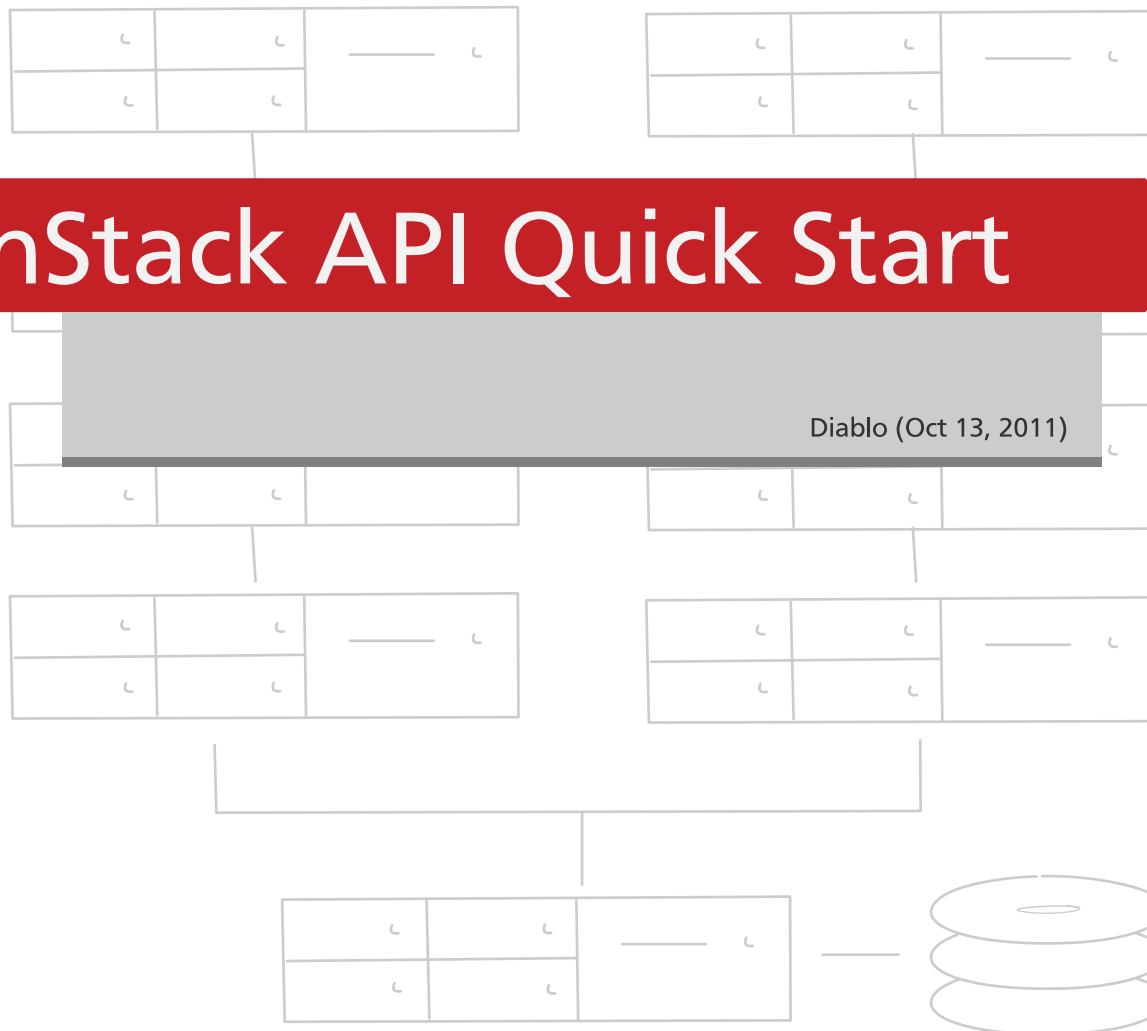


OpenStack API Quick Start



OpenStack API Quick Start

Diablo (2011-10-13)

Copyright © 2010, 2011 OpenStack LLC Some rights reserved.

OpenStack™ projects offer RESTful APIs for manipulating and retrieving objects in the cloud. This manual provides examples for sending REST requests using the Compute API v1.1, Identity API v2.0, and the Image API v1.1.



Except where otherwise noted, this document is licensed under
Creative Commons Attribution ShareAlike 3.0 License.
<http://creativecommons.org/licenses/by-sa/3.0/legalcode>

Table of Contents

1. OpenStack API Quick Start	1
OpenStack API Introduction	1
Getting Credentials	1
Sending Requests to the API	3
2. Setting Up python-novaclient	4
Listing Images	4
Launching Instances	6
3. OpenStack API Examples	8
Compute API Flavor Examples	8
Compute API Server Information Examples	9
Compute API Server Actions Examples	10
Compute API Extensions Examples	10
Image API Examples	11
Identity API Examples	12
Object API Examples	12
Storage Account Services	12
Storage Container Services	13
Storage Object Services	13

1. OpenStack API Quick Start

The OpenStack system has several key projects that are separate installations but can work together depending on your cloud needs: OpenStack Compute, OpenStack Object Storage, OpenStack Identity Service, and OpenStack Image Store. With a standalone OpenStack installation, the OpenStack Compute, OpenStack Identity, and OpenStack Image Store projects are all working together in the background of your installation.

OpenStack API Introduction

This page covers the basics for talking to your OpenStack cloud through the Compute API after authorizing with the Identity Service API. You can then build a cloud by launching images and assigning metadata to instances, all through the API. For an API reference of all the possible commands, see the OpenStack Compute API 1.1 specification and the Identity Service 2.0 Developer Guide published at docs.openstack.org.

Getting Credentials

Credentials are a combination of your username, password, and what tenant (or project) your cloud is running under. You only need to generate an additional token if you are interacting with your cloud directly with API endpoints, and not with a client. Your cloud administrator can give you a username and a password as well as your tenant identifier so you can generate authorization tokens. These tokens are typically good for 24 hours, and when the token expires, you will find out with a 401 (Unauthorized) error and can request another token programmatically. The general workflow goes something like this:

1. Begin API requests by asking for an authorization token from the endpoint your cloud administrator gave you, typically `http://hostname:5000/v2.0/tokens`. You send your username, password, and what group you are with (the "tenant" in auth-speak).

```
$ curl -X 'POST' -v http://host.na.me:5000/v2.0/tokens -d '{"auth":
{"passwordCredentials":{"username": "joecool", "password":"coolword"},
"tenantId":"5"}}' -H 'Content-type: application/json'
```

2. The server returns a response in which the 24-hours token is contained :

```
* About to connect() to host.na.me port 5000 (#0)
* Trying 55.51.11.198... connected
* Connected to host.na.me (55.51.11.198) port 5000 (#0)
> POST /v2.0/tokens HTTP/1.1
> User-Agent: curl/7.19.7 (universal-apple-darwin10.0) libcurl/7.19.7
  OpenSSL/0.9.8r zlib/1.2.3
> Host: host.na.me:5000v> Accept: */*
> Content-type: application/json
> Content-Length: 95
>
> HTTP/1.1 200 OK
> Content-Type: application/json; charset=UTF-8
> Content-Length: 935
> Date: Thu, 06 Oct 2011 19:59:49 GMT
```

```
>
* Connection #0 to host host.name left intact
* Closing connection #0
{"access": {"token": {"expires": "2011-10-07T14:59:49.644963",
"id": "e83abbd8c-a8c8-4f81-897b-541cbcd1dbb5",
"tenant": {"id": "5", "name": "coolu"}}, "serviceCatalog":
[{"endpoints": [{"adminURL": "http://55.51.11.198:8774/v1.1/5",
"region": "RegionOne", "internalURL": "http://55.51.11.198:8774/v1.1/5",
"publicURL": "http://55.51.11.198:8774/v1.1/5"}], "type": "compute", "name":
"nova"},
{"endpoints": [{"adminURL": "http://55.51.11.198:9292/v1.1/5",
"region": "RegionOne", "internalURL": "http://55.51.11.198:9292/v1.1/5",
"publicURL": "http://55.51.11.198:9292/v1.1/5"}], "type": "image", "name":
"glance"},
{"endpoints": [{"adminURL": "http://55.51.11.198:35357/v2.0",
"region": "RegionOne", "internalURL": "http://55.51.11.198:5000/v2.0",
"publicURL": "http://55.51.11.198:5000/v2.0"}],
"type": "identity", "name": "keystone"}], "user": {"id": "4", "roles":
[{"tenantId": "5", "id": "2", "name": "Member"}], "name": "joecool"}}
```

Use that token to send API requests with the X-Auth-Token included as an header field.

3. Repeatedly send API requests with that token in the x-auth-token header until either: 1) the job's done or 2) you get a 401 (Unauthorized) code in return.
4. Request a token again when you get a 401 response until the script's job is done.

For a typical OpenStack deployment running the Identity Service you can request a token with this command in cURL:

```
$ curl -X 'POST' -v http://host.name:5000/v2.0/tokens -d '{"auth":
{"passwordCredentials":{"username": "joecool", "password": "coolword"},
"tenantId": "5"}}' -H 'Content-type: application/json'
```

In return, you should get a 200 OK response with a token in the form of "id": "cd427a33-bb4a-4079-a6d7-0ae148bdeda9" and an expiration date 24 hours from now. Here's what it looks like:

```
* About to connect() to host.name port 5000 (#0)
* Trying hostname... connected
* Connected to host.name (hostname) port 5000 (#0)
> POST /v2.0/tokens HTTP/1.1
> User-Agent: curl/7.19.7 (universal-apple-darwin10.0) libcurl/7.19.7 OpenSSL/
0.9.8r zlib/1.2.3
> Host: host.name:5000
> Accept: */*
> Content-type: application/json
> Content-Length: 85
>
> HTTP/1.1 200 OK
> Content-Type: application/json; charset=UTF-8
> Content-Length: 1213
> Date: Thu, 01 Sep 2011 19:27:30 GMT
>
```

```
* Connection #0 to host host.name left intact
* Closing connection #0
{"access": {"token": {"expires": "2011-10-07T14:59:49.644963",
"id": "e83abbd8c-a8c8-4f81-897b-541cbcd1dbb5",
"tenant": {"id": "5", "name": "coolu"}}},
"serviceCatalog":
[{"endpoints": [{"adminURL": "http://55.51.11.198:8774/v1.1/5",
"region": "RegionOne", "internalURL": "http://55.51.11.198:8774/v1.1/5",
"publicURL": "http://55.51.11.198:8774/v1.1/5"}],
"type": "compute", "name": "nova"},
{"endpoints": [{"adminURL": "http://55.51.11.198:9292/v1.1/5",
"region": "RegionOne", "internalURL": "http://55.51.11.198:9292/v1.1/5",
"publicURL": "http://55.51.11.198:9292/v1.1/5"}], "type": "image", "name":
"glance"},
{"endpoints": [{"adminURL": "http://55.51.11.198:35357/v2.0",
"region": "RegionOne", "internalURL": "http://55.51.11.198:5000/v2.0",
"publicURL": "http://55.51.11.198:5000/v2.0"}],
"type": "identity", "name": "keystone"}],
"user": {"id": "4", "roles": [{"tenantId": "5", "id": "2", "name": "Member"}],
"name": "joecool"}}
```

Sending Requests to the API

You have a couple of options for sending requests to OpenStack through an API. Developers and testers may prefer to use cURL, the command-line tool from <http://curl.haxx.se/>. With cURL you can send HTTP requests and receive responses back from the command line.

If you like to use a more graphical interface, the REST client for Firefox also works well for testing and trying out commands, see <https://addons.mozilla.org/en-US/firefox/addon/restclient/>. You can also download and install rest-client, a Java application to test RESTful web services, from <http://code.google.com/p/rest-client/>.

You need to generate a token as shown above if you use cURL or a REST client.

2. Setting Up python-novaclient

For more serious scripting work, you can use a client like the python-novaclient or openstack-compute clients. The python-novaclient implements the Compute 1.1 API while the openstack-compute client works against the Rackspace Cloud Servers public cloud which is the OpenStack Compute 1.0 API. You only need a username and password to use the python-novaclient tool.

Installing the python-novaclient gives you a `nova` shell command that enables Compute API interactions from the command line. You install the client, and then provide your username and password, set as environment variables for convenience, and then you can have the ability to send commands to your cloud on the command-line.

To install python-novaclient, download the tarball from <http://pypi.python.org/pypi/python-novaclient/2.6.3#downloads> and then install it in your favorite python environment.

```
$ curl -O http://pypi.python.org/packages/source/p/python-novaclient/python-novaclient-2.6.3.tar.gz
$ tar -zxvf python-novaclient-2.6.3.tar.gz
$ cd python-novaclient-2.6.3
$ sudo python setup.py install
```

Now that you have installed the python-novaclient, confirm the installation by entering:

```
$ nova help

usage: nova [--username USERNAME] [--apikey APIKEY] [--projectid PROJECTID]
           [--url URL] [--version VERSION]
           <subcommand> ...
```

In return, you will get a listing of all the commands and parameters for the nova command line client. By setting up the required parameters as environment variables, you can fly through these commands on the command line. You can add `--username` on the nova command, or set them as environment variables:

```
export NOVA_USERNAME=joecool
export NOVA_API_KEY=coolword
export NOVA_PROJECT_ID=coolu
```

Using the Identity Service, you are supplied with an authentication endpoint, which nova recognizes as the NOVA_URL.

```
export NOVA_URL=http://hostname:5000/v2.0
export NOVA_VERSION=1.1
```

Listing Images

Before you can go about the business of building your cloud, you want to know what images are available to you by asking the image service what kinds of configurations are

available. The image service could be compared to iTunes for your cloud - you can view the playlist of images before using your favorite image to create a new instance in the cloud. To get the list of images, their names, status, and ID, use this command:

```
$ nova image-list
```

ID	Name	Status
1	aki-tty	ACTIVE
2	ari-tty	ACTIVE
3	ami-tty	ACTIVE
6	CentOS_5.4_x64	ACTIVE
14	maverick-kernel	ACTIVE
15	maverick	ACTIVE
20	ubuntu-kernel	ACTIVE
21	ubuntu-ramdisk	ACTIVE
22	ubuntu	ACTIVE
24	CentOS_5.6_x64_v5.7.14_Dev1	ACTIVE

Next you need to know the relative sizes of each of these.

```
$ nova flavor-list
```

ID	Name	Memory_MB	Swap	Local_GB	VCPUs	RXTX_Quota	RXTX_Cap
1	m1.tiny	512		0			
2	m1.small	2048		20			
3	m1.medium	4096		40			
4	m1.large	8192		80			
5	m1.xlarge	16384		160			
89	Boom	2		20			

You can also narrow down the list by using `grep` to find only the CentOS images with a command like this:

```
$ nova image-list | grep 'CentOS'
```

6	CentOS_5.4_x64	ACTIVE
24	CentOS_5.6_x64_v5.7.14_Dev1	ACTIVE

With the information about what is available to you, you can choose the combination of image and flavor to create your virtual servers and launch instances.

Launching Instances

To launch a server, you choose an image you want to match up to a size, find the ID for the image and the ID for the flavor so you can size it, and create the command with the IDs. From the information we got previously, we know that an Ubuntu Maverick image has an ID of 15, and if you want to start small with about 2 GB of memory and 20 GB of disk space, you'd choose the m1.small flavor which has an ID of 2. Put those parameters in with the "boot" command and you can create a new virtual server.

```
$ nova boot --flavor=2 --image=15 testserver
```

Property	Value
adminPass	*****
created	2011-09-01T21:40:41Z
flavor	m1.small
hostId	
id	1805
image	maverick
metadata	{}
name	testserver
progress	0
status	BUILD
updated	2011-09-01T21:40:41Z
uuid	ce044452-f22e-4ea4-a3ec-d1cde80cf996

Now, you can view this server in your new cloud by using the nova list command:

```
$ nova list
```

ID	Name	Status	Networks
1805	testserver	ACTIVE	private=10.4.96.81

There are three statuses you may see - ACTIVE, BUILDING, and UNKNOWN. The BUILDING status is transient and you likely will not see it. If you see UNKNOWN, run `nova list` again until it goes away.

To view all the information about a particular server, use `nova show` with the ID of the server that you got from the `nova list` command.

```
$ nova show 1805
```

Property	Value
created	2011-09-01T21:40:41Z
flavor	m1.small
hostId	58a7430169aa42cde5ce2456b0cb5bb5ac1ab0703bab6420e8a49e6e
id	1805
image	maverick
metadata	{}
name	testserver
private network	10.4.96.81
progress	100
status	ACTIVE
updated	2011-09-01T21:40:46Z
uuid	ce044452-f22e-4ea4-a3ec-d1cde80cf996

You can now launch that image again, but add more information to the server when you boot it so that you can more easily identify it amongst your ever-growing elastic cloud. Use the `-meta` option with a key=value pair, where you can make up the string for both the key and the value. For example, you could add a description and also the creator of the server.

```
$ nova boot testserver --meta description='Use for testing purposes' --meta creator=joecool
```

3. OpenStack API Examples

Here are some examples of the basic API commands you can send to an OpenStack deployment containing Compute, the Image Service, with a running Identity Server. In the "service catalog" that is returned when you get a token, you can find the endpoint URLs that you use for these API commands. For example this public URL entry in the service catalog offers the URL for the Image Service, code-named glance:

```
"publicURL": "http://55.51.11.198:9292/v1.1/9"}], "type": "image", "name":  
"glance"
```

For the Compute API, you see a service of type compute, code-named "nova" such as:

```
"publicURL": "http://55.51.11.198:8774/v1.1/9"}], "type": "compute", "name":  
"nova"
```

With these publicURLs in hand, you can now use curl commands for these endpoints and send API requests. The following sections offer walk-throughs of common API commands. Refer to docs.openstack.org/api for a full reference of all API commands for these services.

Compute API Flavor Examples

GET /v1.1/flavors

Gives you a listing of flavors available, which are combinations of disk space, memory and CPU capacity bundled together as resources for virtual machines.

Curl example:

```
curl -D - -H "X-Auth-Token: 3017813d-deca-4764-aa33-9ab59a283ba6" -X 'GET' -v  
http://demo.rcb.me:8774/v1.1/9/flavors  
  
{  
  "flavors": [  
    {  
      "id": 3, "links": [  
        {  
          "href": "http://demo.rcb.me:8774/v1.1/9/flavors/3", "rel": "self"},  
        {  
          "href": "http://demo.rcb.me:8774/9/flavors/3", "rel": "bookmark"}],  
      "name": "m1.medium"},  
    {  
      "id": 4, "links": [  
        {  
          "href": "http://demo.rcb.me:8774/v1.1/9/flavors/4",  
            "rel": "self"},  
        {  
          "href": "http://demo.rcb.me:8774/9/flavors/4", "rel": "bookmark"}],  
      "name": "m1.large"},  
    {  
      "id": 1, "links": [  
        {  
          "href": "http://demo.rcb.me:8774/v1.1/9/flavors/1",  
            "rel": "self"},  
        {  
          "href": "http://demo.rcb.me:8774/9/flavors/1", "rel": "bookmark"}],  
      "name": "m1.tiny"},  
    {  
      "id": 5, "links": [  
        {  
          "href": "http://demo.rcb.me:8774/v1.1/9/flavors/5",  
            "rel": "self"},  
        {  
          "href": "http://demo.rcb.me:8774/9/flavors/5", "rel": "bookmark"}],  
      "name": "m1.xlarge"},  
    {  
      "id": 2, "links": [  
        {  
          "href": "http://demo.rcb.me:8774/v1.1/9/flavors/2",  
            "rel": "self"},  
        {  
          "href": "http://demo.rcb.me:8774/9/flavors/2", "rel": "bookmark"}],  
      "name": "m1.small"}]  
    }
```

GET /v1.1/flavors/<id>

Gives you more detail about the specific flavor.

Curl example:

```
curl -D - -H "X-Auth-Token: 3017813d-deca-4764-aa33-9ab59a283ba6" -X 'GET' -v http://demo.rcb.me:8774/v1.1/9/flavors/5
```

```
{ "flavor": { "rxtx_quota": 0, "name": "m1.xlarge", "links": [ { "href": "http://demo.rcb.me:8774/v1.1/9/flavors/5", "rel": "self" }, { "href": "http://demo.rcb.me:8774/9/flavors/5", "rel": "bookmark" } ], "ram": 16384, "vcpus": 8, "rxtx_cap": 0, "swap": 0, "disk": 160, "id": 5 }
```

GET /v1.1/flavors/detail

Gives you details on all the available instance types (flavors).

Curl example:

```
curl -D - -H "X-Auth-Token: 3017813d-deca-4764-aa33-9ab59a283ba6" -X 'GET' -v http://demo.rcb.me:8774/v1.1/9/flavors/detail
```

```
{ "flavors": [ { "rxtx_quota": 0, "name": "m1.medium", "links": [ { "href": "http://demo.rcb.me:8774/v1.1/9/flavors/3", "rel": "self" }, { "href": "http://demo.rcb.me:8774/9/flavors/3", "rel": "bookmark" } ], "ram": 4096, "vcpus": 2, "rxtx_cap": 0, "swap": 0, "disk": 40, "id": 3 }, { "rxtx_quota": 0, "name": "m1.large", "links": [ { "href": "http://demo.rcb.me:8774/v1.1/9/flavors/4", "rel": "self" }, { "href": "http://demo.rcb.me:8774/9/flavors/4", "rel": "bookmark" } ], "ram": 8192, "vcpus": 4, "rxtx_cap": 0, "swap": 0, "disk": 80, "id": 4 }, { "rxtx_quota": 0, "name": "m1.tiny", "links": [ { "href": "http://demo.rcb.me:8774/v1.1/9/flavors/1", "rel": "self" }, { "href": "http://demo.rcb.me:8774/9/flavors/1", "rel": "bookmark" } ], "ram": 512, "vcpus": 1, "rxtx_cap": 0, "swap": 0, "disk": 0, "id": 1 }, { "rxtx_quota": 0, "name": "m1.xlarge", "links": [ { "href": "http://demo.rcb.me:8774/v1.1/9/flavors/5", "rel": "self" }, { "href": "http://demo.rcb.me:8774/9/flavors/5", "rel": "bookmark" } ], "ram": 16384, "vcpus": 8, "rxtx_cap": 0, "swap": 0, "disk": 160, "id": 5 }, { "rxtx_quota": 0, "name": "m1.small", "links": [ { "href": "http://demo.rcb.me:8774/v1.1/9/flavors/2", "rel": "self" }, { "href": "http://demo.rcb.me:8774/9/flavors/2", "rel": "bookmark" } ], "ram": 2048, "vcpus": 1, "rxtx_cap": 0, "swap": 0, "disk": 20, "id": 2 } ] }
```

Compute API Server Information Examples

GET /v1.1/servers

Gives you a listing of active servers under your account and tenant authorized with a particular token. If there are no servers running, you see just `{"servers": []}` returned but still get a 200 OK response.

Curl example:

```
curl -D - -H "X-Auth-Token: 3017813d-deca-4764-aa33-9ab59a283ba6" -X 'GET' -v http://host.na.me:8774/v1.1/9/servers
```

GET /v1.1/servers/detail

Shows details of all the active servers currently running under your account.

Curl example:

```
curl -D - -H "X-Auth-Token: 3017813d-deca-4764-aa33-9ab59a283ba6" -X 'GET' -v http://demo.rcb.me:8774/v1.1/9/servers/detail
```

POST /v1.1/servers

GET /v1.1/servers/*id*

PUT /v1.1/servers/*id*

DELETE /v1.1/servers/*id*

Compute API Server Actions Examples

POST /v1.1/servers/*id*/reboot

POST /v1.1/servers/*id*/rebuild

POST /v1.1/servers/*id*/resize?

POST /v1.1/servers/*id*/confirmResize?

POST /v1.1/servers/*id*/revertResize?

Compute API Extensions Examples

GET /v1.1/extensions

Shows the installed extensions to the API, which enable more requests and actions for the Compute server.

Curl example:

```
curl -D - -H "X-Auth-Token: 8528179e-0d07-44e9-b972-7427336c94cb" -X 'GET' -v  
http://demo.rcb.me:8774/v1.1/9/extensions
```

```
{ "extensions": [ { "updated": "2011-06-29T00:00:00+00:00",  
  "name": "Hosts", "links": [],  
  "namespace": "http://docs.openstack.org/ext/hosts/api/v1.1",  
  "alias": "os-hosts", "description": "Host administration" },  
  { "updated": "2011-03-25T00:00:00+00:00",  
    "name": "Volumes", "links": [], "namespace": "http://docs.openstack.org/ext/  
volumes/api/v1.1", "alias": "os-volumes", "description": "Volumes support" },  
    { "updated": "2011-05-25 16:12:21.656723",  
      "name": "Admin Controller", "links": [], "namespace": "http://TODO/", "alias":  
      "ADMIN", "description": "The Admin API Extension" },  
      { "updated": "2011-08-08T00:00:00+00:00",  
        "name": "Quotas", "links": [], "namespace": "http://docs.openstack.org/ext/  
quotas-sets/api/v1.1", "alias": "os-quota-sets", "description": "Quotas  
management support" },  
        { "updated": "2011-08-18T00:00:00+00:00",  
          "name": "Rescue", "links": [], "namespace": "http://docs.openstack.org/ext/  
rescue/api/v1.1", "alias": "os-rescue", "description": "Instance rescue  
mode" },  
          { "updated": "2011-06-09T00:00:00+00:00",  
            "name": "Multinic", "links": [], "namespace": "http://docs.openstack.org/  
ext/multinic/api/v1.1", "alias": "NMN", "description": "Multiple network  
support" },  
            { "updated": "2011-09-14T00:00:00+00:00",
```

```
{
  "name": "FlavorExtraData", "links": [], "namespace": "http://
docs.openstack.org/ext/flavor_extra_data/api/v1.1", "alias": "os-flavor-extra-
data", "description": "Provide additional data for flavors"},
{
  "updated": "2011-07-21T00:00:00+00:00",
  "name": "SecurityGroups", "links": [], "namespace": "http://
docs.openstack.org/ext/securitygroups/api/v1.1", "alias": "security_groups",
  "description": "Security group support"},
{
  "updated": "2011-06-23T00:00:00+00:00",
  "name": "FlavorExtraSpecs", "links": [], "namespace": "http://
docs.openstack.org/ext/flavor_extra_specs/api/v1.1", "alias": "os-flavor-
extra-specs", "description": "Instance type (flavor) extra specs"},
{
  "updated": "2011-07-19T00:00:00+00:00",
  "name": "Createserverext", "links": [], "namespace": "http://
docs.openstack.org/ext/createserverext/api/v1.1", "alias": "os-create-server-
ext", "description": "Extended support to the Create Server v1.1 API"},
{
  "updated": "2011-08-08T00:00:00+00:00",
  "name": "Keypairs", "links": [], "namespace": "http://docs.openstack.org/
ext/keypairs/api/v1.1", "alias": "os-keypairs", "description": "Keypair
  Support"},
{
  "updated": "2011-08-25T00:00:00+00:00",
  "name": "VSAs", "links": [], "namespace": "http://docs.openstack.org/ext/
vsa/api/v1.1", "alias": "zadr-vsa", "description": "Virtual Storage Arrays
  support"},
{
  "updated": "2011-08-19T00:00:00+00:00",
  "name": "SimpleTenantUsage", "links": [], "namespace": "http://
docs.openstack.org/ext/os-simple-tenant-usage/api/v1.1", "alias": "os-simple-
tenant-usage", "description": "Simple tenant usage extension"},
{
  "updated": "2011-08-24T00:00:00+00:00",
  "name": "VolumeTypes", "links": [], "namespace": "http://docs.openstack.org/
ext/volume_types/api/v1.1", "alias": "os-volume-types", "description": "Volume
  types support"},
{
  "updated": "2011-08-17T00:00:00+00:00",
  "name": "VirtualInterfaces", "links": [], "namespace": "http://
docs.openstack.org/ext/virtual_interfaces/api/v1.1", "alias":
  "virtual_interfaces", "description": "Virtual interface support"},
{
  "updated": "2011-06-16T00:00:00+00:00",
  "name": "Floating_ips", "links": [], "namespace": "http://docs.openstack.org/
ext/floating_ips/api/v1.1", "alias": "os-floating-ips", "description":
  "Floating IPs support"}
}]
```

Image API Examples

GET /images

Curl example:

```
curl -D - -H "X-Auth-Token: 469dbbaa-40f2-418a-aafe-1abcf03113f7" -X 'GET' -v
http://demo.rcb.me:9292/v1/images
```

```
{
  "images": [
    {
      "name": "uec-oneiric", "container_format": "ami", "disk_format":
      "ami", "checksum": "f8b72b5d643152c1beaa5d15555c9353", "id": 7, "size":
      1476395008},
    {
      "name": "uec-oneiric-kernel", "container_format": "aki", "disk_format":
      "aki", "checksum": "0577116c1df5e3d39aa33aee53a64e2b", "id": 6, "size":
      4731376},
    {
      "name": "uec-natty", "container_format": "ami", "disk_format": "ami",
      "checksum": "1808dfa4fb62f3804707eb0beeb7fa03", "id": 5, "size":
      1476395008},
  ]
}
```

```
{ "name": "uec-natty-kernel", "container_format": "aki", "disk_format": "aki",  
  "checksum": "af6261015c5466ad672c3ed18cecae04", "id": 4, "size": 4594816 },  
{ "name": "tty", "container_format": "ami", "disk_format": "ami", "checksum":  
  "10047a119149e08fb206eea89832eee0", "id": 3, "size": 25165824 },  
{ "name": "tty-ramdisk", "container_format": "ari", "disk_format": "ari",  
  "checksum": "2d222d406f3ed30e03ed44123c33cba6", "id": 2, "size": 5882349 },  
{ "name": "tty-kernel", "container_format": "aki", "disk_format": "aki",  
  "checksum": "3ed2965d3f8d877a3ee3e061fd648e9a", "id": 1, "size": 4404752 } ] }
```

GET /images/details

GET /images/*id*

POST /images

PUT /images/*id* (with metadata headers)

GET /images/*id*/members (knowing who can view an image)

PUT /images/*id*/members (replaces membership list for image)

GET /shared-images/tenant-id (shares an image with a tenant, or makes it public)

DELETE /images/1/members/tenant1 (revokes tenant's access to an image)

Identity API Examples

POST /v2.0/tokens

GET /v2.0/tenants

Object API Examples

Query Parameters

limit For an integer value *n*, limits the number of results to at most *n* values.

marker Given a string value *x*, return object names greater in value than the specified marker.

format Specify either `json` or `xml` to return the respective serialized response, for example append

```
?format=json
```

onto a query to inform the service that you want the information serialized in JSON when it is returned to you.

At this time, a `prefix` query parameter is not supported at the account level.

Storage Account Services

GET /v1.0/<account>

HEAD /v1.0/<account>

Storage Container Services

GET /v1.0/<account>/<container>

HEAD /v1.0/<account>/<container>

PUT /v1.0/<account>/<container>

DELETE /v1.0/<account>/<container>

Storage Object Services

GET /v1.0/<account>/<container>/<object>

HEAD /v1.0/<account>/<container>/<object>

PUT /v1.0/<account>/<container>/<object>

DELETE /v1.0/<account>/<container>/<object>

POST /v1.0/<account>/<container>/<object>