Technote

Isilon

OneFS

Release number 7.2

Isilon Swift Technote

November, 2014

This section contains the following topics:

EMC²

Introduction

The Isilon Swift protocol enables you to access existing file-based data stored on your EMC Isilon cluster as objects using the industry-standard Swift application programming interface (API). The Swift API is implemented as a set of Representational State Transfer (REST) web services over HTTP. Additionally, content and metadata can be ingested as objects and concurrently accessed through other supported EMC Isilon protocols.

The Isilon Swift protocol service is disabled by default. Contact your EMC Isilon representative for obtaining a license key to access the Swift protocol and RESTful APIs for object storage operations.

Storing object data on a new or existing EMC Isilon cluster through Isilon Swift provides the following benefits:

- Consolidate storage for applications regardless of protocol
- Secure multi-tenancy for applications through access zones while uniformly protecting the data with enterprise storage capabilities such as authentication, access control, and identity management
- Manage data through enterprise storage features such as deduplication, replication, tiering, performance monitoring, snapshots, and NDMP backups
- Balance the work load across all of the nodes in a cluster through OneFS SmartConnect
- Store object data more efficiently with forward error correction instead of data replication
- Eliminate storage silos
- Automate the collection, storage, and management of petabytes of unstructured data in an Isilon data lake for later analysis
- Automate data-processing applications to store objects on an Isilon cluster and analyze the data with Hadoop through the OneFS HDFS interface

Note

Except for the CloudPools and Audit features, Isilon Swift is compatible with the other EMC Isilon features.

Supported libraries, HTTP requests, and authentication methods

Isilon Swift supports the following client libraries, HTTP requests, and authentication methods:

- Compatible with the following Swift client applications and APIs:
  - The Swift command-line client and the Python-Swift client library. See http://docs.openstack.org/developer/python-swiftclient for more information.
  - Apache Libcloud, which is a Python library that supports multiple APIs for object storage through a unified API. See http://libcloud.readthedocs.org. for more information on Apache Libcloud.
- Supports the GET, PUT, DELETE, POST, HEAD, and COPY requests to work with the above libraries.
- Supports the TempAuth authentication method. In addition, supports the GET Token request for the following authentication methods:

- Swauth
- Keystone
- Rackspace extension to Keystone

Unsupported features

Isilon Swift currently does not support some of the object storage features.

You cannot perform the following tasks:

- Perform HTTP transfer encodings and content-type guessing
- Automatically distribute copies of the objects across multiple clusters and dynamically failover
- Save versions of objects
- Specify a date to delete an object automatically using the `X-Delete-At` and `X-Delete-After` headers
- Perform a server-side copy through the `PUT X-Copy-From semantics` command
- Access built-in Conditional GET and PUT calls based on ETag matching
- Use the HTTPS protocol
- Access large object manifests
- Authenticate through the BasicAuth method
- Change the access control lists (ACLs) on containers
- Specify a location to store objects, containers, and accounts within a cluster of storage nodes
- Automatically synchronize the contents of a container from one storage node to another
- Use unlimited containers
- Access Geo-distribution
- Use cross-site HTTP
- Use rate-limiting
- Use S3 support
- Use Byte Range Access
- Use TempURL

Object data storage overview

Isilon Swift stores content and metadata as objects through an application programming interface (API), which is implemented as a set of Representational State Transfer (REST) web services over HTTP.

OneFS exposes the Swift API through a Swift protocol driver. An instance of this protocol driver runs on each node in the cluster and handles the API requests.

The Swift API presents the home directories as accounts, directories as containers, and files as objects. Each home directory in the OneFS file system maps to a Swift account. The directories and subdirectories in a home directory map to containers and subcontainers. Files appear as objects. Because each object has a URL, you can access a

file through its URL. All objects have metadata. The API requests that you submit can store and manage containers, objects, and metadata in the OneFS file system.

Store and manage objects through Isilon Swift

Isilon Swift enables you to store content and metadata as objects on an EMC Isilon cluster.

Before you begin

Obtain a license key to unlock the Isilon Swift protocol.

Procedure

1. Connect to a cluster through the HTTP protocol by providing your username and password.

   The default port setting for the connection is 28080. You cannot modify this default setting.

   Upon successful authentication, a token is generated and returned to you.
2. Using your token key, submit the GET, PUT, COPY, DELETE, HEAD, or POST requests over HTTP to perform an object storage operation.

Authentication and access to Isilon Swift

You must authenticate with an EMC Isilon cluster before accessing the Isilon Swift service for processing object storage requests.

Authentication for an Isilon Swift connection takes place in an access zone. A storage administrator can create one or more access zones to provide users from different identify management systems, access to various OneFS resources. A storage administrator can also set up user accounts and configure an EMC Isilon cluster to work with multiple identity management systems, such as Isilon Swift namespaces, NFS namespaces, SMB namespaces, and HDFS namespaces.

As an Isilon Swift user, you are assigned an access zone and access to accounts within an EMC Isilon cluster. You can submit an authentication request through a supported authentication method. OneFS generates an access token for you upon successful authentication. You can use this token for subsequent object storage operations.

Note

An Isilon Swift authentication fails in the following cases:

- You have configured one or more LDAP, Active Directory, NIS, or local providers for authentication but have not specified a home directory when setting up these providers.
- Your home directory is not mounted on `/ifs`.

Authentication methods

You can submit an authentication request to an EMC Isilon cluster through the following authentication methods:

- The standard Swift format used by the OpenStack TempAuth and Swauth modules
- The Rackspace extension to the OpenStack Identity Service, which Apache Libcloud uses as its primary authentication method
- The Get Token request supported by the Keystone authentication method

TempAuth and Swauth authentication methods

The TempAuth and the Swauth authentication methods use a similar request syntax. Provide the username and password for the authentication request through the HTTP headers. In the request syntax you may either specify an IP address or a DNS name.

Request syntax

```
curl -H "X-Auth-User: <any value works here>:<username>" -H "X-Auth-
Key: <password>" -v http://<ip-address>:28080/auth/v1.0 -X GET
```

Example request

```
curl -H "X-Auth-User: myaccount:myusername" -H "X-Auth-Key:
mypassword" -v http://137.69.154.252:28080/auth/v1.0 -X GET
```

Example response

```
< HTTP/1.1 200 OK
< Content-Length: 101
< Content-Type: application/json; charset=utf-8
< Date: Mon, 18 Aug 2014 11:55:06 PDT
< X-Auth-Token: AUTH_tk71e9c58f3e432da1adf3ba70dc236dd8
< X-Storage-Token: AUTH_tk71e9c58f3e432da1adf3ba70dc236dd8
< X-Storage-Url: http://137.69.154.252:28080/v1/AUTH_admin
< X-Trans-Id: tx94aa46b29449450cb0f7c-0053f24c0a
<
* Connection #0 to host 137.69.154.252 left intact
{"storage":
{
"cluster_name": "http://137.69.154.252:28080/v1/AUTH_myaccount",
"default": "cluster_name"
}
}
```

Keystone authentication method

You can initiate the Keystone authentication method by submitting a POST request on a URL.

Request syntax

```
curl -H "Content-Type: application/json; charset=utf-8" -v http://
<node-ip>:28080/v2.0/tokens -X POST -d '{"auth":{"tenantName":"<any-
value-works-here>", "passwordCredentials":{"username": "<username>",
"password":"<password>"}}}'
```

Example request

```
curl -H "Content-Type: application/json; charset=utf-8" -v http://
137.69.154.252:28080/v2.0/tokens -X POST -d '{"auth":
{"tenantName":"isilon", "passwordCredentials":{"username":
"myusername", "password":"mypassword"}}}'
```

Example response

```
< HTTP/1.1 200 OK
< Content-Length: 467
< Content-Type: text/html; charset=UTF-8
< Date: Mon, 18 Aug 2014 13:13:54 PDT
< X-Trans-Id: txdc10f3db5b1444a6a4521-0053f25e82
<
```

```
* Connection #0 to host 137.69.154.252 left intact
{ "access":
{ "token":
{
"expires":"1600-12-31 16:02:20",
"id":"AUTH_tk71e9c58f3e432da1adf3ba70dc236dd8",
"tenant":{ "id":"10", "name":"isilon" } },
"serviceCatalog":
[
{ "endpoints":
[
{
"region":"ISILON", "internalURL":"http://127.0.0.1/v1/AUTH_isilon",
"publicURL":"http://137.69.154.252:28080/v1/AUTH_isilon"
}
],
"type":"object-store", "name":"swift"
}
],
"user":
{
"id":"10",
"roles":[ { "tenantId":"10", "id":"0", "name":"object-
store:myusername" } ],
"name":"mypassword"
}
}
}
```

Rackspace Extension to Keystone authentication method

The Rackspace Extension to Keystone authentication method follows the Apache Libcloud authentication method. You do not need to provide an account name or a tenant name in the request syntax.

Request syntax

```
curl -H "Content-Type: application/json; charset=utf-8" -v http://
<node-ip>:28080/v2.0/tokens -X POST -d '{"auth":{"RAX-
KSKEY:apiKeyCredentials":{"username":”<username>”,
"apiKey":”<password>”}}}'
```

Example request

```
curl -H "Content-Type: application/json; charset=utf-8" -v http://
137.69.154.252:28080/v2.0/tokens -X POST -d '{"auth":{"RAX-
KSKEY:apiKeyCredentials":{"username":"myusername",
"apiKey":"mypassword"}}}'
```

Example response

```
< HTTP/1.1 200 OK
< Content-Length: 467
< Content-Type: text/html; charset=UTF-8
< Date: Mon, 18 Aug 2014 13:23:22 PDT
< X-Trans-Id: tx6412180ff9514fb88d3d3-0053f260ba
<
* Connection #0 to host 137.69.154.252 left intact
{
"access":
{
"token":
{
"expires":"1600-12-31 16:02:20",
```

```
"id":"AUTH_tk71e9c58f3e432da1adf3ba70dc236dd8",
"tenant":
{
"id":"10",
"name":"isilon"
}
},
"serviceCatalog":
[
{
"endpoints":
[
{
"region":"ISILON",
"internalURL":"http://127.0.0.1/v1/AUTH_isilon", "publicURL":"http://
137.69.154.252:28080/v1/AUTH_isilon"
}
],
"type":"object-store",
"name":"swift"
}
],
"user":
{
"id":"10",
"roles":
[
{
"tenantId":"10",
"id":"0",
"name":"object-store:myusername"
}
],
"name":"myusername"
}
}
}
```

Apache Libcloud authentication method

Apache Libcloud is a Python library that supports Isilon Swift and other object storage provider APIs through a unified API.

Sample code

```
from libcloud.storage.types import Provider
from libcloud.storage.providers import get_driver
import sys

ip = "137.69.154.250"

cls = get_driver(Provider.OPENSTACK_SWIFT)

my_username=”admin”
my_password=”a”
driver = cls(my_username, my_password, region='ISILON',
        ex_force_auth_url='http://'+ip+':28080',
        ex_force_base_url='http://'+ip+':28080/v1/AUTH_ISILON',
        ex_force_auth_version='2.0',
        ex_force_service_type='object-store',
        ex_force_service_name='swift')
```

Authentication token overview

When you authenticate successfully with an EMC Isilon cluster, OneFS generates and returns a unique authentication token. OneFS checks your access to directories and files later in that session through this token.

The authentication token is a string of 32 hex characters and is unique for each user. The 32-character string is prefixed with `AUTH_tk`. The token also contains a URL that you should use for subsequent object storage requests. You can use a token that is generated on one node in a cluster on any other node in the cluster.

A token is time-sensitive. The default expiry period is one day. If a token expires, you can obtain a new token by initiating a new authentication request.

Managing object storage through Isilon Swift

The Swift API is an HTTP-based protocol used by Swift client applications to communicate with a Swift proxy server to execute object storage requests.

The Swift API supports the GET, PUT, DELETE, POST, HEAD, and COPY request types described in the following table:

Request type	Description
GET	Retrieves an object, the contents of a container, or an account.
PUT	Uploads an object, or creates a container. **Note** You cannot create an account through the current version of Swift.
DELETE	Deletes an object or a container. **Note** You cannot delete an account through the current version of Swift.
POST	Stores metadata on an object, container, or an account.
HEAD	Retrieves metadata on an object, container, or an account.
COPY	Performs a server-side copy of an object.

GET account

Retrieves an account.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>" -X GET
```

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
"http://137.69.154.240:28080/v1/AUTH_admin?format=json" -X GET
```

Example response

```
< HTTP/1.1 200 OK
< Content-Length: 84
< Content-Type: application/json; charset=utf-8
< Date: Thu, 19 Jun 2014 14:54:32 PDT
< Last-Modified: 2014-06-19 14:54:29
< X-Account-Bytes-Used: 13
< X-Account-Container-Count: 2
< X-Account-Object-Count: 2
< X-Timestamp: 130476887138427115
< X-Trans-Id: tx83b138929e134d24a4cd7-0053a35d1b
<
[
    {
        "count": 1,
        "bytes": 4,
        "name": "container"
    },
    {
        "count": 1,
        "bytes": 9,
        "name": "container2"
    }
]
```

GET container

Retrieves the contents of a container.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>" -X GET
```

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container2?
format=json" -X GET
```

Example response

```
< HTTP/1.1 200 OK
< Accept-Ranges: bytes
< Content-Length: 217
< Content-Type: application/json; charset=utf-8
< Date: Thu, 19 Jun 2014 14:59:44 PDT
< Last-Modified: 2014-06-19 14:58:48
< X-Container-Bytes-Used: 9
< X-Container-Object-Count: 1
< X-Timestamp: 130476887284112371
< X-Trans-Id: tx169d4d5da58647dea7ef0-0053a35d50
<
[
```

```
    {
        "hash": "ef614d88b44d9b768ca06befb206cf3c",
        "last_modified": "2014-06-19 14:58:48",
        "bytes": "9",
        "name": "obj2.txt",
        "content_type": "application/octet-stream"
    }
]
```

GET object

Retrieves an object.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>/<object>" -X
GET
```

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container2/obj2.txt" -
X GET
```

Example response

```
< HTTP/1.1 200 OK
< Content-Length: 9
< Content-Type: application/octet-stream
< Date: Thu, 19 Jun 2014 15:01:13 PDT
< ETag: ef614d88b44d9b768ca06befb206cf3c
< Last-Modified: 2014-06-19 14:58:48
< X-Timestamp: 130476887284112371
< X-Trans-Id: tx2572012006224369ae08f-0053a35da9
```

PUT container

Creates a container.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>" -X PUT
```

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container3" -X PUT
```

Example response

```
< HTTP/1.1 201 Created
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 10:54:50 PDT
< X-Trans-Id: tx0ceffc19132b45d1ba91d-0053a4756a
```

PUT object

Uploads an object.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>/<object>"
-X PUT -d "<object_content>"
```

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container2/obj2.txt"
-X PUT -d "object content"
```

Example response

```
< HTTP/1.1 201 Created
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 10:47:52 PDT
< ETag: e3cf5b79108fc1cc822bd0f9b5b67cef
< X-Trans-Id: tx5e366ec5f04041f9b8500-0053a473c8
```

POST account

Stores metadata on an account.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>"
-X POST -H "<metadata_tag1>: <value1>" "<metadata_tag2>: <value2>"
```

Note

When you submit a POST request on an account, make sure that the metadata name is prefixed with the *X-Account-Meta-* tag.

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin" -X POST
```

```
-H "X-Account-Meta-Owner: Name" -H "X-Account-Meta-DOC-Types:
Technical"
```

Example response

```
< HTTP/1.1 204 No Content
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 11:07:42 PDT
< X-Trans-Id: txa819a723b6f449d593966-0053a4786e
```

POST container

Stores metadata on a container.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>"
-X POST -H "<container_metadata_tag1>: <value1>"
"<container_metadata_tag2>: <value2>"
```

Note

When you submit a POST request on a container, make sure that the metadata name is prefixed with the *X-Container-Meta-* tag.

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container3"
-X POST -H "X-Container-Meta-Owner: Name"
    -H "X-Container-Meta-Doc-Types: Technical"
```

Example response

```
< HTTP/1.1 204 No Content
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 11:09:56 PDT
< X-Trans-Id: tx1250a35c0ac14f2480612-0053a478f4
```

POST object

Stores metadata on an object.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>/<object>"
-X POST -H "<object_metadata_tag1>: <value1>"
"<object_metadata_tag2>: <value2>"
```

Note

When you submit a POST request on an object, make sure that the metadata name is prefixed with the *X-Object-Meta* tag.

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container2/obj2.txt"
-X POST -H "X-Object-Meta-Owner: Name"
-H "X-Object-Meta-Doc-Types: Technical"
```

Example response

```
< HTTP/1.1 204 No Content
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 11:12:36 PDT
< X-Trans-Id: tx7205bce69fe547508c176-0053a47994
```

COPY object

Performs a server-side copy of an object.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>/<object>"
-X COPY -H "Destination: <container>/<sub-container>/<object>"
```

Example request

This example shows a COPY object request submitted to perform a server-side copy of `container2/obj2` to a new object `obj2_copy.txt` into a sub-container:

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container2/obj2.txt"
-X COPY -H "Destination: container/sub-container/obj2_copy.txt"
```

Example response

```
< HTTP/1.1 201 Created
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 11:38:18 PDT
< X-Trans-Id: tx5c765fa5e1e64f5bb6864-0053a47f9a
```

HEAD account

Retrieves metadata from an account.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>" -X HEAD
```

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin" -X HEAD
```

Example response

```
< HTTP/1.1 204 No Content
< X-Account-Meta-Name1: Value1
< X-Account-Meta-Name2: Value2
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 11:16:23 PDT
< Last-Modified: 2014-06-20 11:07:42
< X-Account-Bytes-Used: 15
< X-Account-Container-Count: 3
< X-Account-Object-Count: 2
< X-Timestamp: 130477604900823291
< X-Trans-Id: txe0e5f861dca641268ce4e-0053a47a76
```

HEAD container

Retrieves metadata from a container.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>" -X HEAD
```

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container2" -X HEAD
```

Example response

```
< HTTP/1.1 204 No Content
< X-Container-Meta-Name1: Value1
< X-Container-Meta-Name2: Value2
< Accept-Ranges: bytes
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 11:14:40 PDT
< Last-Modified: 2014-06-20 11:09:56
< X-Container-Bytes-Used: 0
< X-Container-Object-Count: 0
< X-Timestamp: 130477613968019069
< X-Trans-Id: tx56d23577d9e04dad95d73-0053a47a10
```

HEAD object

Retrieves metadata from an object.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>/<object>" -X
HEAD
```

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container2/obj2.txt" -
X HEAD
```

Example response

```
< HTTP/1.1 204 No Content
< X-Object-Meta-Name1: Value1
< X-Object-Meta-Name2: Value2
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 11:13:57 PDT
< ETag: e3cf5b79108fc1cc822bd0f9b5b67cef
< Last-Modified: 2014-06-20 11:12:36
< X-Timestamp: 130477615561086205
< X-Trans-Id: tx4951a255ec3e43e5b6c09-0053a479e5
```

DELETE container

Deletes a container.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>" -X DELETE
```

Example request

```
curl -H "X-Auth-Token: AUTH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container3" -X DELETE
```

Example response

```
< HTTP/1.1 204 No Content
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 11:34:26 PDT
< X-Trans-Id: tx04f0072a9573448ea0e52-0053a47eb2
```

DELETE object

Deletes an object.

The syntax described below uses curl, the python-swiftclient API. See http://docs.openstack.org/api/openstack-object-storage/1.0/content/ for more information on the request types and parameters used in the syntax and examples.

Request syntax

```
curl -H "X-Auth-Token: <token>" -v
    "http://<ip_address>:28080/v1/<account>/<container>/<object>" -X
DELETE
```

Example request

```
TH_tk9b2f1d4d640b31fee0b3f6d644aab52e" -v
    "http://137.69.154.240:28080/v1/AUTH_admin/container2/obj2.txt" -
X DELETE
```

Example response

```
< HTTP/1.1 204 No Content
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Fri, 20 Jun 2014 11:33:35 PDT
< X-Trans-Id: txbfe6a6b594904ad1a55fd-0053a47e7f
```

Paging through large lists of objects or containers

When you submit a GET request on an account with a large number of containers or objects, the Swift API returns the first 10,000 items.

If there are more than 10,000 items in an account or a container, you can use the `limit` and `marker URL` parameters to page through the items.

For example, if there are five objects `obj1`, `obj2`, `obj3`, `obj4`, and `obj5` in a container named `small_container`, type the following command to page through two of these objects at a time:

```
curl -H "X-Auth-Token: <token>" "http://<ip>:28080/v1/<account>/
small_container?limit=2" -X GET
```

This command returns `obj1` and `obj2`.

Get the next two objects by setting the `marker URL` parameter to the name of the object that was last returned as follows:

```
curl -H "X-Auth-Token: <token>" "http://<ip>:28080/v1/<account>/
small_container?limit=2&marker=obj2" -X GET
```

This command returns `obj3` and `obj4`.

Get the last item `obj5` by typing the following command:

```
curl -H "X-Auth-Token: <token>" "http://<ip>:28080/v1/<account>/
small_container?limit=2&marker=obj4" -X GET
```

By viewing the header parameter in the GET or HEAD requests, you can determine whether there are more items in the container or account than what is actually returned. The `X-Container-Object-Count` parameter displays the number of objects and the `X-Account-Container-Count` parameter displays the number of containers.

Troubleshooting and known issues

If you encounter problems when running Isilon Swift, take one of the recommended actions listed below and make note of the limitations for the specific request.

- You cannot submit a PUT request to create a zero-length object because PUT is incorrectly interpreted as a sub-container PUT.
- You cannot submit a DELETE request to delete a container if the container has empty subdirectories. As a workaround, delete such containers through the normal sub-container DELETE command executed through curl or the Swift command-line utility. If you are authenticating through the Apache Libcloud authentication method, delete the empty subdirectories through NFS, SMB, or a related protocol.
- When you submit a GET object request from your account (home) directory, the results returned are inaccurate. As a workaround, move all your files into subdirectories within your home directory and resubmit the request.
- If your account has a home directory specified but not created at the time of authenticating with Isilon Swift, you may successfully authenticate with Isilon Swift. However, you will receive a `404 Not Found` error message when you execute a request on the files in your home directory. To avoid this, make sure that your home directory is already created on `/ifs` before authenticating with Isilon Swift.
- The authentication system locks out users who change their home directories. User tokens that allow authentication are associated with the home directory for the user. Changing the home directory breaks this association. As a workaround, manually delete the token file for the user, which is stored at `/ifs/.ifsvar/modules/swift/user-tokens-<node-name>/<zone>@<username>` for each node, and restart the Swift service on all nodes. Alternatively, restore the original home directory for the user.
- If you are authenticating with Active Directory, make sure that the `assume-default-domain` setting is enabled for the domain. Otherwise, you must specify the domain along with the username in the Swift authentication request. For example, if your username is `test-name` and the domain is `mydomain.com`, then you must provide `test-name@mydomain.com` in the authentication request.

Published November, 2014

For the most up-to-date regulatory document for your product line, go to EMC Online Support (https://support.emc.com).