an account, you identify yourself to Object Storage by using an authentication token. To get a token, you present your credentials to an authentication service. The authentication service returns a token and the URL for the account. Depending on which authentication service that you use, the URL for the account appears in:

- **OpenStack Identity Service**. The URL is defined in the service catalog.
- **Tempauth**. The URL is provided in the `X-Storage-Url` response header.

In both cases, the URL is the full URL and includes the account resource. For information about authentication, see the section called “Authentication” [5].

The Object Storage API supports the standard, non-serialized response format, which is the default, and both JSON and XML serialized response formats. See the section called "Serialized response formats" [6].

The Object Storage system organizes data in a hierarchy, as follows:

- **Account**. Represents the top-level of the hierarchy.

  Your service provider creates your account and you own all resources in that account. The account defines a namespace for containers. A container might have the same name in two different accounts.

  In the OpenStack environment, *account* is synonymous with a project or tenant.

- **Container**. Defines a namespace for objects. An object with the same name in two different containers represents two different objects. You can create any number of containers within an account.

  In addition to containing objects, you can also use the container to control access to objects by using an access control list (ACL). You cannot store an ACL with individual objects.

  In addition, you configure and control many other features, such as object versioning, at the container level. See the section called "Object versioning" [8].

  You can bulk-delete up to 10,000 containers in a single request. See the section called "Bulk delete" [21].

- **Object**. Stores data content, such as documents, images, and so on. You can also store custom metadata with an object.

  With the Object Storage API, you can:

  - Store an unlimited number of objects. Each object can be as large as 5 GB, which is the default. You can configure the maximum object size.

  - Upload and store objects of any size with large object creation. See the section called "Large objects" [11].

  - Use cross-origin resource sharing to manage object security. See the section called "Assign CORS headers to requests" [15].

  - Compress files. See the section called "Use Content-Encoding metadata" [16].

  - Override browser behavior for an object. See the section called "Use the Content-Disposition metadata" [16].

  - Schedule objects for deletion. See the section called "Schedule objects for deletion" [16].

  - Bulk-delete up to 10,000 objects in a single request. See the section called "Bulk delete" [21].

  - Auto-extract archive files. See the section called "Auto-extract archive files" [19].

- Generate a URL that provides time-limited **GET** access to an object. See the section called "Temporary URL middleware" [24].
- Upload objects directly to the Object Storage system from a browser by using form **POST** middleware. See the section called "Form POST middleware" [27].

The account, container, and object hierarchy affects the way you interact with the Object Storage API.

Specifically, the resource path reflects this structure and has this format:

```
/v1/{account}/{container}/{object}
```

For example, for the `flowers/rose.jpg` object in the `images` container in the `12345678912345` account, the resource path is:

```
/v1/12345678912345/images/flowers/rose.jpg
```

Notice that the object name contains the `/` character. This slash does not indicate that Object Storage has a sub-hierarchy called `flowers` because containers do not store objects in actual sub-folders. However, the inclusion of `/` or a similar convention inside object names enables you to create pseudo-hierarchical folders and directories. See the section called "Pseudo-hierarchical folders and directories" [17].

For example, if the endpoint for Object Storage is `objects.mycloud.com`, the returned URL is `https://objects.mycloud.com/v1/12345678912345`.

To access a container, append the container name to the resource path.

To access an object, append the container and the object name to the path.

If you have a large number of containers or objects, you can use query parameters to page through large lists of containers or objects. Use the *`marker`*, *`limit`*, and *`end_marker`* query parameters to control how many items are returned in a list and where the list starts or ends. See the section called "Page through large lists of containers or objects" [19].

Object Storage HTTP requests have the following default constraints. Your service provider might use different default values.

Item	Maximum value	Notes
Number of HTTP headers	90	
Length of HTTP headers	4096 bytes	
Length per HTTP request line	8192 bytes	
Length of HTTP request	5 GB	
Length of container names	256 bytes	Cannot contain the / character.
Length of object names	1024 bytes	By default, there are no character restrictions.

You must UTF-8-encode and then URL-encode container and object names before you call the API binding. If you use an API binding that performs the URL-encoding for you, do not URL-encode the names before you call the API binding. Otherwise, you double-encode these names. Check the length restrictions against the URL-encoded string.

These sections describe the operations that you can perform with the Object Storage API:

- the section called "Accounts" [32]. Use to perform account-level tasks.

  Lists containers for a specified account. Creates, updates, and deletes account metadata. Shows account metadata.

- the section called "Containers" [42]. Use to perform container-level tasks.

  Lists objects in a specified container. Creates, shows details for, and deletes containers. Creates, updates, shows, and deletes container metadata.

- the section called "Objects" [56]. Use to perform object-level tasks.

  Creates, replaces, shows details for, and deletes objects. Copies objects with another object with a new or different name. Updates object metadata.

Environment variables required to run examples

To run the cURL command examples for the Object Storage API requests, set these environment variables:

- `publicURL`. The public URL that is the HTTP endpoint from where you can access Object Storage. It includes the Object Storage API version number and your account name. For example, `https://23.253.72.207/v1/my_account`.

- `token`. The authentication token for Object Storage.

To obtain these values, run the **swift stat -v** command.

As shown in this example, the public URL appears in the `StorageURL` field, and the token appears in the `Auth Token` field:

```
StorageURL: https://23.253.72.207/v1/my_account
Auth Token: {token}
Account: my_account
Containers: 2
Objects: 3
Bytes: 47
Meta Book: MobyDick
X-Timestamp: 1389453423.35964
X-Trans-Id: txee55498935404a2caad89-0052dd3b77
Content-Type: text/plain; charset=utf-8
Accept-Ranges: bytes
```

Discoverability

Your Object Storage system might not enable all features that this document describes. These features are:

- the section called "Large objects" [11]

- the section called "Schedule objects for deletion" [16]

- the section called “Auto-extract archive files” [19]
- the section called “Bulk delete” [21]
- the section called “Container synchronization” [22]
- the section called “Container quotas” [24]
- the section called “Temporary URL middleware” [24]
- the section called “Form POST middleware” [27]
- the section called “Create static website” [30]

To discover which features are enabled in your Object Storage system, use the `/info` request. However, your service provider might have disabled the `/info` request, or you might be using an older version that does not support the `/info` request.

To use the `/info` request, send a **GET** request using the `/info` path to the Object Store endpoint as shown in this example:

```
# curl https://storage.clouddrive.com/info
```

This example shows a truncated response body:

```
{
   "swift":{
      "version":"1.11.0"
   },
   "staticweb":{

   },
   "tempurl":{

   }
}
```

This output shows that the Object Storage system has enabled the static website and temporary URL features.

Authentication

The *owner* of an Object Storage account controls access to that account and its containers and objects. An owner is the user who has the `admin` role for that tenant. The tenant is also known as the project or account. As the account owner, you can modify account metadata and create, modify, and delete containers and objects.

To identify yourself as the account owner, include an authentication token in the `X-Auth-Token` header in the API request.

Depending on the token value in the `X-Auth-Token` header, one of the following actions occur:

- `X-Auth-Token` contains the token for the account owner.

  The request is permitted and has full access to make changes to the account.

- The `X-Auth-Token` header is omitted or it contains a token for a non-owner or a token that is not valid.

  The request fails with a 401 Unauthorized or 403 Forbidden response.

  You have no access to accounts or containers, unless an access control list (ACL) explicitly grants access.

  The account owner can grant account and container access to users through access control lists (ACLs). For more information about ACLs, see the section called "Container ACLs" [81].

The following table describes the authentication services that you can use with Object Storage:

Authentication service	Description
OpenStack Identity Service (Keystone)	The Object Storage account is synonymous with the project or tenant ID. For information about the Identity Service, see the section called "Authenticate with the Identity Service" [78].
Tempauth middleware	Object Storage includes this middleware. User and account management is performed in the Object Storage system itself. For information about Tempauth, see the section called "Authenticate with Tempauth" [79].
swauth (in GitHub) or other custom middleware	This custom middleware is modeled on Tempauth, so usage is typically similar to Tempauth. Specifically, you use the `X-Auth-Token` header to pass an authentication token to an API request.

Authentication tokens expire after a time period that the authentication service defines. When a token expires, use of the token causes requests to fail with a 401 Unauthorized response. To continue, you must obtain a new token.

Serialized response formats

By default, the Object Storage API uses a `text/plain` response format. In addition, both JSON and XML data serialization response formats are supported.

Note

To run the cURL command examples, you must export environment variables.

To define the response format, use one of these methods:

Method	Description
`format=`*`format`* query parameter	Append this parameter to the URL for a **GET** request, where *`format`* is `json` or `xml`.
`Accept` request header	Include this header in the **GET** request. The valid header values are: • `text/plain`. Plain text response format. The default.

Method	Description
	• `application/json`. JSON data serialization response format. • `application/xml` or `text/xml`. XML data serialization response format.

Example 1.1. JSON example with format query parameter

For example, this request uses the *`format`* query parameter to ask for a JSON response:

```
# curl -i $publicURL?format=json -X GET -H "X-Auth-Token: $token"
```

```
HTTP/1.1 200 OK
Content-Length: 96
X-Account-Object-Count: 1
X-Timestamp: 1389453423.35964
X-Account-Meta-Subject: Literature
X-Account-Bytes-Used: 14
X-Account-Container-Count: 2
Content-Type: application/json; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: tx274a77a8975c4a66aeb24-0052d95365
Date: Fri, 17 Jan 2014 15:59:33 GMT
```

Object Storage lists container names with additional information in JSON format:

```
[
   {
      "count":0,
      "bytes":0,
      "name":"janeausten"
   },
   {
      "count":1,
      "bytes":14,
      "name":"marktwain"
   }
]
```

Example 1.2. XML example with Accept header

This request uses the `Accept` request header to ask for an XML response:

```
# curl -i $publicURL -X GET -H "X-Auth-Token: $token" -H "Accept: application/
xml; charset=utf-8"
```

```
HTTP/1.1 200 OK
Content-Length: 263
X-Account-Object-Count: 3
X-Account-Meta-Book: MobyDick
X-Timestamp: 1389453423.35964
X-Account-Bytes-Used: 47
X-Account-Container-Count: 2
Content-Type: application/xml; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: txf0b4c9727c3e491694019-0052e03420
Date: Wed, 22 Jan 2014 21:12:00 GMT
```

Object Storage lists container names with additional information in XML format:

```
<?xml version="1.0" encoding="UTF-8"?>
<account name="AUTH_73f0aa26640f4971864919d0eb0f0880">
    <container>
        <name>janeausten</name>
        <count>2</count>
        <bytes>33</bytes>
    </container>
    <container>
        <name>marktwain</name>
        <count>1</count>
        <bytes>14</bytes>
    </container>
</account>
```

The remainder of the examples in this guide use standard, non-serialized responses. However, all **GET** requests that perform list operations accept the *format* query parameter or Accept request header.

Object versioning

You can store multiple versions of your content so that you can recover from unintended overwrites. Object versioning is an easy way to implement version control, which you can use with any type of content.

Note

You cannot version a large-object manifest file, but the large-object manifest file can point to versioned segments.

It is strongly recommended that you put non-current objects in a different container than the container where current object versions reside.

To enable and use object versioning

1. To enable object versioning, ask your cloud provider to set the allow_versions option to TRUE in the container configuration file.

2. Create an archive container to store older versions of objects.

   Create the archive container:

   ```
   # curl -i $publicURL/archive -X PUT -H "Content-Length: 0" -H "X-Auth-
   Token: $token"
   ```

   ```
   HTTP/1.1 201 Created
   Content-Length: 0
   Content-Type: text/html; charset=UTF-8
   X-Trans-Id: tx46f8c29050834d88b8d7e-0052e1859d
   Date: Thu, 23 Jan 2014 21:11:57 GMT
   ```

3. Create a current container to store current versions of objects.

   Include the X-Versions-Location header. This header defines the container that holds the non-current versions of your objects. You must UTF-8-encode and then URL-encode the container name before you include it in the X-Versions-Location header. This header enables object versioning for all objects in the current container.

Changes to objects in the `current` container automatically create non-current versions in the `archive` container.

Create the `current` container:

```
# curl -i $publicURL/current -X PUT -H "Content-Length: 0" -H "X-Auth-
Token: $token" -H "X-Versions-Location: archive"
```

```
HTTP/1.1 201 Created
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txb91810fb717347d09eec8-0052e18997
Date: Thu, 23 Jan 2014 21:28:55 GMT
```

4. Create the first version of an object in the `current` container:

```
# curl -i $publicURL/current/my_object --data-binary 1 -X PUT -H "Content-
Length: 0" -H "X-Auth-Token: $token"
```

```
HTTP/1.1 201 Created
Last-Modified: Thu, 23 Jan 2014 21:31:22 GMT
Content-Length: 0
Etag: d41d8cd98f00b204e9800998ecf8427e
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx5992d536a4bd4fec973aa-0052e18a2a
Date: Thu, 23 Jan 2014 21:31:22 GMT
```

Nothing is written to the non-current version container when you initially **PUT** an object in the `current` container. However, subsequent **PUT** requests that edit an object trigger the creation of a version of that object in the `archive` container.

These non-current versions are named as follows:

```
<length><object_name><timestamp>
```

Where `length` is the 3-character, zero-padded hexadecimal character length of the object, `<object_name>` is the object name, and `<timestamp>` is the time when the object was initially created as a current version.

5. Create a second version of the object in the `current` container:

```
# curl -i $publicURL/current/my_object --data-binary 2 -X PUT -H "Content-
Length: 0" -H "X-Auth-Token: $token"
```

```
HTTP/1.1 201 Created
Last-Modified: Thu, 23 Jan 2014 21:41:32 GMT
Content-Length: 0
Etag: d41d8cd98f00b204e9800998ecf8427e
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx468287ce4fc94eada96ec-0052e18c8c
Date: Thu, 23 Jan 2014 21:41:32 GMT
```

6. Issue a **GET** request to a versioned object to get the current version of the object. You do not have to do any request redirects or metadata lookups.

List older versions of the object in the `archive` container:

```
# curl -i $publicURL/archive?prefix=009my_object -X GET -H "X-Auth-Token:
 $token"
```

```
HTTP/1.1 200 OK
Content-Length: 30
X-Container-Object-Count: 1
Accept-Ranges: bytes
X-Timestamp: 1390513280.79684
X-Container-Bytes-Used: 0
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx9a441884997542d3a5868-0052e18d8e
Date: Thu, 23 Jan 2014 21:45:50 GMT

009my_object/1390512682.92052
```

Note

A **POST** request to a versioned object updates only the metadata for the object and does not create a new version of the object. New versions are created only when the content of the object changes.

7. Issue a **DELETE** request to a versioned object to remove the current version of the object and replace it with the next-most current version in the non-current container.

```
# curl -i $publicURL/current/my_object -X DELETE -H "X-Auth-Token: $token"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx006d944e02494e229b8ee-0052e18edd
Date: Thu, 23 Jan 2014 21:51:25 GMT
```

List objects in the `archive` container to show that the archived object was moved back to the `current` container:

```
# curl -i $publicURL/archive?prefix=009my_object -X GET -H "X-Auth-Token:
 $token"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
X-Container-Object-Count: 0
Accept-Ranges: bytes
X-Timestamp: 1390513280.79684
X-Container-Bytes-Used: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx044f2a05f56f4997af737-0052e18eed
Date: Thu, 23 Jan 2014 21:51:41 GMT
```

This next-most current version carries with it any metadata last set on it. If want to completely remove an object and you have five versions of it, you must **DELETE** it five times.

8. To disable object versioning for the `current` container, remove its `X-Versions-Location` metadata header by sending an empty key value.

```
# curl -i $publicURL/current -X PUT -H "Content-Length: 0" -H "X-Auth-
Token: $token" -H "X-Versions-Location: "
```

```
HTTP/1.1 202 Accepted
Content-Length: 76
Content-Type: text/html; charset=UTF-8
```

```
X-Trans-Id: txe2476de217134549996d0-0052e19038
Date: Thu, 23 Jan 2014 21:57:12 GMT

<html><h1>Accepted</h1><p>The request is accepted for processing.</p></
html>
```

Large objects

By default, the content of an object cannot be greater than 5 GB. However, you can use a number of smaller objects to construct a large object. The large object is comprised of two types of objects:

- **Segment objects** store the object content. You can divide your content into segments, and upload each segment into its own segment object. Segment objects do not have any special features. You create, update, download, and delete segment objects just as you would normal objects.
- A **manifest object** links the segment objects into one logical large object. When you download a manifest object, Object Storage concatenates and returns the contents of the segment objects in the response body of the request. This behavior extends to the response headers returned by **GET** and **HEAD** requests. The `Content-Length` response header value is the total size of all segment objects. Object Storage calculates the `ETag` response header value by taking the `ETag` value of each segment, concatenating them together, and returning the MD5 checksum of the result. The manifest object types are:

  Static large objects — The manifest object content is an ordered list of the names of the segment objects in JSON format. See the section called “Static large objects” [11].

  Dynamic large objects — The manifest object has no content but it has a `X-Object-Manifest` metadata header. The value of this header is `{container}/{prefix}`, where `{container}` is the name of the container where the segment objects are stored, and `{prefix}` is a string that all segment objects have in common. See the section called “Dynamic large objects” [13].

Note

If you make a **COPY** request by using a manifest object as the source, the new object is a normal, and not a segment, object. If the total size of the source segment objects exceeds 5 GB, the **COPY** request fails. However, you can make a duplicate of the manifest object and this new object can be larger than 5 GB.

Static large objects

To create a static large object, divide your content into pieces and create (upload) a segment object to contain each piece.

You must record the `ETag` response header that the **PUT** operation returns. Alternatively, you can calculate the MD5 checksum of the segment prior to uploading and include this in the `ETag` request header. This ensures that the upload cannot corrupt your data.

List the name of each segment object along with its size and MD5 checksum in order.

Create a manifest object. Include the *?multipart-manifest=put* query string at the end of the manifest object name to indicate that this is a manifest object.

The body of the **PUT** request on the manifest object comprises a json list, where each element contains the following attributes:

- `path`. The container and object name in the format: `{container-name}/{object-name}`
- `etag`. The MD5 checksum of the content of the segment object. This value must match the `ETag` of that object.
- `size_bytes`. The size of the segment object. This value must match the `Content-Length` of that object.

Example 1.3. Static large object manifest list

This example shows three segment objects. You can use several containers and the object names do not have to conform to a specific pattern, in contrast to dynamic large objects.

```
[
   {
      "path":"mycontainer/objseg1",
      "etag":"0228c7926b8b642dfb29554cd1f00963",
      "size_bytes":1468006
   },
   {
      "path":"mycontainer/pseudodir/seg-obj2",
      "etag":"5bfc9ea51a00b790717eeb934fb77b9b",
      "size_bytes":1572864
   },
   {
      "path":"other-container/seg-final",
      "etag":"b9c3da507d2557c1ddc51f27c54bae51",
      "size_bytes":256
   }
]
```

The `Content-Length` request header must contain the length of the json content—not the length of the segment objects. However, after the **PUT** operation completes, the `Content-Length` metadata is set to the total length of all the object segments. A similar situation applies to the `ETag`. If used in the **PUT** operation, it must contain the MD5 checksum of the json content. The `ETag` metadata value is then set to be the MD5 checksum of the concatenated `ETag` values of the object segments. You can also set the `Content-Type` request header and custom object metadata.

When the **PUT** operation sees the *?multipart-manifest=put* query parameter, it reads the request body and verifies that each segment object exists and that the sizes and ETags match. If there is a mismatch, the **PUT**operation fails.

If everything matches, the manifest object is created. The `X-Static-Large-Object` metadata is set to `true` indicating that this is a static object manifest.

Normally when you perform a **GET** operation on the manifest object, the response body contains the concatenated content of the segment objects. To download the manifest list,

use the *?multipart-manifest=get* query parameter. The resulting list is not formatted the same as the manifest you originally used in the **PUT** operation.

If you use the **DELETE** operation on a manifest object, the manifest object is deleted. The segment objects are not affected. However, if you add the *?multipart-manifest=delete* query parameter, the segment objects are deleted and if all are successfully deleted, the manifest object is also deleted.

To change the manifest, use a **PUT** operation with the *?multipart-manifest=put* query parameter. This request creates a manifest object. You can also update the object metadata in the usual way.

Dynamic large objects

You must segment objects that are larger than 5 GB before you can upload them. You then upload the segment objects like you would any other object and create a dynamic large manifest object. The manifest object tells Object Storage how to find the segment objects that comprise 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.

To ensure the download works correctly, you must upload all the object segments to the same container and ensure that each object name is prefixed in such a way that it sorts in the order in which it should be concatenated. You also create and upload a manifest file. The manifest file is 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. You must UTF-8-encode and then URL-encode the container and common prefix in the `X-Object-Manifest` header.

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

Example 1.4. Upload segment of large object request: HTTP

```
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
```

No response body is returned. A status code of 2*nn* (between 200 and 299, inclusive) 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 1.5. Upload next segment of large object request: HTTP

```
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
```

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 causes the concatenated object to be that much larger but you do not need to recreate the manifest file for subsequent additional segments.

Example 1.6. Upload manifest request: HTTP

```
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}/{prefix}
```

Example 1.7. Upload manifest response: HTTP

```
[...]
```

The `Content-Type` in the response for a **GET** or **HEAD** on the manifest is 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.

Comparison of static and dynamic large objects

While static and dynamic objects have similar behavior, this table describes their differences:

Table 1.1. Static and dynamic large objects

	Static large object	Dynamic large object
End-to-end integrity	Assured. The list of segments includes the MD5 checksum (`ETag`) of each segment. You cannot upload the manifest object if the `ETag` in the list differs from the uploaded segment object. If a segment is somehow lost, an attempt to download the manifest object results in an error.	Not guaranteed. The eventual consistency model means that although you have uploaded a segment object, it might not appear in the container listing until later. If you download the manifest before it appears in the container, it does not form part of the content returned in response to a **GET** request.
Upload order	You must upload the segment objects before you upload the manifest object.	You can upload manifest and segment objects in any order. You are recommended to upload the manifest object after the segments in case a premature download of the manifest occurs. However, this is not enforced.
Removal or addition of segment objects	You cannot add or remove segment objects from the manifest. However, you can create a completely new manifest object of the same name with a different manifest list.	You can upload new segment objects or remove existing segments. The names must simply match the `{prefix}` supplied in `X-Object-Manifest`.

	Static large object	Dynamic large object
Segment object size and number	Segment objects must be at least 1 MB in size (by default). The final segment object can be any size. At most, 1000 segments are supported (by default).	Segment objects can be any size.
Segment object container name	The manifest list includes the container name of each object. Segment objects can be in different containers.	All segment objects must be in the same container.
Manifest object metadata	The object has `X-Static-Large-Object` set to `true`. You do not set this metadata directly. Instead the system sets it when you **PUT** a static manifest object.	The `X-Object-Manifest` value is the `{container}/{prefix}`, which indicates where the segment objects are located. You supply this request header in the **PUT** operation.
Copying the manifest object	Include the *`?multipart-manifest=get`* query string in the **COPY** request. The new object contains the same manifest as the original. The segment objects are not copied. Instead, both the original and new manifest objects share the same set of segment objects.	The **COPY** operation does not create a manifest object. To duplicate a manifest object, use the **GET** operation to read the value of `X-Object-Manifest` and use this value in the `X-Object-Manifest` request header in a **PUT** operation. This creates a new manifest object that shares the same set of segment objects as the original manifest object.

Assign CORS headers to requests

Cross-Origin Resource Sharing (CORS) is a specification that defines how browsers and servers communicate across origins by using HTTP headers, such as those assigned by Object Storage API requests. The Object Storage API supports these headers. For more information, see 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

You can assign these headers to only objects.

Example 1.8. Assign CORS header request: HTTP

This example assigns the file origin to the `Origin` header, which ensures that the file originated from a reputable source:

```
PUT $publicURL/{container}/{object} HTTP/1.1
Host: storage.clouddrive.com
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
Origin: http://storage.clouddrive.com
```

Use Content-Encoding metadata

When you create an object or update its metadata, you can optionally set the `Content-Encoding` metadata. This metadata enables you to indicate that the object content is compressed without losing the identity of the underlying media type (`Content-Type`) of the file, such as a video.

Example 1.9. Content-Encoding header request: HTTP

This example assigns an attachment type to the `Content-Encoding` header that indicates how the file is 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
```

Use the Content-Disposition metadata

To override the default behavior for a browser, use the `Content-Disposition` header to specify the override behavior and assign this header to an object. For example, this header might specify that the browser use a download program to save this file rather than show the file, which is the default.

Example 1.10. Override browser default behavior request: HTTP

This example assigns an attachment type to the `Content-Disposition` header. This attachment type indicates that the file is to be downloaded as `goodbye.txt`:

```
# curl -i  $publicURL/marktwain/goodbye -X POST -H "X-Auth-Token: $token" -H
 "Content-Length: 14" -H "Content-Type: application/octet-stream" -H "Content-
Disposition: attachment; filename=goodbye.txt"
```

```
HTTP/1.1 202 Accepted
Content-Length: 76
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txa9b5e57d7f354d7ea9f57-0052e17e13
Date: Thu, 23 Jan 2014 20:39:47 GMT

<html><h1>Accepted</h1><p>The request is accepted for processing.</p></html>
```

Schedule objects for deletion

To discover whether your Object Storage system supports this feature, see the section called “Discoverability” [4]. Alternatively, check with your service provider.

Scheduling an object for deletion is helpful for objects that you do not want to permanently store, such as log files, recurring full backups of a dataset, or documents or images that become outdated at a specified future time.

To schedule an object for deletion, include one of these headers with the **PUT** or **POST** request on the object:

- `X-Delete-After`

  An integer value. Specifies the number of seconds in the future when you want to delete the object.

  This header is converted to an `X-Delete-At` header that is set to the sum of the `X-Delete-After` value plus the current time, in seconds.

- `X-Delete-At`

  A UNIX Epoch timestamp, in integer form. For example, `1348691905` represents `Wed, 26 Sep 2012 20:38:25 GMT`. Specifies the time when you want the object to expire, not be served, and be deleted completely from the object store.

Use the **POST** method to assign expiration headers to existing objects that you want expire.

Delete object at specified time request: HTTP

In the example, the `X-Delete-At` header is assigned 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.

```
# curl -i $publicURL/marktwain/goodbye -X PUT -H "X-Auth-Token: $token" -H "X-Delete-At: 1390581073" -H "Content-Length: 14" -H "Content-Type: application/octet-stream"
```

Delete object after specified interval request: HTTP

This example sets the `X-Delete-After` header to a value in seconds that is equivalent to 10 days. After this time, the object expires.

```
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
```

Pseudo-hierarchical folders and directories

Although you cannot nest directories in OpenStack Object Storage, you can simulate a hierarchical structure within a single container by adding forward slash characters (`/`) in the object name. To navigate the pseudo-directory structure, you can use the `delimiter` query parameter. this examples show you how to use pseudo-hierarchical folders and directories.

Note

In this example, the objects reside in a container called `backups`. Within that container, the objects are organized in a pseudo-directory called `photos`. Keep in mind that the container name is not displayed in the example, but that it

is a part of the object URLs. For instance, the URL of the picture `me.jpg` is `https://storage.swiftdrive.com/v1/CF_xer7_343/backups/photos/me.jpg`.

Example 1.11. List pseudo-hierarchical folders request: HTTP

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 2xx (between 200 and 299, inclusive) 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. You can use any character as a delimiter. However, to use `delimiter` with pseudo-directories, use the parameter slash (`/`).

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

The system returns status code 2xx (between 200 and 299, inclusive) and the requested matching objects. Because you 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 the `subdir` section of json and xml results.

```
photos/
```

Use the `prefix` and `delimiter` parameters 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 2xx (between 200 and 299, inclusive) and the objects and pseudo-directories within the top level pseudo-directory.

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

You can create an unlimited number of nested pseudo-directories. To navigate through them, use a longer `prefix` parameter coupled with the `delimiter` parameter. In this sample output, there is a pseudo-directory called `dogs` within the pseudo-directory `animals`. To navigate directly to the files contained within `dogs`, enter the following command:

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

The system returns status code 2*nn* (between 200 and 299, inclusive) 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
```

Page through large lists of containers or objects

If you have a large number of containers or objects, you can use the *marker*, *limit*, and *end_marker* parameters to control how many items are returned in a list and where the list starts or ends.

- ***marker* parameter**. When you request a list of containers or objects, Object Storage returns a maximum of 10,000 names for each request. To get subsequent names, you must make another request with the *marker* parameter. Set the `marker` parameter to the name of the last item returned in the previous list. You must URL-encode the *marker* value before you send the HTTP request. Object Storage returns a maximum of 10,000 names again.

- ***limit* parameter**. To return fewer than 10,000 names, use the *limit* parameter. If the number of names returned equals the specified *limit* (or 10,000 if you omit the *limit* parameter), you can assume there are more names to list. If the number of names in the list is exactly divisible by the *limit* value, the last request has no content.

- ***end_marker* parameter**. Limits the result set to names that are less than the *end_marker* parameter value. You must URL-encode the *end_marker* value before you send the HTTP request.

For examples of how to page through large lists, see Chapter 3, "Object Storage API examples" [76].

Auto-extract archive files

To discover whether your Object Storage system supports this feature, see the section called "Discoverability" [4]. Alternatively, check with your service provider.

Use the auto-extract archive feature to upload a tar(1) archive file.

The Object Storage system extracts files from the archive file and creates an object.

Auto-extract archive request

To upload an archive file, you make a **PUT** request. Add the *extract-archive=format* query parameter to indicate that you are uploading a tar(1) archive file instead of normal content.

Valid values for the *format* variable are `tar`, `tar.gz`, or `tar.bz2`.

The path you specify in the **PUT** request is a prefix for the resulting object names.

For example, if the first object in the tar(1) archive is `/home/file1.txt` and you specify the `/v1/12345678912345/mybackup/castor/` path, the operation creates the

`castor/home/file1.txt` object in the `mybackup` container in the `12345678912345` account.

In the **PUT** request, you can specify the path for:

- An account
- Optionally, a specific container
- Optionally, a specific object prefix

Create an archive for auto-extract

You must use the tar(1) utility to create the tar(1) archive file.

You can upload regular files but you cannot upload other items, such as empty directories, symbolic links, and so on.

You must UTF-8-encode the member names.

The archive auto-extract feature supports these formats:

- The POSIX.1-1988 Ustar format.
- The GNU tar format. Includes the long name, long link, and sparse extensions.
- The POSIX.1-2001 pax format.

  Use gzip(1) or bzip2(1) to compress the archive.

  Use the *`extract-archive`* query parameter to specify the format. Valid values for this parameter are `tar`, `tar.gz`, or `tar.bz2`.

Auto-extract archive response

When Object Storage processes the request, it performs multiple sub-operations. Even if all sub-operations fail, the operation returns a 201 `Created` status. Some sub-operations might succeed while others fail: Examine the response body to determine the results of each auto-extract archive sub-operation.

You can set the `Accept` request header to one of these values to define the response format:

- `text/plain`. Formats response as plain text. If you omit the `Accept` header, `text/plain` is the default.
- `application/json`. Formats response as JSON.
- `application/xml` or `text/xml`. Formats response as XML.

The following auto-extract archive files example shows a `text/plain` response body where no failures occurred:

```
Number Files Created: 10
```

```
Errors:
```

The following auto-extract archive files example shows a `text/plain` response where some failures occurred. In this example, the Object Storage system is configured to reject certain character strings so that the 400 Bad Request error occurs for any objects that use the restricted strings.

```
Number Files Created: 8
Errors:
/v1/12345678912345/mycontainer/home/xx%3Cyy, 400 Bad Request
/v1/12345678912345/mycontainer/../image.gif, 400 Bad Request
```

The following example shows the failure response in `application/json` format.

```
{
   "Number Files Created":1,
   "Errors":[
      [
         "/v1/12345678912345/mycontainer/home/xx%3Cyy",
         "400 Bad Request"
      ],
      [
         "/v1/12345678912345/mycontainer/../image.gif",
         "400 Bad Request"
      ]
   ]
}
```

Bulk delete

To discover whether your Object Storage system supports this feature, see the section called “Discoverability” [4]. Alternatively, check with your service provider.

With bulk delete, you can delete up to 10,000 objects or containers (configurable) in one request.

Bulk delete request

To perform a bulk delete operation, add the *`bulk-delete`* query parameter to the path of a **POST** or **DELETE** operation.

Note

The **DELETE** operation is supported for backwards compatibility.

The path is the account, such as `/v1/12345678912345`, that contains the objects and containers.

In the request body of the **POST** or **DELETE** operation, list the objects or containers to be deleted. Separate each name with a newline character. You can include a maximum of 10,000 items (configurable) in the list.

In addition, you must:

- UTF-8-encode and then URL-encode the names.

- To indicate an object, specify the container and object name as: *CONTAINER_NAME/OBJECT_NAME*.
- To indicate a container, specify the container name as: *CONTAINER_NAME*. Ensure that the container is empty. If it contains objects, Object Storage cannot delete the container.
- Set the `Content-Type` request header to `text/plain`.

Bulk delete response

When Object Storage processes the request, it performs multiple sub-operations. Even if all sub-operations fail, the operation returns a 200 status. The bulk operation returns a response body that contains details that indicate which sub-operations have succeeded and failed. Some sub-operations might succeed while others fail: Examine the response body to determine the results of each delete sub-operation.

You can set the `Accept` request header to one of these values to define the response format:

- `text/plain`. Formats response as plain text. If you omit the `Accept` header, `text/plain` is the default.
- `application/json`. Formats response as JSON.
- `application/xml` or `text/xml`. Formats response as XML.

The response body contains the following information:

- The number of files actually deleted.
- The number of not found objects.
- Errors. A list of object names and associated error statuses for the objects that failed to delete. The format depends on the value that you set in the `Accept` header.

The following bulk delete response is in `application/xml` format. In this example, the `mycontainer` container is not empty, so it cannot be deleted.

```
<?xml version="1.0" encoding="UTF-8"?>
<delete>
    <number_deleted>2</number_deleted>
    <number_not_found>4</number_not_found>
    <errors>
        <object>
            <name>/v1/12345678912345/mycontainer</name>
            <status>409 Conflict</status>
        </object>
    </errors>
</delete>
```

Container synchronization

To discover whether your Object Storage system supports container synchronization, see the section called “Discoverability” [4]. Alternatively, check with your service provider.

Container synchronization enables you to synchronize the contents of a source container with a destination container. After you set up container synchronization, the system automatically copies objects from the source container to the destination container. Also, the system deletes objects in the destination container that were deleted in the source container.

The system copies objects in a way that object metadata is retained, such as `Last-Modified` and any custom metadata you might have set for the object.

You can configure the source and destination containers, as follows:

- The source container can be on a different or the same Object Storage system that the destination container is on.
- The destination container can be a source container for synchronization for another destination container.
- The destination container can be the original source container: both containers synchronize with each other. Any object that you add to or delete from a container is automatically copied to or deleted from the other container.

The Object Storage system performs the synchronization in the background, and makes no guarantees about performance or timeliness.

Some Object Storage features, such as large object creation, might require the use of several containers. Container synchronization handles each container separately; if your object segments are located in a different container, they are not transferred unless you also set up container synchronization on that container. However, even if both the manifest and segment containers are synchronized, there is no guarantee that the manifest is transferred before the segment objects. An attempt to download the large object from the destination container might fail, be incomplete, or have jumbled content. Object versioning is not supported.

To configure a *source* container for synchronization, set the following metadata headers:

- `X-Container-Sync-To`. Set this metadata header to the following value:

  ```
  //REALM/SYSTEM/DESTINATION_ACCOUNT/DESTINATION_CONTAINER_NAME
  ```

  Your service provider can give you the appropriate values for *REALM* and *SYSTEM*. The objects are sent to the *DESTINATION_ACCOUNT/DESTINATION_CONTAINER_NAME* container. These names can be different from the source account and container names.

- `X-Container-Sync-Key`. Set this metadata header to an arbitrary string value. This value serves as a shared secret. Secure this value just as you would a password.

To configure a *destination* container to receive objects, set the `X-Container-Sync-Key` metadata header to the `X-Container-Sync-Key` value in the source container.

Note

To configure a destination container as the source container for another destination container, set the `X-Container-Sync-To` metadata header as you would for a source container.

Container quotas

You can set quotas on the size and number of objects stored in a container by setting the following metadata:

- `X-Container-Meta-Quota-Bytes`. The size, in bytes, of objects that can be stored in a container.
- `X-Container-Meta-Quota-Count`. The number of objects that can be stored in a container.

When you exceed a container quota, subsequent requests to create objects fail with a 413 Request Entity Too Large error.

The Object Storage system uses an *eventual consistency* model. When you create a new object, the container size and object count might not be immediately updated. Consequently, you might be allowed to create objects even though you have actually exceeded the quota.

At some later time, the system updates the container size and object count to the actual values. At this time, subsequent requests fails. In addition, if you are currently under the `X-Container-Meta-Quota-Bytes` limit and a request uses chunked transfer encoding, the system cannot know if the request will exceed the quota so the system allows the request. However, once the quota is exceeded, any subsequent uploads that use chunked transfer encoding fail.

Temporary URL middleware

To discover whether your Object Storage system supports this feature, see the section called "Discoverability" [4]. Alternatively, check with your service provider.

A temporary URL gives users temporary access to objects. For example, a website might want to provide a link to download a large object in Object Storage, but the Object Storage account has no public access. The website can generate a URL that provides time-limited **GET** access to the object. When the web browser user clicks on the link, the browser downloads the object directly from Object Storage, eliminating the need for the website to act as a proxy for the request.

Ask your cloud administrator to enable the temporary URL feature. For information, see Temporary URL in the *OpenStack Configuration Reference*.

Note

To use **POST** requests to upload objects to specific Object Storage locations, use form **POST** instead of temporary URL middleware. See the section called "Form POST middleware" [27].

Temporary URL format

A temporary URL is comprised of the URL for an object with added query parameters:

Example 1.12. Temporary URL format

```
https://swift-cluster.example.com/v1/my_account/container/❶object
?temp_url_sig=da39a3ee5e6b4b0d3255bfef95601890afd8070❷9
&temp_url_expires=1323479485❸
&filename=My+Test+File.p❹df
```

The example shows these elements:

❶ **Object URL**

Required. The full path URL to the object.

❷ **temp_url_sig**

Required. An HMAC-SHA1 cryptographic signature that defines the allowed HTTP method, expiration date, full path to the object, and the secret key for the temporary URL.

For more information about secret keys, see the section called "Account secret keys" [25].

For more information about signatures, see the section called "HMAC-SHA1 signature for temporary URLs" [26].

❸ **temp_url_expires**

Required. An expiration date as a UNIX Epoch timestamp, which is an integer value. For example, `1390852007` represents `Mon, 27 Jan 2014 19:46:47 GMT`.

For more information, see Epoch & Unix Timestamp Conversion Tools.

❸ **filename**

Optional. Overrides the default file name. Object Storage generates a default file name for **GET** temporary URLs that is based on the object name. Object Storage returns this value in the `Content-Disposition` response header. Browsers can interpret this file name value as a file attachment to be saved.

Account secret keys

Object Storage supports up to two secret keys. You set secret keys at the account level.

To set these keys, set one or both of the following request headers to arbitrary values:

- `X-Account-Meta-Temp-URL-Key`
- `X-Account-Meta-Temp-URL-Key-2`

The arbitrary values serve as the secret keys.

Object Storage checks signatures against both keys, if present, to enable key rotation without invalidating existing temporary URLs.

For example, use the **swift post** command to set the secret key to *MYKEY*:

```
$ swift post -m "X-Account-Meta-Temp-URL-Key: MYKEY"
```

Note

Changing these headers invalidates any previously generated temporary URLs within 60 seconds, which is the memcache time for the key.

HMAC-SHA1 signature for temporary URLs

Temporary URL middleware uses an HMAC-SHA1 cryptographic signature. This signature includes these elements:

- The allowed method. Typically, **GET** or **PUT**.
- Expiry time. In Example 1.13, "HMAC-SHA1 signature for temporary URLs" [26], the expiry time is set to `86400` seconds (or 1 day) into the future.
- The path. Starting with `/v1/` onwards and including a container name and object. In Example 1.13, "HMAC-SHA1 signature for temporary URLs" [26], the path is `/v1/my_account/container/object`. Do not URL-encode the path at this stage.
- The secret key. Set as the `X-Account-Meta-Temp-URL-Key` header value.

This sample Python code shows how to compute a signature for use with temporary URLs:

Example 1.13. HMAC-SHA1 signature for temporary URLs

```
import hmac
from hashlib import sha1
from time import time
method = 'GET'
duration_in_seconds = 60*60*24
expires = int(time() + duration_in_seconds)
path = '/v1/my_account/container/object'
key = 'MYKEY'
hmac_body = '%s\n%s\n%s' % (method, expires, path)
signature = hmac.new(key, hmac_body, sha1).hexdigest()
```

Do not URL-encode the path when you generate the HMAC-SHA1 signature. However, when you make the actual HTTP request, you should properly URL-encode the URL.

The *MYKEY* value is the value you set in the `X-Account-Meta-Temp-URL-Key` request header on the account.

For more information, see RFC 2104: HMAC: Keyed-Hashing for Message Authentication.

swift-temp-url script

Object Storage provides the **swift-temp-url** script that auto-generates the *temp_url_sig* and *temp_url_expires* query parameters. For example, you might run this command:

```
$ bin/swift-temp-url GET 3600 /v1/my_account/container/object MYKEY
```

This command returns the path:

```
/v1/my_account/container/object
?temp_url_sig=5c4cc8886f36a9d0919d708ade98bf0cc71c9e91
&temp_url_expires=1374497657
```

To create the temporary URL, prefix this path with the Object Storage storage host name. For example, prefix the path with `https://swift-cluster.example.com`, as follows:

```
https://swift-cluster.example.com/v1/my_account/container/object
?temp_url_sig=5c4cc8886f36a9d0919d708ade98bf0cc71c9e91
&temp_url_expires=1374497657
```

Form POST middleware

To discover whether your Object Storage system supports this feature, see the section called "Discoverability" [4]. Alternatively, check with your service provider.

You can upload objects directly to the Object Storage system from a browser by using the form **POST** middleware. This middleware uses account secret keys to generate a cryptographic signature for the request. This means that you do not need to send an authentication token in the `X-Auth-Token` header to perform the request.

The form **POST** middleware uses the same secret keys as the temporary URL middleware uses. For information about how to set these keys, see the section called "Account secret keys" [25].

For information about the form **POST** middleware configuration options, see Form post in the *OpenStack Configuration Reference*.

Form POST format

To upload objects to a cluster, you can use an HTML form **POST** request.

The format of the form **POST** request is:

Example 1.14. Form POST format

```
&lt;![CDATA[
<form action="SWIFT_URL"❶
    method="POST"❷
    enctype="multipart/form-data"❸>
    <input type="hidden" name="redirect" value="REDIRECT_URL"❹/>
    <input type="hidden" name="max_file_size" value="BYTES"❺/>
    <input type="hidden" name="max_file_count" value="COUNT"❻/>
    <input type="hidden" name="expires" value="UNIX_TIMESTAMP"❼/>
    <input type="hidden" name="signature" value="HMAC"❽/>
    <input type="file" name="FILE_NA❾ME"/>
    <br/>
    <input type="submit"/❿>
</form>
]]&gt;
```

The example shows these attributes:

❶ **action="*SWIFT_URL*"**

Set to full URL where the objects are to be uploaded. The names of uploaded files are appended to the specified *SWIFT_URL*. So, you can upload directly to the root of a container with a URL like:

```
https://swift-cluster.example.com/v1/my_account/container/
```

Optionally, you can include an object prefix to separate uploads, such as:

```
https://swift-cluster.example.com/v1/my_account/container/OBJECT_PREFIX
```

❷ **method="POST"**

Must be POST.

❸ **enctype="multipart/form-data"**

Must be `multipart/form-data`.

❹ **name="redirect" value="*REDIRECT_URL*"**

Redirects the browser to the *REDIRECT_URL* after the upload completes. The URL has status and message query parameters added to it, which specify the HTTP status code for the upload and an optional error message. The 2*nn* status code indicates success.

The *REDIRECT_URL* can be an empty string. If so, the `Location` response header is not set.

❺ **name="max_file_size" value="*BYTES*"**

Required. Indicates the size, in bytes, of the maximum single file upload.

❻ **name="max_file_count" value= "*COUNT*"**

Required. Indicates the maximum number of files that can be uploaded with the form.

❼ **name="expires" value="*UNIX_TIMESTAMP*"**

The UNIX timestamp that specifies the time before which the form must be submitted before it becomes no longer valid.

❽ **name="signature" value="*HMAC*"**

The HMAC-SHA1 signature of the form. See the section called “HMAC-SHA1 signature for form POST” [28].

❾ **type="file" name="*FILE_NAME*"**

File name of the file to be uploaded. You can include from one to the `max_file_count` value of files.

The file attributes must appear *after* the other attributes to be processed correctly.

If attributes appear after the file attributes, they are not sent with the sub-request because all attributes in the file cannot be parsed on the server side unless the whole file is read into memory; the server does not have enough memory to service these requests. Attributes that follow the file attributes are ignored.

❿ **type= "submit"**

Must be `submit`.

HMAC-SHA1 signature for form POST

Form **POST** middleware uses an HMAC-SHA1 cryptographic signature. This signature includes these elements from the form:

- The path. Starting with `/v1/` onwards and including a container name and, optionally, an object prefix. In Example 1.15, "HMAC-SHA1 signature for form POST" [29], the path is `/v1/my_account/container/object_prefix`. Do not URL-encode the path at this stage.
- A redirect URL. If there is no redirect URL, use the empty string.
- Maximum file size. In Example 1.15, "HMAC-SHA1 signature for form POST" [29], the `max_file_size` is `104857600` bytes.
- The maximum number of objects to upload. In Example 1.15, "HMAC-SHA1 signature for form POST" [29], `max_file_count` is `10`.
- Expiry time. In Example 1.15, "HMAC-SHA1 signature for form POST" [29], the expiry time is set to `600` seconds into the future.
- The secret key. Set as the `X-Account-Meta-Temp-URL-Key` header value.

The following example code generates a signature for use with form **POST**:

Example 1.15. HMAC-SHA1 signature for form POST

```
import hmac
from hashlib import sha1
from time import time
path = '/v1/my_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()
```

For more information, see RFC 2104: HMAC: Keyed-Hashing for Message Authentication.

Form POST example

The following example shows how to submit a form by using a cURL command. In this example, the object prefix is `photos/` and the file being uploaded is called `flower.jpg`.

This example uses the **swift-form-signature** script to compute the `expires` and `signature` values.

```
$ bin/swift-form-signature /v1/my_account/container/photos/ https://example.
com/done.html 5373952000 1 200 MYKEY
Expires: 1390825338
Signature: 35129416ebda2f1a21b3c2b8939850dfc63d8f43
```

```
$ curl -i https://swift-cluster.example.com/v1/my_account/container/photos/ -X
 POST \
       -F max_file_size=5373952000 -F max_file_count=1 -F expires=1390825338 \
       -F signature=35129416ebda2f1a21b3c2b8939850dfc63d8f43 \
       -F redirect=https://example.com/done.html \
       -F file=@flower.jpg
```

Create static website

To discover whether your Object Storage system supports this feature, see the section called "Discoverability" [4]. Alternatively, check with your service provider.

You can use your Object Storage account to create a static website. This mode is normally only active for anonymous requests, which provide no authentication token. To use it with authenticated requests, set the header `X-Web-Mode` to TRUE on the request. To determine whether the static website feature is enabled, contact your service provider.

For example:

```
[DEFAULT]
...

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

...

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

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

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

Static web middleware through swift

Example 1.16. Make container publicly readable

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

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

Example 1.17. Set site index file

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

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

Example 1.18. Enable file listing

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

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

Example 1.19. Enable CSS for file listing

Style the file listing.

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

Set error pages for static website

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

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

Set the `X-Container-Meta-Web-Error` metadata once for your entire static website.

Example 1.20. Set error pages for static website request

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

Any 2nn response indicates success.

2. Object Storage API operations

Accounts 32
Containers 42
Objects 56

Manage the accounts, containers, and objects in the Object Storage system.

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

Method	URI	Description
	Accounts	
GET	`/v1/{account}{?limit,marker, end_marker,format,prefix, delimiter}`	Shows details for a specified account and lists containers, sorted by name, in the account.
POST	`/v1/{account}`	Creates, updates, or deletes account metadata.
HEAD	`/v1/{account}`	Shows metadata for a specified account.
	Containers	
GET	`/v1/{account}/{container}{?limit, marker,end_marker,prefix,format, delimiter,path}`	Shows details for a specified container and lists objects, sorted by name, in the container.
PUT	`/v1/{account}/{container}`	Creates a container.
DELETE	`/v1/{account}/{container}`	Deletes an empty container.
POST	`/v1/{account}/{container}`	Creates, updates, or deletes custom metadata for a container.
HEAD	`/v1/{account}/{container}`	Shows container metadata, including the number of objects and the total bytes of all objects stored in the container.
	Objects	
GET	`/v1/{account}/{container}/{object} {?signature,expires,multipart-manifest}`	Downloads the object content and gets the object metadata.
PUT	`/v1/{account}/{container}/{object} {?multipart-manifest,signature, expires}`	Creates a new object with specified data content and metadata, or replaces an existing object with specified data content and metadata.
COPY	`/v1/{account}/{container}/{object}`	Copies an object to another object in the object store.
DELETE	`/v1/{account}/{container}/{object} {?multipart-manifest}`	Permanently deletes an object from the object store.
HEAD	`/v1/{account}/{container}/{object} {?signature,expires}`	Shows object metadata.
POST	`/v1/{account}/{container}/{object}`	Creates or updates object metadata.

Accounts

List containers for a specified account. Create, update, and delete account metadata. Show account metadata.

Method	URI	Description
GET	`/v1/{account}{?limit,marker, end_marker,format,prefix, delimiter}`	Shows details for a specified account and lists containers, sorted by name, in the account.

Method	URI	Description
POST	`/v1/{account}`	Creates, updates, or deletes account metadata.
HEAD	`/v1/{account}`	Shows metadata for a specified account.

Show account details and list containers

Method	URI	Description
GET	`/v1/{account}{?limit,marker, end_marker,format,prefix, delimiter}`	Shows details for a specified account and lists containers, sorted by name, in the account.

The sort order for the name is based on a binary comparison, a single built-in collating sequence that compares string data by using the SQLite `memcmp( )` function, regardless of text encoding.

This operation does not accept a request body.

Example requests and responses:

- Show account details and list containers, and ask for a JSON response:

  ```
  curl -i $publicURL?format=json -X GET -H "X-Auth-Token: $token"
  ```

  See the example response below.

- List containers and ask for an XML response: `curl -i $publicURL?format=xml -X GET -H "X-Auth-Token: $token"`

  See the example response below.

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

The response body returns a list of containers. The default response (`text/plain`) returns one container per line.

If you use query parameters to page through a long list of containers, you have reached the end of the list if the number of items in the returned list is less than the request `limit` value. The list contains more items if the number of items in the returned list equals the `limit` value.

If the request succeeds, the operation returns one of these status codes:

- 200. Success. The response body lists the containers.
- 204. Success. The response body shows no containers. Either the account has no containers or you are paging through a long list of names by using the `marker`, `limit`, or `end_marker` query parameters, and you have reached the end of the list.

Normal response codes: 200, 204

Request

This table shows the header parameters for the show account details and list containers request:

Name	Type	Description
`X-Auth-Token`	String	Authentication token.

Name	Type	Description
	(Required)	
`X-Newest`	Boolean *(Optional)*	If set to `True`, Object Storage queries all replicas to return the most recent one. If you omit this header, Object Storage responds faster after it finds one valid replica. Because setting this header to `True` is more expensive for the back end, use it only when it is absolutely needed.
`Accept`	String *(Optional)*	Instead of using the `format` query parameter, set this header to `application/json`, `application/xml`, or `text/xml`.

This table shows the URI parameters for the show account details and list containers request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.

This table shows the query parameters for the show account details and list containers request:

Name	Type	Description
`limit`	Int *(Optional)*	For an integer value *n*, limits the number of results to *n*.
`marker`	String *(Optional)*	For a string value *x*, returns container names that are greater in value than the specified marker.
`end_marker`	String *(Optional)*	For a string value *x*, returns container names that are less in value than the specified marker.
`format`	String *(Optional)*	The response format. Valid values are `json`, `xml`, or `plain`. The default is `plain`. If you append the `format=xml` or `format=json` query parameter to the storage account URL, the response shows extended container information serialized in the specified format. If you append the `format=plain` query parameter, the response lists the container names separated by newlines.
`prefix`	String *(Optional)*	Prefix value. Object names in the response begin with this value.
`delimiter`	Char *(Optional)*	Delimiter value, which returns the object names that are nested in the container.

Response

This table shows the header parameters for the show account details and list containers response:

Name	Type	Description
`Content-Length`	String *(Required)*	The length of the response body that contains the list of names. If the operation fails, this value is the length of the error text in the response body.
`Content-Type`	String	The MIME type of the list of names. If the operation fails, this value is the MIME type of the error text in the response body.

Name	Type	Description
	(Required)	
`X-Account-Object-Count`	Int *(Required)*	The number of objects in the account.
`X-Account-Bytes-Used`	Int *(Required)*	The total number of bytes that are stored in Object Storage for the account.
`X-Account-Container-Count`	Int *(Required)*	The number of containers.
`X-Account-Meta-name`	String *(Optional)*	The custom account metadata item, where `{name}` is the name of the metadata item. One `X-Account-Meta-{name}` response header appears for each metadata item (for each `{name}`).
`X-Account-Meta-Temp-URL-Key`	String *(Optional)*	The secret key value for temporary URLs. If not set, this header is not returned by this operation.
`X-Account-Meta-Temp-URL-Key-2`	String *(Optional)*	A second secret key value for temporary URLs. If not set, this header is not returned by this operation.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Example 2.1. List containers response: HTTP and JSON

```
HTTP/1.1 200 OK
Content-Length: 96
X-Account-Object-Count: 1
X-Timestamp: 1389453423.35964
X-Account-Meta-Subject: Literature
X-Account-Bytes-Used: 14
X-Account-Container-Count: 2
Content-Type: application/json; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: tx274a77a8975c4a66aeb24-0052d95365
Date: Fri, 17 Jan 2014 15:59:33 GMT
```

```
[
   {
      "count":0,
      "bytes":0,
      "name":"janeausten"
   },
   {
      "count":1,
      "bytes":14,
      "name":"marktwain"
   }
]
```

Example 2.2. List containers response: HTTP and XML

```
HTTP/1.1 200 OK
Content-Length: 262
```

```
X-Account-Object-Count: 1
X-Timestamp: 1389453423.35964
X-Account-Meta-Subject: Literature
X-Account-Bytes-Used: 14
X-Account-Container-Count: 2
Content-Type: application/xml; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: tx69f60bc9f7634a01988e6-0052d9544b
Date: Fri, 17 Jan 2014 16:03:23 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<account name="my_account">
    <container>
        <name>janeausten</name>
        <count>0</count>
        <bytes>0</bytes>
    </container>
    <container>
        <name>marktwain</name>
        <count>1</count>
        <bytes>14</bytes>
    </container>
</account>
```

This operation does not return a response body.

Create, update, or delete account metadata

Method	URI	Description
POST	/v1/{account}	Creates, updates, or deletes account metadata.

To create, update, or delete metadata, use the `X-Account-Meta-{name}` header, where `{name}` is the name of the metadata item.

Subsequent requests for the same key and value pair overwrite the previous value.

To delete a metadata header, send an empty value for that particular header, such as for the `X-Account-Meta-Book` header. If the tool you use to communicate with Object Storage, such as an older version of cURL, does not support empty headers, send the `X-Remove-Account-Meta-{name}: *arbitrary value*` header. For example, `X-Remove-Account-Meta-Book: x`. The operation ignores the arbitrary value.

If the container already has other custom metadata items, a request to create, update, or delete metadata does not affect those items.

This operation does not accept a request body.

Example requests and responses:

- Create account metadata:

```
curl -i $publicURL -X POST -H "X-Auth-Token: $token" -H "X-
Account-Meta-Book: MobyDick" -H "X-Account-Meta-Subject:
Literature"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx8c2dd6aee35442a4a5646-0052d954fb
Date: Fri, 17 Jan 2014 16:06:19 GMT
```

- Update account metadata:

```
curl -i $publicURL -X POST -H "X-Auth-Token: $token" -H "X-
Account-Meta-Subject: AmericanLiterature"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx1439b96137364ab581156-0052d95532
Date: Fri, 17 Jan 2014 16:07:14 GMT
```

- Delete account metadata:

```
curl -i $publicURL -X POST -H "X-Auth-Token: $token" -H "X-
Remove-Account-Meta-Subject: x"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
```

```
X-Trans-Id: tx411cf57701424da99948a-0052d9556f
Date: Fri, 17 Jan 2014 16:08:15 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

If the request succeeds, the operation returns the 204 status code.

To confirm your changes, issue a show account metadata request.

Normal response codes: 204

Request

This table shows the header parameters for the create, update, or delete account metadata request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	Authentication token.
`X-Account-Meta-Temp-URL-Key`	String *(Optional)*	The secret key value for temporary URLs.
`X-Account-Meta-Temp-URL-Key-2`	String *(Optional)*	A second secret key value for temporary URLs. The second key enables you to rotate keys by having an old and new key active at the same time.
`X-Account-Meta-name`	String *(Optional)*	The account metadata. The `{name}` is the name of metadata item that you want to add, update, or delete. To delete this item, send an empty value in this header. You must specify a `X-Account-Meta-{name}` header for each metadata item (for each `{name}`) that you want to add, update, or delete.
`Content-Type`	String *(Optional)*	Changes the MIME type for the object.
`X-Detect-Content-Type`	Boolean *(Optional)*	If set to `true`, Object Storage guesses the content type based on the file extension and ignores the value sent in the `Content-Type` header, if present.

This table shows the URI parameters for the create, update, or delete account metadata request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.

Response

This table shows the header parameters for the create, update, or delete account metadata response:

Name	Type	Description
`Content-Length`	String *(Required)*	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.

Name	Type	Description
`Content-Type`	String *(Required)*	If the operation fails, this value is the MIME type of the error text in the response body.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Show account metadata

Method	URI	Description
HEAD	`/v1/{account}`	Shows metadata for a specified account.

Metadata for the account includes:

- Number of containers
- Number of objects
- Total number of bytes that are stored in Object Storage for the account

Because the storage system can store large amounts of data, take care when you represent the total bytes response as an integer; when possible, convert it to a 64-bit unsigned integer if your platform supports that primitive type.

This operation does not accept a request body.

Do not include metadata headers in this request.

Show account metadata request:

```
curl -i $publicURL -X HEAD -H "X-Auth-Token: $token"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
X-Account-Object-Count: 1
X-Account-Meta-Book: MobyDick
X-Timestamp: 1389453423.35964
X-Account-Bytes-Used: 14
X-Account-Container-Count: 2
Content-Type: text/plain; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: txafb3504870144b8ca40f7-0052d955d4
Date: Fri, 17 Jan 2014 16:09:56 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

If the account or authentication token is not valid, the operation returns the 401 Unauthorized error code.

Normal response codes: 204

Error response codes: unauthorized (401)

Request

This table shows the header parameters for the show account metadata request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	Authentication token.
`X-Newest`	Boolean	If set to `True`, Object Storage queries all replicas to return the most recent one. If you omit this header, Object Storage responds faster

Name	Type	Description
	(Optional)	after it finds one valid replica. Because setting this header to `True` is more expensive for the back end, use it only when it is absolutely needed.

This table shows the URI parameters for the show account metadata request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.

Response

This table shows the header parameters for the show account metadata response:

Name	Type	Description
`X-Account-Object-Count`	Int *(Required)*	The number of objects in the account.
`X-Account-Container-Count`	Int *(Required)*	The number of containers.
`X-Account-Bytes-Used`	Int *(Required)*	The total number of bytes that are stored in Object Storage for the account.
`X-Account-Meta-name`	String *(Optional)*	The custom account metadata item, where `{name}` is the name of the metadata item. One `X-Account-Meta-{name}` response header appears for each metadata item (for each `{name}`).
`X-Account-Meta-Temp-URL-Key`	String *(Optional)*	The secret key value for temporary URLs. If not set, this header is not returned by this operation.
`X-Account-Meta-Temp-URL-Key-2`	String *(Optional)*	A second secret key value for temporary URLs. If not set, this header is not returned by this operation.
`Content-Length`	String *(Required)*	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.
`Content-Type`	String *(Required)*	If the operation fails, this value is the MIME type of the error text in the response body.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Containers

List objects in a specified container. Create, show details for, and delete containers. Create, update, show, and delete container metadata.

Method	URI	Description
GET	`/v1/{account}/{container}{?limit, marker,end_marker,prefix,format, delimiter,path}`	Shows details for a specified container and lists objects, sorted by name, in the container.

Method	URI	Description
PUT	`/v1/{account}/{container}`	Creates a container.
DELETE	`/v1/{account}/{container}`	Deletes an empty container.
POST	`/v1/{account}/{container}`	Creates, updates, or deletes custom metadata for a container.
HEAD	`/v1/{account}/{container}`	Shows container metadata, including the number of objects and the total bytes of all objects stored in the container.

Show container details and list objects

Method	URI	Description
GET	`/v1/{account}/{container}{?limit, marker,end_marker,prefix,format, delimiter,path}`	Shows details for a specified container and lists objects, sorted by name, in the container.

Specify query parameters in the request to filter the list and return a subset of object names. Omit query parameters to return the complete list of object names that are stored in the container, up to 10,000 names. The 10,000 maximum value is configurable. To view the value for the cluster, issue a **GET** `/info` request.

Example requests and responses:

- Show container details for and list objects in the `marktwain` container, and ask for a JSON response:

  ```
  curl -i $publicURL/marktwain?format=json -X GET -H "X-Auth-Token: $token"
  ```

- Show container details for and list objects in the `marktwain` container, and ask for an XML response:

  ```
  curl -i $publicURL/marktwain?format=xml -X GET -H "X-Auth-Token: $token"
  ```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

If you use query parameters to page through a long list of objects, you have reached the end of the list if the number of items in the returned list is less than the request `limit` value. The list contains more items if the number of items in the returned list equals the `limit` value.

If the request succeeds, the operation returns one of these status codes:

- 200. Success. The response body lists the objects.
- 204. Success. The response body shows no objects. Either the container has no objects or you are paging through a long list of names by using the `marker`, `limit`, or `end_marker` query parameters, and you have reached the end of the list.

If the container does not exist, the 404 Not Found error code is returned.

Normal response codes: 200, 204

Error response codes: NotFound (404)

Request

This table shows the header parameters for the show container details and list objects request:

Name	Type	Description
`X-Auth-Token`	String	Authentication token.

Name	Type	Description
	(Required)	
`X-Newest`	Boolean *(Optional)*	If set to `True`, Object Storage queries all replicas to return the most recent one. If you omit this header, Object Storage responds faster after it finds one valid replica. Because setting this header to `True` is more expensive for the back end, use it only when it is absolutely needed.
`Accept`	String *(Optional)*	Instead of using the `format` query parameter, set this header to `application/json`, `application/xml`, or `text/xml`.

This table shows the URI parameters for the show container details and list objects request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.

This table shows the query parameters for the show container details and list objects request:

Name	Type	Description
`limit`	Int *(Optional)*	For an integer value *n*, limits the number of results to *n*.
`marker`	String *(Optional)*	For a string value *x*, returns container names that are greater in value than the specified marker.
`end_marker`	String *(Optional)*	For a string value *x*, returns container names that are less in value than the specified marker.
`prefix`	String *(Optional)*	Prefix value. Object names in the response begin with this value.
`format`	String *(Optional)*	The response format. Valid values are `json`, `xml`, or `plain`. The default is `plain`. If you append the `format=xml` or `format=json` query parameter to the storage account URL, the response shows extended container information serialized in the specified format. If you append the `format=plain` query parameter, the response lists the container names separated by newlines.
`delimiter`	Char *(Optional)*	Delimiter value, which returns the object names that are nested in the container.
`path`	String *(Optional)*	For a string value, returns the object names that are nested in the pseudo path. Equivalent to setting delimiter to `/` and `prefix` to the path with a `/` at the end.

Response

This table shows the header parameters for the show container details and list objects response:

Name	Type	Description
`Content-Length`	String *(Required)*	The length of the response body that contains the list of names. If the operation fails, this value is the length of the error text in the response body.

Name	Type	Description
X-Container-Object-Count	Int *(Required)*	The number of objects.
Accept-Ranges	String *(Required)*	The type of ranges that the object accepts.
X-Container-Meta-name	String *(Required)*	The custom container metadata item, where {name} is the name of the metadata item. One X-Container-Meta-{name} response header appears for each metadata item (for each {name}).
X-Container-Bytes-Used	Int *(Required)*	The count of bytes used in total.
Content-Type	String *(Required)*	The MIME type of the list of names. If the operation fails, this value is the MIME type of the error text in the response body.
X-Trans-Id	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
Date	Datetime *(Required)*	The transaction date and time.

Example 2.3. Show container details response: HTTP and JSON

```
HTTP/1.1 200 OK
Content-Length: 341
X-Container-Object-Count: 2
Accept-Ranges: bytes
X-Container-Meta-Book: TomSawyer
X-Timestamp: 1389727543.65372
X-Container-Bytes-Used: 26
Content-Type: application/json; charset=utf-8
X-Trans-Id: tx26377fe5fab74869825d1-0052d6bdff
Date: Wed, 15 Jan 2014 16:57:35 GMT
```

```
[
   {
      "hash":"451e372e48e0f6b1114fa0724aa79fa1",
      "last_modified":"2014-01-15T16:41:49.390270",
      "bytes":14,
      "name":"goodbye",
      "content_type":"application/octet-stream"
   },
   {
      "hash":"ed076287532e86365e841e92bfc50d8c",
      "last_modified":"2014-01-15T16:37:43.427570",
      "bytes":12,
      "name":"helloworld",
      "content_type":"application/octet-stream"
   }
]
```

Example 2.4. Show container details response: HTTP and XML

```
HTTP/1.1 200 OK
Content-Length: 500
X-Container-Object-Count: 2
```

```
Accept-Ranges: bytes
X-Container-Meta-Book: TomSawyer
X-Timestamp: 1389727543.65372
X-Container-Bytes-Used: 26
Content-Type: application/xml; charset=utf-8
X-Trans-Id: txc75ea9a6e66f47d79e0c5-0052d6be76
Date: Wed, 15 Jan 2014 16:59:35 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<container name="marktwain">
    <object>
        <name>goodbye</name>
        <hash>451e372e48e0f6b1114fa0724aa79fa1</hash>
        <bytes>14</bytes>
        <content_type>application/octet-stream</content_type>
        <last_modified>2014-01-15T16:41:49.390270</last_modified>
    </object>
    <object>
        <name>helloworld</name>
        <hash>ed076287532e86365e841e92bfc50d8c</hash>
        <bytes>12</bytes>
        <content_type>application/octet-stream</content_type>
        <last_modified>2014-01-15T16:37:43.427570</last_modified>
    </object>
</container>
```

This operation does not return a response body.

Create container

Method	URI	Description
PUT	/v1/{account}/{container}	Creates a container.

You do not need to check if a container already exists before issuing a **PUT** operation because the operation is idempotent: It creates a container or updates an existing container, as appropriate.

Example requests and responses:

- Create a container with no metadata: `curl -i $publicURL/steven -X PUT -H "Content-Length: 0" -H "X-Auth-Token: $token"`

```
HTTP/1.1 201 Created
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx7f6b7fa09bc2443a94df0-0052d58b56
Date: Tue, 14 Jan 2014 19:09:10 GMT
```

- Create a container with metadata:

```
curl -i $publicURL/marktwain -X PUT -H "X-Auth-Token: $token" -H "X-Container-Meta-Book: TomSawyer"
```

```
HTTP/1.1 201 Created
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx06021f10fc8642b2901e7-0052d58f37
Date: Tue, 14 Jan 2014 19:25:43 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

Normal response codes: 201, 204

Request

This table shows the header parameters for the create container request:

Name	Type	Description
X-Auth-Token	String *(Required)*	Authentication token.
X-Container-Read	String *(Optional)*	Sets an ACL that grants read access.
X-Container-Write	String *(Optional)*	Sets an ACL that grants write access.
X-Container-Sync-To	String *(Optional)*	Sets the destination for container synchronization.
X-Container-Sync-Key	String *(Optional)*	Sets the secret key for container synchronization.

Name	Type	Description
`X-Versions-Location`	String *(Optional)*	Enables versioning on this container. The value is the name of another container. You must UTF-8-encode and then URL-encode the name before you include it in the header. To disable versioning, set the header to an empty string.
`X-Container-Meta-name`	String *(Optional)*	The container metadata, where `{name}` is the name of metadata item. You must specify a `X-Container-Meta-{name}` header for each metadata item (for each `{name}`) that you want to add or update.
`Content-Type`	String *(Optional)*	Changes the MIME type for the object.
`X-Detect-Content-Type`	Boolean *(Optional)*	If set to `true`, Object Storage guesses the content type based on the file extension and ignores the value sent in the `Content-Type` header, if present.

This table shows the URI parameters for the create container request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.

Response

This table shows the header parameters for the create container response:

Name	Type	Description
`Content-Length`	String *(Required)*	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.
`Content-Type`	String *(Required)*	If the operation fails, this value is the MIME type of the error text in the response body.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Delete container

Method	URI	Description
DELETE	`/v1/{account}/{container}`	Deletes an empty container.

This operation fails unless the container is empty. An empty container has no objects.

Delete the `steven` container:

```
curl -i $publicURL/steven -X DELETE -H "X-Auth-Token: $token"
```

If the container does not exist, the response is:

```
HTTP/1.1 404 Not Found
Content-Length: 70
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx4d728126b17b43b598bf7-0052d81e34
Date: Thu, 16 Jan 2014 18:00:20 GMT
```

If the container exists and the deletion succeeds, the response is:

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txf76c375ebece4df19c84c-0052d81f14
Date: Thu, 16 Jan 2014 18:04:04 GMT
```

If the container exists but is not empty, the response is:

```
HTTP/1.1 409 Conflict
Content-Length: 95
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx7782dc6a97b94a46956b5-0052d81f6b
Date: Thu, 16 Jan 2014 18:05:31 GMT

<html><h1>Conflict</h1><p>There was a conflict when trying to complete your
 request.</p></html>
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

Normal response codes: 204

Error response codes: NotFound (404), Conflict (409)

Request

This table shows the header parameters for the delete container request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	Authentication token.

This table shows the URI parameters for the delete container request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.

Response

This table shows the header parameters for the delete container response:

Name	Type	Description
`Content-Length`	String *(Required)*	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.
`Content-Type`	String *(Required)*	If the operation fails, this value is the MIME type of the error text in the response body.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Create, update, or delete container metadata

Method	URI	Description
POST	`/v1/{account}/{container}`	Creates, updates, or deletes custom metadata for a container.

To create, update, or delete a custom metadata item, use the `X-Container-Meta-{name}` header, where `{name}` is the name of the metadata item.

Subsequent requests for the same key and value pair overwrite the previous value.

To delete container metadata, send an empty value for that header, such as for the `X-Container-Meta-Book` header. If the tool you use to communicate with Object Storage, such as an older version of cURL, does not support empty headers, send the `X-Remove-Container-Meta-{name}: *arbitrary value*` header. For example, `X-Remove-Container-Meta-Book: x`. The operation ignores the arbitrary value.

If the container already has other custom metadata items, a request to create, update, or delete metadata does not affect those items.

This operation does not accept a request body.

Example requests and responses:

- Create container metadata:

```
curl -i $publicURL/marktwain -X POST -H "X-Auth-Token: $token"
-H "X-Container-Meta-Author: MarkTwain" -H "X-Container-Meta-
Century: Nineteenth"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx05dbd434c651429193139-0052d82635
Date: Thu, 16 Jan 2014 18:34:29 GMT
```

- Update container metadata:

```
curl -i $publicURL/marktwain -X POST -H "X-Auth-Token: $token" -H
"X-Container-Meta-Author: SamuelClemens"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txe60c7314bf614bb39dfe4-0052d82653
Date: Thu, 16 Jan 2014 18:34:59 GMT
```

- Delete container metadata:

```
curl -i $publicURL/marktwain -X POST -H "X-Auth-Token: $token" -H
"X-Remove-Container-Meta-Century: x"
```

```
HTTP/1.1 204 No Content
```

```
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx7997e18da2a34a9e84ceb-0052d826d0
Date: Thu, 16 Jan 2014 18:37:04 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

If the request succeeds, the operation returns the 204 status code.

To confirm your changes, issue a show container metadata request.

Normal response codes: 204

Request

This table shows the header parameters for the create, update, or delete container metadata request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	Authentication token.
`X-Container-Read`	String *(Optional)*	Sets an ACL that grants read access.
`X-Remove-Container-name`	String *(Optional)*	Removes the metadata item named `{name}`. For example, `X-Remove-Container-Read` removes the `X-Container-Read` metadata item.
`X-Container-Write`	String *(Optional)*	Sets an ACL that grants write access.
`X-Container-Sync-To`	String *(Optional)*	Sets the destination for container synchronization.
`X-Container-Sync-Key`	String *(Optional)*	Sets the secret key for container synchronization.
`X-Versions-Location`	String *(Optional)*	Enables versioning on this container. The value is the name of another container. You must UTF-8-encode and then URL-encode the name before you include it in the header. To disable versioning, set the header to an empty string.
`X-Remove-Versions-Location`	String *(Optional)*	Set to any value to disable versioning.
`X-Container-Meta-name`	String *(Optional)*	The container metadata, where `{name}` is the name of metadata item. You must specify a `X-Container-Meta-{name}` header for each metadata item (for each `{name}`) that you want to add or update.
`Content-Type`	String *(Optional)*	Changes the MIME type for the object.
`X-Detect-Content-Type`	Boolean *(Optional)*	If set to `true`, Object Storage guesses the content type based on the file extension and ignores the value sent in the `Content-Type` header, if present.

This table shows the URI parameters for the create, update, or delete container metadata request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.

Response

This table shows the header parameters for the create, update, or delete container metadata response:

Name	Type	Description
`Content-Length`	String *(Required)*	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.
`Content-Type`	String *(Required)*	If the operation fails, this value is the MIME type of the error text in the response body.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Show container metadata

Method	URI	Description
HEAD	`/v1/{account}/{container}`	Shows container metadata, including the number of objects and the total bytes of all objects stored in the container.

Show container metadata request:

```
curl -i $publicURL/marktwain -X HEAD -H "X-Auth-Token: $token"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
X-Container-Object-Count: 1
Accept-Ranges: bytes
X-Container-Meta-Book: TomSawyer
X-Timestamp: 1389727543.65372
X-Container-Meta-Author: SamuelClemens
X-Container-Bytes-Used: 14
Content-Type: text/plain; charset=utf-8
X-Trans-Id: tx0287b982a268461b9ec14-0052d826e2
Date: Thu, 16 Jan 2014 18:37:22 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

If the request succeeds, the operation returns the 204 status code.

Normal response codes: 204

Request

This table shows the header parameters for the show container metadata request:

Name	Type	Description
`X-Auth-Token`	String *(Optional)*	Authentication token. If you omit this header, your request fails unless the account owner has granted you access through an access control list (ACL).
`X-Newest`	Boolean *(Optional)*	If set to `True`, Object Storage queries all replicas to return the most recent one. If you omit this header, Object Storage responds faster after it finds one valid replica. Because setting this header to `True` is more expensive for the back end, use it only when it is absolutely needed.

This table shows the URI parameters for the show container metadata request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.

Response

This table shows the header parameters for the show container metadata response:

Name	Type	Description
`Content-Length`	String	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.

Name	Type	Description
	(Required)	
`X-Container-Object-Count`	Int *(Required)*	The number of objects.
`Accept-Ranges`	String *(Required)*	The type of ranges that the object accepts.
`X-Container-Meta-name`	String *(Required)*	The custom container metadata item, where `{name}` is the name of the metadata item. One `X-Container-Meta-{name}` response header appears for each metadata item (for each `{name}`).
`X-Container-Bytes-Used`	Int *(Required)*	The count of bytes used in total.
`X-Container-Read`	String *(Optional)*	The ACL that grants read access. If not set, this header is not returned by this operation.
`X-Container-Write`	String *(Optional)*	The ACL that grants write access. If not set, this header is not returned by this operation.
`X-Container-Sync-To`	String *(Optional)*	The destination for container synchronization. If not set, this header is not returned by this operation.
`X-Container-Sync-Key`	String *(Optional)*	The secret key for container synchronization. If not set, this header is not returned by this operation.
`X-Versions-Location`	String *(Required)*	Enables versioning on this container. The value is the name of another container. You must UTF-8-encode and then URL-encode the name before you include it in the header. To disable versioning, set the header to an empty string.
`Content-Type`	String *(Required)*	If the operation fails, this value is the MIME type of the error text in the response body.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Objects

Create, replace, show details for, and delete objects. Copy objects with another object with a new or different name. Update object metadata.

Method	URI	Description
GET	`/v1/{account}/{container}/{object}{?signature,expires,multipart-manifest}`	Downloads the object content and gets the object metadata.
PUT	`/v1/{account}/{container}/{object}{?multipart-manifest,signature,expires}`	Creates a new object with specified data content and metadata, or replaces an existing object with specified data content and metadata.
COPY	`/v1/{account}/{container}/{object}`	Copies an object to another object in the object store.
DELETE	`/v1/{account}/{container}/{object}{?multipart-manifest}`	Permanently deletes an object from the object store.

Method	URI	Description
HEAD	`/v1/{account}/{container}/{object}{?signature,expires}`	Shows object metadata.
POST	`/v1/{account}/{container}/{object}`	Creates or updates object metadata.

Get object content and metadata

Method	URI	Description
GET	/v1/{account}/{container}/{object}{?signature,expires,multipart-manifest}	Downloads the object content and gets the object metadata.

This operation returns the object metadata in the response headers and the object content in the response body.

If this is a large object, the response body contains the concatenated content of the segment objects. To get the manifest instead of concatenated segment objects for a static large object, use the `multipart-manifest` query parameter.

Example requests and responses:

- Show object details for the `goodbye` object in the `marktwain` container: `curl -i $publicURL/marktwain/goodbye -X GET -H "X-Auth-Token: $token"`

```
HTTP/1.1 200 OK
Content-Length: 14
Accept-Ranges: bytes
Last-Modified: Wed, 15 Jan 2014 16:41:49 GMT
Etag: 451e372e48e0f6b1114fa0724aa79fa1
X-Timestamp: 1389804109.39027
X-Object-Meta-Orig-Filename: goodbyeworld.txt
Content-Type: application/octet-stream
X-Trans-Id: tx8145a190241f4cf6b05f5-0052d82a34
Date: Thu, 16 Jan 2014 18:51:32 GMT

Goodbye World!
```

- Show object details for the `goodbye` object, which does not exist, in the `janeausten` container:

  `curl -i $publicURL/janeausten/goodbye -X GET -H "X-Auth-Token: $token"`

```
HTTP/1.1 404 Not Found
Content-Length: 70
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx073f7cbb850c4c99934b9-0052d82b04
Date: Thu, 16 Jan 2014 18:55:00 GMT

<html><h1>Not Found</h1><p>The resource could not be found.</p></html>
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

Normal response codes: 200

Error response codes: NotFound (404)

Request

This table shows the header parameters for the get object content and metadata request:

Name	Type	Description
`X-Auth-Token`	String *(Optional)*	Authentication token. If you omit this header, your request fails unless the account owner has granted you access through an access control list (ACL).
`X-Newest`	Boolean *(Optional)*	If set to `True`, Object Storage queries all replicas to return the most recent one. If you omit this header, Object Storage responds faster after it finds one valid replica. Because setting this header to `True` is more expensive for the back end, use it only when it is absolutely needed.
`Range`	Dict *(Optional)*	The ranges of content to get. You can use the `Range` header to get portions of data by using one or more range specifications. To specify many ranges, separate the range specifications with a comma. The types of range specifications are: • **Byte range specification**. Use FIRST_BYTE_OFFSET to specify the start of the data range, and LAST_BYTE_OFFSET to specify the end. You can omit the LAST_BYTE_OFFSET and if you do, the value defaults to the offset of the last byte of data. • **Suffix byte range specification**. Use LENGTH bytes to specify the length of the data range. The following forms of the header specify the following ranges of data: • `Range: bytes=-5`. The last five bytes. • `Range: bytes=10-15`. The five bytes of data after a 10-byte offset. • `Range: bytes=10-15,-5`. A multi-part response that contains the last five bytes and the five bytes of data after a 10-byte offset. The `Content-Type` of the response is then `multipart/byteranges`. • `Range: bytes=4-6`. Bytes 4 to 6 inclusive. • `Range: bytes=2-2`. Byte 2, the third byte of the data. • `Range: bytes=6-`. Byte 6 and after. • `Range: bytes=1-3,2-5`. A multi-part response that contains bytes 1 to 3 inclusive, and bytes 2 to 5 inclusive. The `Content-Type` of the response is then `multipart/byteranges`.
`If-Match`	Dict *(Optional)*	See http://www.ietf.org/rfc/rfc2616.txt.
`If-None-Match`	Dict *(Optional)*	See http://www.ietf.org/rfc/rfc2616.txt.
`If-Modified-Since`	Dict *(Optional)*	See http://www.ietf.org/rfc/rfc2616.txt.
`If-Unmodified-Since`	Dict *(Optional)*	See http://www.ietf.org/rfc/rfc2616.txt.

This table shows the URI parameters for the get object content and metadata request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.

Name	Type	Description
`{container}`	String	The unique name for the container.
`{object}`	String	The unique name for the object.

This table shows the query parameters for the get object content and metadata request:

Name	Type	Description
`signature`	String *(Optional)*	Used with temporary URLs to sign the request. For more information about temporary URLs, see *OpenStack Object Storage API v1 Reference* .
`expires`	String *(Optional)*	Used with temporary URLs to specify the expiry time of the signature. For more information about temporary URLs, see *OpenStack Object Storage API v1 Reference* .
`multipart-manifest`	String *(Optional)*	If you include the `multipart-manifest=get` query parameter and the object is a large object, the object contents are not returned. Instead, the manifest is returned in the `X-Object-Manifest` response header for dynamic large objects or in the response body for static large objects.

Response

This table shows the header parameters for the get object content and metadata response:

Name	Type	Description
`Content-Length`	String *(Required)*	The length of the object content in the response body, in bytes.
`Accept-Ranges`	String *(Required)*	The type of ranges that the object accepts.
`Last-Modified`	String *(Required)*	The date and time that the object was created or the last time that the metadata was changed.
`ETag`	String *(Required)*	For objects smaller than 5 GB, this value is the MD5 checksum of the object content. The value is not quoted. For manifest objects, this value is the MD5 checksum of the concatenated string of MD5 checksums and ETags for each of the segments in the manifest, and not the MD5 checksum of the content that was downloaded. Also the value is enclosed in double-quote characters. You are strongly recommended to compute the MD5 checksum of the response body as it is received and compare this value with the one in the ETag header. If they differ, the content was corrupted, so retry the operation.
`Content-Type`	String *(Required)*	The MIME type of the object.
`Content-Encoding`	String *(Optional)*	If set, the value of the `Content-Encoding` metadata. If not set, this header is not returned by this operation.
`Content-Disposition`	String *(Optional)*	If set, specifies the override behavior for the browser. For example, this header might specify that the browser use a download program to save this file rather than show the file, which is the default. If not set, this header is not returned by this operation.
`X-Delete-At`	String *(Optional)*	If set, the time when the object will be deleted by the system in the format of a UNIX Epoch timestamp.

Name	Type	Description
		If not set, this header is not returned by this operation.
`X-Object-Meta-name`	String *(Required)*	The custom object metadata item, where `{name}` is the name of the metadata item. One `X-Object-Meta-{name}` response header appears for each metadata item (for each `{name}`).
`X-Object-Manifest`	String *(Optional)*	If set, to this is a dynamic large object manifest object. The value is the container and object name prefix of the segment objects in the form `container/prefix`.
`X-Static-Large-Object`	Bool *(Required)*	Set to `True` if this object is a static large object manifest object.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Create or replace object

Method	URI	Description
PUT	/v1/{account}/{container}/{object}{?multipart-manifest,signature,expires}	Creates a new object with specified data content and metadata, or replaces an existing object with specified data content and metadata.

The **PUT** operation always creates a new object. If you use this operation on an existing object, you replace the existing object and metadata rather than modifying the object. Consequently, this operation returns a `201 Created` status code.

If you use this operation to copy a manifest object, the new object is a normal object and not a copy of the manifest. Instead it is a concatenation of all the segment objects. This means that you cannot copy objects larger than 5 GB.

Example requests and responses:

- Create object:

```
curl -i $publicURL/janeausten/helloworld.txt -X PUT -H "Content-
Length: 1" -H "Content-Type: text/html; charset=UTF-8" -H "X-
Auth-Token: $token"
```

```
HTTP/1.1 201 Created
Last-Modified: Fri, 17 Jan 2014 17:28:35 GMT
Content-Length: 116
Etag: d41d8cd98f00b204e9800998ecf8427e
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx4d5e4f06d357462bb732f-0052d96843
Date: Fri, 17 Jan 2014 17:28:35 GMT
```

- Replace object:

```
curl -i $publicURL/janeausten/helloworld -X PUT -H "Content-
Length: 0" -H "X-Auth-Token: $token"
```

```
HTTP/1.1 201 Created
Last-Modified: Fri, 17 Jan 2014 17:28:35 GMT
Content-Length: 116
Etag: d41d8cd98f00b204e9800998ecf8427e
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx4d5e4f06d357462bb732f-0052d96843
Date: Fri, 17 Jan 2014 17:28:35 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

The `201 Created` status code indicates a successful write.

If the request times out, the operation returns the `408 Request Timeout` error code.

The `411 Length Required` error code indicates a missing `Transfer-Encoding` or `Content-Length` request header.

If the MD5 checksum of the data that is written to the object store does not match the optional `ETag` value, the operation returns the `422 Unprocessable Entity` error code.

Normal response codes: 201

Error response codes: timeout (408), lengthRequired (411), unprocessableEntity (422)

Request

This table shows the header parameters for the create or replace object request:

Name	Type	Description
`X-Object-Manifest`	String *(Optional)*	Set to specify that this is a dynamic large object manifest object. The value is the container and object name prefix of the segment objects in the form `container/prefix`. You must UTF-8-encode and then URL-encode the names of the container and prefix before you include them in this header.
`X-Auth-Token`	String *(Optional)*	Authentication token. If you omit this header, your request fails unless the account owner has granted you access through an access control list (ACL).
`Content-Length`	Int *(Optional)*	Set to the length of the object content. Do not set if chunked transfer encoding is being used.
`Transfer-Encoding`	String *(Optional)*	Set to `chunked` to enable chunked transfer encoding. If used, do not set the `Content-Length` header to a non-zero value.
`Content-Type`	String *(Optional)*	Changes the MIME type for the object.
`X-Detect-Content-Type`	Boolean *(Optional)*	If set to `true`, Object Storage guesses the content type based on the file extension and ignores the value sent in the `Content-Type` header, if present.
`X-Copy-From`	String *(Optional)*	If set, this is the name of an object used to create the new object by copying the `X-Copy-From` object. The value is in form `{container}/{object}`. You must UTF-8-encode and then URL-encode the names of the container and object before you include them in the header. Using **PUT** with `X-Copy-From` has the same effect as using the **COPY** operation to copy an object.
`ETag`	String *(Optional)*	The MD5 checksum value of the request body. For example, the MD5 checksum value of the object content. You are strongly recommended to compute the MD5 checksum value of object content and include it in the request. This enables the Object Storage API to check the integrity of the upload. The value is not quoted.
`Content-Disposition`	String *(Optional)*	If set, specifies the override behavior for the browser. For example, this header might specify that the browser use a download program to save this file rather than show the file, which is the default.
`Content-Encoding`	String *(Optional)*	If set, the value of the `Content-Encoding` metadata.
`X-Delete-At`	Int *(Optional)*	The certain date, in the format of a UNIX Epoch timestamp, when the object is removed.
`X-Delete-After`	Int *(Optional)*	Specifies the number of seconds after which the object is removed. Internally, the Object Storage system stores this value in the `X-Delete-At` metadata item.
`X-Object-Meta-name`	String *(Optional)*	The container metadata, where `{name}` is the name of the metadata item. You must specify a `X-Object-Meta-{name}` header for each metadata item (for each `{name}`) that you want to add or update.

This table shows the URI parameters for the create or replace object request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.
`{object}`	String	The unique name for the object.

This table shows the query parameters for the create or replace object request:

Name	Type	Description
`multipart-manifest`	String *(Optional)*	If `?multipart-manifest=put`, the object is a static large object manifest and the body contains the manifest.
`signature`	String *(Optional)*	Used with temporary URLs to sign the request. For more information about temporary URLs, see *OpenStack Object Storage API v1 Reference* .
`expires`	String *(Optional)*	Used with temporary URLs to specify the expiry time of the signature. For more information about temporary URLs, see *OpenStack Object Storage API v1 Reference* .

Response

This table shows the header parameters for the create or replace object response:

Name	Type	Description
`Content-Length`	String *(Required)*	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.
`ETag`	String *(Required)*	For objects smaller than 5 GB, this value is the MD5 checksum of the uploaded object content. The value is not quoted. If you supplied an `ETag` request header and the operation was successful, the values are the same. If you did not supply an `ETag` request header, check the `ETag` response header value against the object content you have just uploaded. For static large objects, this value is the MD5 checksum of the concatenated string of MD5 checksums and ETags for each of the segments in the manifest, and not the MD5 checksum of the content that was uploaded. Also the value is enclosed in double-quotes. For dynamic large objects, the value is the MD5 checksum of the empty string.
`Content-Type`	String *(Required)*	The MIME type of the object.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Copy object

Method	URI	Description
COPY	/v1/{account}/{container}/{object}	Copies an object to another object in the object store.

You can copy an object to a new object with the same name. Copying to the same name is an alternative to using **POST** to add metadata to an object. With **POST**, you must specify all the metadata. With **COPY**, you can add additional metadata to the object.

Alternatively, you can use **PUT** with the `X-Copy-From` request header to accomplish the same operation as the **COPY** object operation.

The **PUT** operation always creates a new object. If you use this operation on an existing object, you replace the existing object and metadata rather than modifying the object. Consequently, this operation returns a `201 Created` success node.

If you use this operation to copy a manifest object, the new object is a normal object and not a copy of the manifest. Instead it is a concatenation of all the segment objects. This means that you cannot copy objects larger than 5 GB in size. All metadata is preserved during the object copy. If you specify metadata on the request to copy the object, either **PUT** or **COPY**, the metadata overwrites any conflicting keys on the target (new) object.

Example requests and responses:

- Copy the `goodbye` object from the `marktwain` container to the `janeausten` container: `curl -i $publicURL/marktwain/goodbye -X COPY -H "X-Auth-Token: $token" -H "Destination: janeausten/goodbye"`

```
HTTP/1.1 201 Created
Content-Length: 0
X-Copied-From-Last-Modified: Thu, 16 Jan 2014 21:19:45 GMT
X-Copied-From: marktwain/goodbye
Last-Modified: Fri, 17 Jan 2014 18:22:57 GMT
Etag: 451e372e48e0f6b1114fa0724aa79fa1
Content-Type: text/html; charset=UTF-8
X-Object-Meta-Movie: AmericanPie
X-Trans-Id: txdcb481ad49d24e9a81107-0052d97501
Date: Fri, 17 Jan 2014 18:22:57 GMT
```

- Alternatively, you can use **PUT** to copy the `goodbye` object from the `marktwain` container to the `janeausten` container. This request requires a `Content-Length` header even if it is set to zero (0).

  `curl -i $publicURL/janeausten/goodbye -X PUT -H "X-Auth-Token: $token" -H "X-Copy-From: /marktwain/goodbye" -H "Content-Length: 0"`

```
HTTP/1.1 201 Created
Content-Length: 0
X-Copied-From-Last-Modified: Thu, 16 Jan 2014 21:19:45 GMT
X-Copied-From: marktwain/goodbye
Last-Modified: Fri, 17 Jan 2014 18:22:57 GMT
Etag: 451e372e48e0f6b1114fa0724aa79fa1
Content-Type: text/html; charset=UTF-8
```

```
X-Object-Meta-Movie: AmericanPie
X-Trans-Id: txdcb481ad49d24e9a81107-0052d97501
Date: Fri, 17 Jan 2014 18:22:57 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

Note

When several replicas exist, the system copies from the most recent replica. That is, the **COPY** operation behaves as though the `X-Newest` header is in the request.

Normal response codes: 201

Request

This table shows the header parameters for the copy object request:

Name	Type	Description
`X-Auth-Token`	String *(Optional)*	Authentication token. If you omit this header, your request fails unless the account owner has granted you access through an access control list (ACL).
`Destination`	String *(Required)*	The container and object name of the destination object in the form of `/container/object`. You must UTF-8-encode and then URL-encode the names of the destination container and object before you include them in this header.
`Content-Type`	String *(Optional)*	Changes the MIME type for the object.
`Content-Encoding`	String *(Optional)*	If set, the value of the `Content-Encoding` metadata.
`Content-Disposition`	String *(Optional)*	If set, specifies the override behavior for the browser. For example, this header might specify that the browser use a download program to save this file rather than show the file, which is the default.
`X-Object-Meta-name`	String *(Optional)*	The container metadata, where `{name}` is the name of the metadata item. You must specify a `X-Object-Meta-{name}` header for each metadata item (for each `{name}`) that you want to add or update.

This table shows the URI parameters for the copy object request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.
`{object}`	String	The unique name for the object.

Response

This table shows the header parameters for the copy object response:

Name	Type	Description
`Content-Length`	String	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.

Name	Type	Description
	(Required)	
`X-Copied-From-Last-Modified`	String *(Optional)*	For a copied object, shows the last modified date and time for the container and object name from which the new object was copied.
`X-Copied-From`	String *(Optional)*	For a copied object, shows the container and object name from which the new object was copied. The value is in form `{container}/{object}`.
`Last-Modified`	String *(Required)*	The date and time that the object was created or the last time that the metadata was changed.
`ETag`	String *(Required)*	The MD5 checksum of the copied object content. The value is not quoted.
`Content-Type`	String *(Required)*	The MIME type of the object.
`X-Object-Meta-name`	String *(Required)*	The custom object metadata item, where `{name}` is the name of the metadata item. One `X-Object-Meta-{name}` response header appears for each metadata item (for each `{name}`).
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Delete object

Method	URI	Description
DELETE	`/v1/{account}/{container}/{object}{?multipart-manifest}`	Permanently deletes an object from the object store.

You can use the **COPY** method to copy the object to a new location. Then, use the **DELETE** method to delete the original object.

Object deletion occurs immediately at request time. Any subsequent **GET**, **HEAD**, **POST**, or **DELETE** operations return a `404 Not Found` error code.

For static large object manifests, you can add the `?multipart-manifest=delete` query parameter. This operation deletes the segment objects and if all deletions succeed, this operation deletes the manifest object.

Example request and response:

- Delete the `helloworld` object from the `marktwain` container: `curl -i $publicURL/marktwain/helloworld -X DELETE -H "X-Auth-Token: $token"`

```
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx36c7606fcd1843f59167c-0052d6fdac
Date: Wed, 15 Jan 2014 21:29:16 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

Normally the **DELETE** operation does not return a response body. However, with the `multipart-manifest=delete` query parameter, the response body contains a list of manifest and segment objects and the status of their delete operations.

Error response codes: 400, 500, ...

Request

This table shows the header parameters for the delete object request:

Name	Type	Description
`X-Auth-Token`	String *(Optional)*	Authentication token. If you omit this header, your request fails unless the account owner has granted you access through an access control list (ACL).

This table shows the URI parameters for the delete object request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.
`{object}`	String	The unique name for the object.

This table shows the query parameters for the delete object request:

Name	Type	Description
`multipart-manifest`	String *(Optional)*	If you include the `multipart-manifest=delete` query parameter and the object is a static large object, the segment objects and the manifest object are deleted. If you omit the `multipart-manifest=delete` query parameter and this is a static large object, the manifest object is deleted and the segment objects are not deleted. For a bulk delete, the response body looks the same as it does for a normal bulk delete. In contrast, a plain object **DELETE** response has an empty body.

Response

This table shows the header parameters for the delete object response:

Name	Type	Description
`Content-Length`	String *(Required)*	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.
`Content-Type`	String *(Required)*	The MIME type of the object.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Show object metadata

Method	URI	Description
HEAD	`/v1/{account}/{container}/{object}{?signature,expires}`	Shows object metadata.

If the `Content-Length` response header is non-zero, the example cURL command stalls after it prints the response headers because it is waiting for a response body. However, the Object Storage system does not return a response body for the **HEAD** operation.

Example requests and responses:

- Show object metadata:

```
curl -i $publicURL/marktwain/goodbye -X HEAD -H "X-Auth-Token:
$token"
```

```
HTTP/1.1 200 OK
Content-Length: 14
Accept-Ranges: bytes
Last-Modified: Thu, 16 Jan 2014 21:12:31 GMT
Etag: 451e372e48e0f6b1114fa0724aa79fa1
X-Timestamp: 1389906751.73463
X-Object-Meta-Book: GoodbyeColumbus
Content-Type: application/octet-stream
X-Trans-Id: tx37ea34dcd1ed48ca9bc7d-0052d84b6f
Date: Thu, 16 Jan 2014 21:13:19 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

If the request succeeds, the operation returns the 204 status code.

Normal response codes: 204

Request

This table shows the header parameters for the show object metadata request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	Authentication token.
`X-Newest`	Boolean *(Optional)*	If set to `True`, Object Storage queries all replicas to return the most recent one. If you omit this header, Object Storage responds faster after it finds one valid replica. Because setting this header to `True` is more expensive for the back end, use it only when it is absolutely needed.

This table shows the URI parameters for the show object metadata request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.

Name	Type	Description
`{object}`	String	The unique name for the object.

This table shows the query parameters for the show object metadata request:

Name	Type	Description
`signature`	String *(Optional)*	Used with temporary URLs to sign the request. For more information about temporary URLs, see *OpenStack Object Storage API v1 Reference* .
`expires`	String *(Optional)*	Used with temporary URLs to specify the expiry time of the signature. For more information about temporary URLs, see *OpenStack Object Storage API v1 Reference* .

Response

This table shows the header parameters for the show object metadata response:

Name	Type	Description
`Last-Modified`	String *(Required)*	The date and time that the object was created or the last time that the metadata was changed.
`Content-Length`	String *(Required)*	The length of the object content in the response body, in bytes.
`Content-Length`	String *(Required)*	**HEAD** operations do not return content. However, in this operation the value in the `Content-Length` header is not the size of the response body. Instead it contains the size of the object, in bytes.
`Content-Type`	String *(Required)*	The MIME type of the object.
`ETag`	String *(Required)*	For objects smaller than 5 GB, this value is the MD5 checksum of the object content. The value is not quoted. For manifest objects, this value is the MD5 checksum of the concatenated string of MD5 checksums and ETags for each of the segments in the manifest, and not the MD5 checksum of the content that was downloaded. Also the value is enclosed in double-quote characters. You are strongly recommended to compute the MD5 checksum of the response body as it is received and compare this value with the one in the ETag header. If they differ, the content was corrupted, so retry the operation.
`Content-Encoding`	String *(Optional)*	If set, the value of the `Content-Encoding` metadata. If not set, this header is not returned by this operation.
`Content-Disposition`	String *(Optional)*	If set, specifies the override behavior for the browser. For example, this header might specify that the browser use a download program to save this file rather than show the file, which is the default. If not set, this header is not returned by this operation.
`X-Delete-At`	String *(Optional)*	If set, the time when the object will be deleted by the system in the format of a UNIX Epoch timestamp. If not set, this header is not returned by this operation.
`X-Object-Manifest`	String *(Optional)*	If set, to this is a dynamic large object manifest object. The value is the container and object name prefix of the segment objects in the form `container/prefix`.
`X-Object-Meta-name`	String	The custom object metadata item, where `{name}` is the name of the metadata item.

Name	Type	Description
	(Required)	One `X-Object-Meta-{name}` response header appears for each metadata item (for each `{name}`).
`X-Static-Large-Object`	Bool *(Required)*	Set to `True` if this object is a static large object manifest object.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

Create or update object metadata

Method	URI	Description
POST	/v1/{account}/{container}/{object}	Creates or updates object metadata.

To create or update custom metadata, use the `X-Object-Meta-{name}` header, where `{name}` is the name of the metadata item.

In addition to the custom metadata, you can also update these system metadata items: `Content-Type` `Content-Encoiding` `Content-Disposition` `X-Delete-At`. However you cannot update other system metadata such as `Content-Length` or `Last-Modified`.

You can use **COPY** as an alternate to the **POST** operation by copying to the same object. With the **POST** operation you must specify all metadata items, whereas with the **COPY** operation, you need to specify only changed or additional items.

All metadata is preserved during the object copy. If you specify metadata on the request to copy the object, either **PUT** or **COPY**, the metadata overwrites any conflicting keys on the target (new) object.

A **POST** request deletes any existing custom metadata that you added with a previous **PUT** or **POST** request. Consequently, you must specify all custom metadata in the request. However, system metadata is unchanged by the **POST** request unless you explicitly supply it in a request header.

You can also set the `X-Delete-At` or `X-Delete-After` header to define when to expire the object.

When used as described in this section, the **POST** operation creates or replaces metadata. This form of the operation has no request body. The form **POST** feature can also use the **POST** operation to upload objects. For more information about form **POST** see *OpenStack Object Storage API v1 Reference* .

Example requests and responses:

- Create object metadata:

```
curl -i $publicURL/marktwain/goodbye -X POST -H "X-Auth-Token:
$token" -H "X-Object-Meta-Book: GoodbyeColumbus"
```

```
HTTP/1.1 202 Accepted
Content-Length: 76
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txb5fb5c91ba1f4f37bb648-0052d84b3f
Date: Thu, 16 Jan 2014 21:12:31 GMT

<html><h1>Accepted</h1><p>The request is accepted for processing.</p></html>
```

- Update object metadata:

```
curl -i $publicURL/marktwain/goodbye -X POST -H "X-Auth-Token:
$token" -H "X-Object-Meta-Book: GoodbyeOldFriend"
```

```
HTTP/1.1 202 Accepted
Content-Length: 76
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx5ec7ab81cdb34ced887c8-0052d84ca4
Date: Thu, 16 Jan 2014 21:18:28 GMT

<html><h1>Accepted</h1><p>The request is accepted for processing.</p></html>
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

Normal response codes: 202

Request

This table shows the header parameters for the create or update object metadata request:

Name	Type	Description
`X-Auth-Token`	String *(Optional)*	Authentication token. If you omit this header, your request fails unless the account owner has granted you access through an access control list (ACL).
`X-Object-Meta-name`	String *(Optional)*	The container metadata, where `{name}` is the name of the metadata item. You must specify a `X-Object-Meta-{name}` header for each metadata item (for each `{name}`) that you want to add or update.
`X-Delete-At`	Int *(Optional)*	The certain date, in the format of a UNIX Epoch timestamp, when the object is removed.
`Content-Disposition`	String *(Optional)*	If set, specifies the override behavior for the browser. For example, this header might specify that the browser use a download program to save this file rather than show the file, which is the default.
`Content-Encoding`	String *(Optional)*	If set, the value of the `Content-Encoding` metadata.
`X-Delete-After`	Int *(Optional)*	Specifies the number of seconds after which the object is removed. Internally, the Object Storage system stores this value in the `X-Delete-At` metadata item.
`Content-Type`	String *(Optional)*	Changes the MIME type for the object.
`X-Detect-Content-Type`	Boolean *(Optional)*	If set to `true`, Object Storage guesses the content type based on the file extension and ignores the value sent in the `Content-Type` header, if present.

This table shows the URI parameters for the create or update object metadata request:

Name	Type	Description
`{account}`	String	The unique name for the account. An account is also known as the project or tenant.
`{container}`	String	The unique name for the container.
`{object}`	String	The unique name for the object.

Response

This table shows the header parameters for the create or update object metadata response:

Name	Type	Description
`Content-Length`	String *(Required)*	If the operation succeeds, this value is zero (0). If the operation fails, this value is the length of the error text in the response body.
`Content-Type`	String *(Required)*	The MIME type of the object.
`X-Trans-Id`	Uuid *(Required)*	A unique transaction identifier for this request. Your service provider might need this value if you report a problem.
`Date`	Datetime *(Required)*	The transaction date and time.

3. Object Storage API examples

cURL commands 76
Authenticate 78
Account services 79
Container services 81
Object services 86

This section introduces the cURL command language and demonstrates how to use cURL commands to make Object Storage API calls.

Note

For more examples, see Object Storage API v1.

cURL commands

cURL is a command-line tool that you can use to interact with REST interfaces. cURL lets you to transmit and receive HTTP requests and responses from the command line or a shell script, which enables you to work with the API directly. It is available for Linux distributions, Mac OS X, and Windows. For information about cURL, see http://curl.haxx.se/.

To run the cURL request examples shown in this guide, copy each example from the HTML version of this guide directly to the command line or a script.

Before you can run these examples, you must set environment variables. See the section called "Environment variables required to run examples" [4].

This example cURL command shows account details and lists containers in the account.

```
# curl -i $publicURL?format=json \
         -X GET -H "X-Auth-Token: $token"
```

```
HTTP/1.1 200 OK
Content-Length: 96
X-Account-Object-Count: 1
X-Timestamp: 1389453423.35964
X-Account-Meta-Subject: Literature
X-Account-Bytes-Used: 14
X-Account-Container-Count: 2
Content-Type: application/json; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: tx274a77a8975c4a66aeb24-0052d95365
Date: Fri, 17 Jan 2014 15:59:33 GMT
```

The response, in JSON format, is:

```
[
   {
      "count":0,
      "bytes":0,
      "name":"janeausten"
   },
   {
      "count":1,
      "bytes":14,
      "name":"marktwain"
   }
]
```

Note

The carriage returns in the cURL request examples are escaped with a backslash (\) character. The escape character allows continuation of the command across multiple lines. However, do not include the escape character in the JSON or XML request body within the cURL command.

The cURL examples in this guide use the following command-line options:

Table 3.1. cURL command-line options

Option	Description
`-d`	Sends the specified data in a **POST** request to the HTTP server. Use this option to send a JSON or XML request body to the server.
`-H`	Specifies an extra HTTP header in the request. You can specify any number of extra headers. Precede each header with the `-H` option.
`-i`	Includes the HTTP response headers in the output.
`-s`	Silent or quiet mode. Does not show progress or error messages. Makes cURL mute.
`-T`	Transfers the specified local file to the remote URL.
`-X`	Specifies the request method to use when communicating with the HTTP server. The specified request is used instead of the default method, which is **GET**.

json.tool

For commands that return a response, you can append the following code to the command to call the json.tool to pretty-print output:

```
| python -m json.tool
```

To use the `json.tool`, import the `json` module. For information about the `json.tool`, see json — JSON encoder and decoder.

If you run a Python version older than 2.6, import the `simplejson` module and use the `simplejson.tool`. For information about the `simple.json` tool, see simplejson — JSON encoder and decoder.

If you do not want to pretty-print JSON output, omit this code.

Example of an XML response

To request an XML response, append the `format=xml` query parameter to the request.

This example cURL command shows account information and list containers in the account, and asks for the response in XML:

```
# curl -i $publicURL?format=xml \
         -X GET -H "X-Auth-Token: $token"
```

```
HTTP/1.1 200 OK
Content-Length: 262
X-Account-Object-Count: 1
X-Timestamp: 1389453423.35964
X-Account-Meta-Subject: Literature
X-Account-Bytes-Used: 14
X-Account-Container-Count: 2
Content-Type: application/xml; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: tx69f60bc9f7634a01988e6-0052d9544b
Date: Fri, 17 Jan 2014 16:03:23 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<account name="my_account">
    <container>
        <name>janeausten</name>
        <count>0</count>
        <bytes>0</bytes>
    </container>
    <container>
        <name>marktwain</name>
        <count>1</count>
        <bytes>14</bytes>
    </container>
</account>
```

Authenticate

The following examples show you how to authenticate with the Identity Service or Tempauth.

Authenticate with the Identity Service

This section provides an overview of the authentication process. For request and response details, see Authenticate in the *OpenStack Identity Service API v2.0 Reference*.

To authenticate with the Identity Service

1. Send your credentials and a tenant ID or tenant name to the Identity Service.

   The response includes an authentication token and service catalog.

2. Select the service catalog entry where `type` is `object-store`. Use the `publicURL` endpoint, which contains a URL with the full path to the Object Storage account. The URL has the format, `https://hostname/v1/account`.

Authenticate with Tempauth

To authenticate with Tempauth

1. Supply your user name and API access key in headers, as follows:

   - `X-Auth-User` header. Specify your Object Storage user name.
   - `X-Auth-Key` header. Specify your access key.

   The following example shows a sample request:

```
# curl -i https://storage.clouddrive.com/v1/auth \
-H "X-Auth-User: jdoe" -H "X-Auth-Key: jdoepassword"
```

2. When authentication succeeds, you receive a 204 No Content status code. Any 2*nn* response indicates success.

   The `X-Auth-Token` response header contains the authentication token. The `X-Storage-Url` response header contains a URL that includes a full path to the Object Storage account. The URL has the format, `https://hostname/v1/account`.

   The following example shows a sample response:

```
HTTP/1.1 204 No Content
Date: Mon, 12 Nov 2010 15:32:21
Server: Apache
X-Storage-Url: $publicURL
X-Auth-Token: $token
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

Account services

Show storage usage

To show how much data you have stored in the system and the number of containers that you are using, send a **HEAD** request to the Object Storage service.

Use the *-X* switch to specify the **HEAD** method.

Use the *-i* switch to send the HTTP response to terminal output.

Include the authentication token in the `X-Auth-Token` header.

```
# curl -i $publicURL -X HEAD -H "X-Auth-Token: $token"
```

```
HTTP/1.1 204 No Content
Content-Length: 0
X-Account-Object-Count: 1
X-Account-Meta-Book: MobyDick
```

```
X-Timestamp: 1389453423.35964
X-Account-Bytes-Used: 14
X-Account-Container-Count: 2
Content-Type: text/plain; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: txafb3504870144b8ca40f7-0052d955d4
Date: Fri, 17 Jan 2014 16:09:56 GMT
```

The `X-Account-Bytes-Used` response header shows the total bytes stored for the entire account.

The `X-Account-Container-Count` response header shows the number of containers in this storage account.

Show account details

This example shows account details, lists containers, and asks for a JSON response:

```
# curl -i $publicURL?format=json -X GET -H "X-Auth-Token: $token"
```

```
HTTP/1.1 200 OK
Content-Length: 96
X-Account-Object-Count: 1
X-Timestamp: 1389453423.35964
X-Account-Meta-Subject: Literature
X-Account-Bytes-Used: 14
X-Account-Container-Count: 2
Content-Type: application/json; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: tx274a77a8975c4a66aeb24-0052d95365
Date: Fri, 17 Jan 2014 15:59:33 GMT
```

```
[
   {
      "count":0,
      "bytes":0,
      "name":"janeausten"
   },
   {
      "count":1,
      "bytes":14,
      "name":"marktwain"
   }
]
```

This example shows account details, lists containers, and asks for an XML response:

```
# curl -i $publicURL?format=xml -X GET -H "X-Auth-Token: $token"
```

```
HTTP/1.1 200 OK
Content-Length: 262
X-Account-Object-Count: 1
X-Timestamp: 1389453423.35964
X-Account-Meta-Subject: Literature
X-Account-Bytes-Used: 14
X-Account-Container-Count: 2
Content-Type: application/xml; charset=utf-8
Accept-Ranges: bytes
X-Trans-Id: tx69f60bc9f7634a01988e6-0052d9544b
Date: Fri, 17 Jan 2014 16:03:23 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<account name="my_account">
    <container>
        <name>janeausten</name>
        <count>0</count>
        <bytes>0</bytes>
    </container>
    <container>
        <name>marktwain</name>
        <count>1</count>
        <bytes>14</bytes>
    </container>
</account>
```

Container services

Container ACLs

The `X-Container-Read` metadata header defines the access control list (ACL) permissions for who can read objects in a container. Before you set this header, only users with a valid authentication token for the account can read objects in that container.

List containers to show the absence of the `X-Container-Read` header:

```
# curl –X GET -i -H "X-Auth-Token: $token" $publicURL/jerry
```

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

Set the `X-Container-Read` header to enable read and list access to everyone:

```
# curl –X PUT -i \
-H "X-Auth-Token: $token" \
-H "X-Container-Read: .r:*,.rlistings" \
$publicURL/jerry
```

```
HTTP/1.1 202 Accepted
Content-Length: 58
Content-Type: text/html; charset=UTF-8
X-Trans-Id: txf2befb56b1854a50995f710f2db48089
Date: Tue, 15 Nov 2011 03:33:16 GMT

202 Accepted

The request is accepted for processing.
```

For a list of valid `X-Container-Read` header values, see ACLs.

To see the metadata change:

```
# curl –X GET -i -H "X-Auth-Token: $token" $publicURL/jerry
```

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

```
X-Container-Read: .r:*,.rlistings
X-Container-Bytes-Used: 0
Accept-Ranges: bytes
X-Trans-Id: txb40eb86d949345f7bc66b01e8b63c3a5
Content-Length: 0
Date: Tue, 15 Nov 2011 03:33:36 GMT
```

After you give everyone read access, anyone can access any object in the container from a browser. To do so, a user appends the object name to the `X-Storage-URL` header value used in the session. For example:

```
$publicURL/jerry/cereal.jpg
```

Create a container

To create a container, issue a **PUT** request. You do not need to check if a container already exists before you issue a **PUT** request. The operation creates a container or updates an existing container, as appropriate.

Example requests and responses:

- Create a container with no metadata:

```
# curl -i $publicURL/steven -X PUT -H "Content-Length: 0" -H "X-Auth-Token:
 $token"
```

```
HTTP/1.1 201 Created
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx7f6b7fa09bc2443a94df0-0052d58b56
Date: Tue, 14 Jan 2014 19:09:10 GMT
```

- Create a container with metadata:

```
# curl -i $publicURL/marktwain -X PUT -H "X-Auth-Token: $token" -H "X-
Container-Meta-Book: TomSawyer"
```

```
HTTP/1.1 201 Created
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx06021f10fc8642b2901e7-0052d58f37
Date: Tue, 14 Jan 2014 19:25:43 GMT
```

For a complete description of HTTP 1.1 header definitions, see Header Field Definitions.

Page through large lists of containers

Note

You can also use this technique to page through large lists of objects.

For more information about how to page through large lists of containers and objects, see the section called “Page through large lists of containers or objects” [19].

For a list of five container names, if you specify a *`limit`* of two, two items are returned. You can assume there are more names to list, so make another request with a *`marker`* of the last item returned.

For example, assume the following list of container names:

```
apples
bananas
kiwis
oranges
pears
```

To page through a large list of containers

1. Use a *limit* of two:

   ```
   GET $publicURL?limit=2
   Host: storage.swiftdrive.com
   X-Auth-Token: $token
   ```

   ```
   apples
   bananas
   ```

   Because two container names are returned, there are more names to list.

2. Make another request with a *marker* parameter set to the name of the last item returned:

   ```
   GET $publicURL?limit=2&marker=bananas
   Host: storage.swiftdrive.com
   X-Auth-Token: $token
   ```

   ```
   kiwis
   oranges
   ```

   Again, two items are returned, and there might be more.

3. Make another request with a *marker* of the last item returned:

   ```
   GET $publicURL?limit=2&marker=oranges
   Host: storage.swiftdrive.com
   X-Auth-Token: $token
   ```

   ```
   pears
   ```

   You now receive a one-item response, which is fewer than the *limit* number of names. This indicates that this is the end of the list.

4. Use the *end_marker* parameter to limit the result set to object names that are less than the *end_marker* parameter value:

   ```
   GET $publicURL/?end_marker=oranges
   Host: storage.swiftdrive.com
   X-Auth-Token: $token
   ```

   ```
   apples
   bananas
   kiwis
   ```

Get, copy, and delete objects

Now, retrieve an object that you previously uploaded. First, remove the local copy:

```
# ls -l
```

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

```
# rm JingleRocky.jpg
# ls -l
```

```
total 416
-rw-r--r--@ 1 petecj2  staff  100864 Nov  7 14:47 RockyAndBuster.jpg
-rw-r--r--@ 1 petecj2  staff  107103 Nov  7 14:47 SittingBuster.jpg
```

Be sure not to use -i switch here because you want the raw data, which you pipe to a file:

```
# curl –X GET -H "X-Auth-Token: $token" $publicURL/dogs/JingleRocky.jpg >
 JingleRocky.jpg
```

```
# ls -l
```

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

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

```
# curl –X PUT -i -H "X-Auth-Token: $token" -H "X-Copy-From: /dogs/JingleRocky.
jpg" -H "Content-Length: 0" $publicURL/elaine/JingleRocky.jpg
```

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

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

You can then confirm the new location of the object. Issue a **GET** request with the destination container:

```
# curl –X GET -i -H "X-Auth-Token: $token" $publicURL/elaine/
```

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

JingleRocky.jpg
```

To delete an object from its container:

```
# curl -X DELETE -i -H "X-Auth-Token: $token" $publicURL/elaine/JingleRocky.
jpg
```

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

List containers to confirm the deletion:

```
# curl -X GET -i -H "X-Auth-Token: $token" $publicURL/elaine/
```

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

Get container metadata and delete containers

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

```
# curl -X HEAD -i \
    -H "X-Auth-Token: $token" \
    $publicURL/dogs
HTTP/1.1 204 No Content
X-Container-Object-Count: 0
X-Container-Bytes-Used: 0
Accept-Ranges: bytes
X-Trans-Id: tx3dd984f9482341dd97546e9d49d65e90
Content-Length: 0
Date: Mon, 07 Nov 2011 20:39:41 GMT
```

To delete a container:

```
# curl -X DELETE -i \
    -H "X-Auth-Token: $token" \
    $publicURL/george
HTTP/1.1 204 No Content
Content-Length: 0
Content-Type: text/html; charset=UTF-8
X-Trans-Id: tx3fa3857f266f44319d9b8f4bf7ce7fc8
Date: Mon, 07 Nov 2011 20:42:58 GMT
```

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

```
# curl -X GET -i \
    -H "X-Auth-Token: $token" \
    $publicURL
HTTP/1.1 200 OK
X-Account-Object-Count: 0
X-Account-Bytes-Used: 0
X-Account-Container-Count: 4
Accept-Ranges: bytes
Content-Length: 24
Content-Type: text/plain; charset=utf-8
```

```
X-Trans-Id: tx2475741852b849ce9403e382fe3f8015
Date: Mon, 07 Nov 2011 20:43:08 GMT

cosmo
dogs
elaine
jerry
```

Object services

Create static large objects

To create a static large object:

1. Split the content into pieces.
2. Upload each piece into a segment object.
3. Create a manifest object.

This example places the segment objects into the `segments` container and the manifest object into the `images` container. Using a dedicated container for segment objects is convenient.

Assuming you have already split the image into three files, you can upload them. You have removed non-essential response headers so you can see the important details.

```
# curl –X PUT -i -H "X-Auth-Token: $token" -T ./piece1 $publicURL/segments/
terrier-jpg-one
```

```
HTTP/1.1 201 Created
Content-Length: 4000000
Etag: f7365c1419b4f349592c00bd0cfb9b9a
```

```
# curl –X PUT -i -H "X-Auth-Token: $token" -T ./piece2 $publicURL/segments/
terrier-jpg-two
```

```
HTTP/1.1 201 Created
Content-Length: 2000000
Etag: ad81e97b10e870613aecb5ced52adbaa
```

```
# curl –X PUT -i -H "X-Auth-Token: $token" -T ./piece3 $publicURL/segments/
terrier-jpg-three
```

```
HTTP/1.1 201 Created
Content-Length: 1000
Etag: 00b046c9d74c3e8f93b320c5e5fdc2c3
```

At this stage, you can create the manifest listing. Notice that the size and ETag are copied from the previous uploads. Create a file called `manifest.json` with the following content:

```
[
   {
      "path":"segments/terrier-jpg-one",
      "etag":"f7365c1419b4f349592c00bd0cfb9b9a",
      "size_bytes":4000000
```

```
   },
   {
      "path":"segments/terrier-jpg-two",
      "etag":"ad81e97b10e870613aecb5ced52adbaa",
      "size_bytes":2000000
   },
   {
      "path":"segments/terrier-jpg-three",
      "etag":"00b046c9d74c3e8f93b320c5e5fdc2c3",
      "size_bytes":1000
   }
]
```

The final operation is to upload this content into a manifest object. To indicate that this is a manifest object, you must specify the *multipart-manifest=put* query parameter.

```
# curl –X PUT -i -H "X-Auth-Token: $token" -T ./manifest.json  $publicURL/
images/terrier-jpg?multipart-manifest=put
```

Examine the static large object. Notice that its size is the total size of all the segments:

```
# curl –X HEAD -i -H "X-Auth-Token: $token" $publicURL/images/terrier-jpg
```

```
HTTP/1.1 200 OK
Content-Length: 6001000
Etag: "0c922c37f915efb1c9b97e6328b3e660"
```