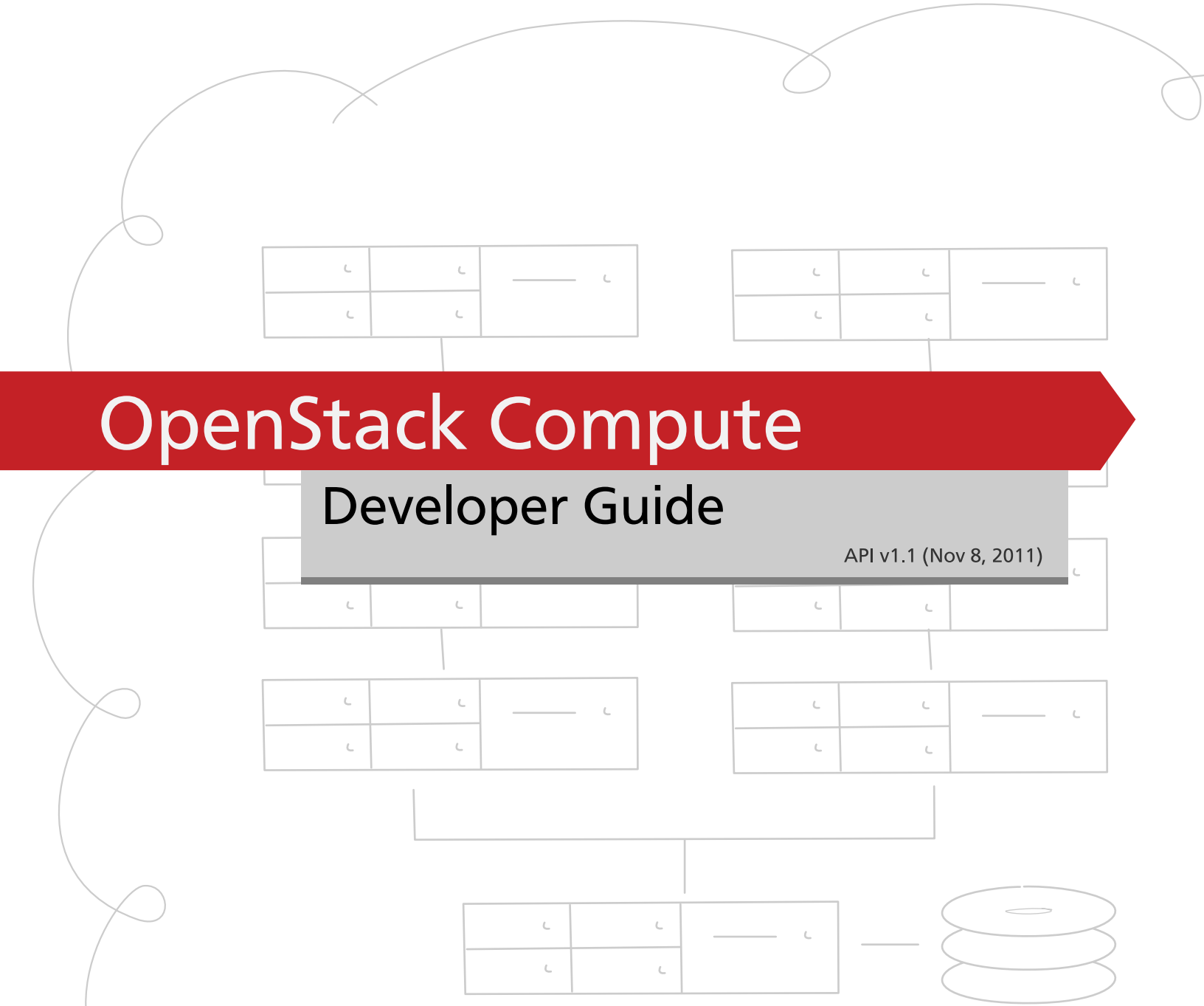
OpenStack Compute

Developer Guide

API v1.1 (Nov 8, 2011)



openstack™

docs.openstack.org

OpenStack Compute Developer Guide

API v1.1 (2011-11-08)

This document is intended for software developers interested in developing applications using the OpenStack Compute Application Programming Interface (API).

Table of Contents

List of Tables

List of Examples

1. Overview

OpenStack Compute is a compute service that provides server capacity in the cloud. Compute Servers come in different flavors of memory, cores, disk space, and CPU, and can be provisioned in minutes. Interactions with Compute Servers can occur programmatically via the OpenStack Compute API.

We welcome feedback, comments, and bug reports at bugs.launchpad.net/nova.

1.1. Intended Audience

This Guide is intended to assist software developers who want to develop applications using the OpenStack Compute API. To use the information provided here, you should first have a general understanding of the OpenStack Compute service and have access to an account from an OpenStack Compute provider. You should also be familiar with:

- ReSTful web services
- HTTP/1.1
- JSON and/or XML data serialization formats

1.2. Document Change History

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

Revision Date	Summary of Changes
November 8, 2011	Removed DRAFT designation.
September 8, 2011	• Added *limit* and *marker* parameters to list operations. • The rebuild action behaves just like create: an imageRef is used and a password may be specified. • Added tenant and userId attributes to server and image. • Added vcpus attribute to flavors. • We now use a flavorRef in the resize action.
July 23, 2011	• Added missing response examples for server update. • Ensure consistent HTTP status codes for all resources. • Clarifications on setting and changing a server password. • Minor updates to metadata section for clarity. • Discuss alternate links. • Removed version number from compute media types — use a media type parameter instead. • Bought back the flavorRef and imageRef server attributes these are now only used when creating a server. • Made the create image operation a server action. • Added minDisk and minRam filters to flavor lists. • Added minDisk and minRam attributes to images. • Asynchronous faults may now contain a timestamp. • Changes-since request returns an empty list rather than a 304. • Added DELETED image status. • Fix content length in Example 3.2, “Response with Headers: XML”. • Fixed bad request error code in Section 4.1.2.1, “Server Passwords”. • Compact image, server, and flavor lists should contain IDs, names, and links (Any kind of link may be included — not just self links). • Changed metadata URI from .../meta to .../metadata for consistency.
June 29, 2011	• Renamed Primary IP to Access IP.
June 23, 2011	• Many minor updates based on community feedback. • Removed sections on Content Compression, Persistent Connections, and Caching — these are operator specific. Added section on HTTP. • A Location header is returned when creating servers/images. • Added filters to collection of Image, Servers, and Flavors. • Added asynchronous faults. • Updates to links and references. Remove serverRef, imageRef, and flavorRef and instead embed one entity in another to provide links. • Added primary IP addresses. • Added forbidden fault. • We now use a single bookmark link per entity regardless of mimetype. • Collections are now sorted by create time. • Previous links are no longer required. • Added the ability to create or update multiple metadata items simultaneously. • Minor cleanups to server and image state machine. • Update to JSON collection format. • Replace integer IDs with UUIDs. • Removed affinityID, this will likely come in as an extension.
April 25, 2011	• Some minor cleanups in preparation for OpenStack Summit discussion.
Mar. 1, 2011	• Many minor updates based on community feedback. • Updates to resource linking and references. • Better description of paginated collections. • Metadata supported in servers and images. • Dropped support for shared IP groups. • IPs organized by network id, vs simply having public and private IPs. • Generalized affinity id.
Feb. 9, 2011	• Initial release.

1.3. Additional Resources

You can download the most current version of this document from the OpenStack Docs website at http://docs.openstack.org.

For more details about the Cloud Servers service that this API is based upon, please refer to http://www.rackspacecloud.com/cloud_hosting_products/servers. Related documents, including an API Language Binding Guide, are available at the same site, as are links to Rackspace's official support channels, including knowledge base articles, forums, phone, chat, and email.

2. Concepts

To use the OpenStack Compute API effectively, you should understand several key concepts:

2.1. Server

A server is a virtual machine instance in the compute system. Flavor and image are requisite elements when creating a server.

2.2. Flavor

A flavor is an available hardware configuration for a server. Each flavor has a unique combination of disk space, memory capacity and priority for CPU time.

2.3. Image

An image is a collection of files used to create or rebuild a server. Operators provide a number of pre-built OS images by default. You may also create custom images from cloud servers you have launched. These custom images are useful for backup purposes or for producing "gold" server images if you plan to deploy a particular server configuration frequently.

2.4. Reboot

The reboot function allows for either a soft or hard reboot of a server. With a soft reboot, the operating system is signaled to restart, which allows for a graceful shutdown of all processes. A hard reboot is the equivalent of power cycling the server.

2.5. Rebuild

The rebuild function removes all data on the server and replaces it with the specified image. Server ID and IP addresses remain the same.

2.6. Resize

The resize function converts an existing server to a different flavor, in essence, scaling the server up or down. The original server is saved for a period of time to allow rollback if there is a problem. All resizes should be tested and explicitly confirmed, at which time the original server is removed. All resizes are automatically confirmed after 24 hours if they are not confirmed or reverted.

3. General API Information

The OpenStack Compute API is defined as a ReSTful HTTP service. The API takes advantage of all aspects of the HTTP protocol (methods, URIs, media types, response codes, etc.) and providers are free to use existing features of the protocol such as caching, persistent connections, and content compression among others. For example, providers who employ a caching layer may respond with a 203 when a request is served from the cache instead of a 200. Additionally, providers may offer support for conditional **GET** requests using ETags, or they may send a redirect in response to a **GET** request. Clients should be written to account for these differences.

3.1. Authentication

Each HTTP request against the OpenStack Compute system requires the inclusion of specific authentication credentials. A single deployment may support multiple authentication schemes (OAuth, Basic Auth, Token). The authentication scheme used is determined by the provider of the OpenStack Compute system. Please contact contact your provider to determine the best way to authenticate against this API.

Note

Some authentication schemes may require that the API operate using SSL over HTTP (HTTPS).

3.2. Request/Response Types

The OpenStack Compute API supports both the JSON and XML data serialization formats. The request format is specified using the `Content-Type` header and is required for operations that have a request body. The response format can be specified in requests using either the `Accept` header or adding an .xml or .json extension to the request URI. Note that it is possible for a response to be serialized using a format different from the request (see example below). If no response format is specified, JSON is the default. If conflicting formats are specified using both an `Accept` header and a query extension, the query extension takes precedence.

Table 3.1. JSON and XML Response Formats

Format	Accept Header	Query Extension	Default
JSON	application/json	.json	Yes
XML	application/xml	.xml	No

Example 3.1. Request with Headers: JSON

```
POST /v1.1/214412/servers HTTP/1.1
Host: servers.api.openstack.org
Content-Type: application/json
Accept: application/xml
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
{
    "server" : {
        "name" : "new-server-test",
        "imageRef" : "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54",
        "flavorRef" : "52415800-8b69-11e0-9b19-734f1195ff37",
        "metadata" : {
            "My Server Name" : "Apache1"
        },
        "personality" : [
            {
                "path" : "/etc/banner.txt",
                "contents" : "ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
 dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
 IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
 c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
 QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
 ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
 dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
 c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
 b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
            }
        ]
    }
}
```

Example 3.2. Response with Headers: XML

```
HTTP/1.1 200 OK
Date: Mon, 12 Nov 2007 15:55:01 GMT
Server: Apache
Content-Length: 1863
Content-Type: application/xml; charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom"
        id="52415800-8b69-11e0-9b19-734f565bc83b"
        tenantId="1234" userId="5678"
        name="new-server-test"
        hostId="e4d909c290d0fb1ca068ffaddf22cbd0" progress="0"
        status="BUILD" adminPass="GFf1j9aP"
        created="2010-11-11T12:00:00Z"
        accessIPv4="67.23.10.138"
        accessIPv6="::babe:67.23.10.138">
  <image id="52415800-8b69-11e0-9b19-734f6f006e54"
         name="CentOS 5.2">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
  </image>
  <flavor id="52415800-8b69-11e0-9b19-734f1195ff37"
          name="256 MB Server">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
  </flavor>
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <addresses>
    <network id="public">
      <ip version="4" addr="67.23.10.138"/>
      <ip version="6" addr="::babe:67.23.10.138"/>
    </network>
    <network id="private">
      <ip version="4" addr="10.176.42.19"/>
      <ip version="6" addr="::babe:10.176.42.19"/>
    </network>
  </addresses>
  <atom:link
      rel="self"
      href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
  <atom:link
      rel="bookmark"
      href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
```

```
</server>
```

Notice, in the above example, that the content type is set to application/json but it asks for an application/xml response with the `Accept` header. An alternative method of achieving the same result is illustrated below – this time we utilize a URI extension instead of an `Accept` header.

Example 3.3. Request with Extension: JSON

```
POST /v1.1/214412/servers.xml HTTP/1.1
Host: servers.api.openstack.org
Content-Type: application/json
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

```
{
    "server" : {
        "name" : "new-server-test",
        "imageRef" : "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54",
        "flavorRef" : "52415800-8b69-11e0-9b19-734f1195ff37",
        "metadata" : {
            "My Server Name" : "Apache1"
        },
        "personality" : [
            {
                "path" : "/etc/banner.txt",
                "contents" : "ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
 dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
 IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
 c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
 QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
 ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
 dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
 c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
 b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
            }
        ]
    }
}
```

3.3. Links and References

Often resources need to refer to other resources. For example, when creating a server, you must specify the image from which to build the server. You can specify the image by providing an ID or a URL to a remote image. When providing an ID, it is assumed that the resource exists in the current OpenStack deployment.

Example 3.4. ID Image Reference: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom"
        name="new-server-test"
        imageRef="52415800-8b69-11e0-9b19-734f6f006e54"
        flavorRef="52415800-8b69-11e0-9b19-734f1195ff37"
        >
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <personality>
    <file path="/etc/banner.txt">
        ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
        dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
        IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
        c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
        QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
        ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
        dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
        c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
        b25zLiINCg0KLVJpY2hhcmQgQmFjaA==
    </file>
  </personality>
</server>
```

Example 3.5. ID Image Reference: JSON

```
{
    "server": {
        "name": "new-server-test",
        "image": "52415800-8b69-11e0-9b19-734f6f006e54",
        "flavor": "52415800-8b69-11e0-9b19-734f1195ff37",
        "metadata": {
            "My Server Name": "Apache1"
        },
        "personality": [
            {
                "path": "/etc/banner.txt",
                "contents": "ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
 dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
 IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
 c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
 QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
 ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
 dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
 c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
 b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
            }
        ]
    }
}
```

Example 3.6. Full Image Reference: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        imageRef="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
        flavorRef="52415800-8b69-11e0-9b19-734f1195ff37"
        name="new-server-test"
        >
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <personality>
    <file path="/etc/banner.txt">
        ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
        dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
        IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
        c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
        QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
        ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
        dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
        c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
        b25zLiINCg0KLVJpY2hhcmQgQmFjaA==
    </file>
  </personality>
</server>
```

Example 3.7. Full Image Reference: JSON

```
{
    "server" : {
        "name" : "new-server-test",
        "imageRef" : "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54",
        "flavorRef" : "52415800-8b69-11e0-9b19-734f1195ff37",
        "metadata" : {
            "My Server Name" : "Apache1"
        },
        "personality" : [
            {
                "path" : "/etc/banner.txt",
                "contents" : "ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
 dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
 IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
 c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
 QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
 ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
 dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
 c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
 b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
            }
        ]
    }
}
```

For convenience, resources contain links to themselves. This allows a client to easily obtain a resource URIs rather than to construct them. There are three kinds of link relations associated with resources. A `self` link contains a versioned link to the resource. These links should be used in cases where the link will be followed immediately. A `bookmark` link provides a permanent link to a resource that is appropriate for long term storage. An `alternate` link can contain an alternate representation of the resource. For example, an OpenStack Compute image may have an alternate representation in the OpenStack Image service. Note that the type attribute here is used to provide a hint as to the type of representation to expect when following the link.

Example 3.8. Server with Self Links: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom"
        id="52415800-8b69-11e0-9b19-734fcece0043" name="my-server">
  <atom:link
      rel="self"
      href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
  <atom:link
      rel="bookmark"
      href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
</server>
```

Example 3.9. Server with Self Links: JSON

```
{
    "server" : {
        "id" : "52415800-8b69-11e0-9b19-734fcece0043",
        "name" : "my-server",
        "links": [
            {
                "rel" : "self",
                "href" : "http://servers.api.openstack.org/v1.1/1234/
servers/52415800-8b69-11e0-9b19-734fcece0043"
            },
            {
                "rel" : "bookmark",
                "href" : "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"
            }
        ]
    }
}
```

Example 3.10. Image with Alternate Link: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<image
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    xmlns:atom="http://www.w3.org/2005/Atom"
    id="52415800-8b69-11e0-9b19-734f5736d2a2"
    name="My Server Backup">
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
    <atom:link
        rel="alternate" type="application/vnd.openstack.image"
        href="http://glance.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
</image>
```

Example 3.11. Server with Alternate Link: JSON

```
{
    "image" : {
        "id" : "52415800-8b69-11e0-9b19-734f5736d2a2",
        "name" : "My Server Backup",
        "links": [
            {
                "rel" : "self",
                "href" : "http://servers.api.openstack.org/v1.1/1234/
images/52415800-8b69-11e0-9b19-734f5736d2a2"
            },
            {
                "rel" : "bookmark",
                "href" : "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
            },
            {
                "rel"  : "alternate",
                "type" : "application/vnd.openstack.image",
                "href" : "http://glance.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
            }
        ]
    }
}
```

3.4. Paginated Collections

To reduce load on the service, list operations will return a maximum number of items at a time. The maximum number of items returned is determined by the compute provider. To navigate the collection, the parameters *limit* and *marker* can be set in the URI (e.g.?*limit*=100&*marker*=1234). The *marker* parameter is the ID of the last item in the previous list. Items are sorted by create time in descending order. When a create time is not available they are sorted by ID. The *limit* parameter sets the page size. Both parameters are optional. If the client requests a *limit* beyond that which is supported by the deployment an overLimit (413) fault may be thrown. A marker with an invalid ID will return an itemNotFound (404) fault.

Note

Paginated collections never return itemNotFound (404) faults when the collection is empty — clients should expect an empty collection.

For convenience, collections are required to contain atom "next" links. They may optionally also contain "previous" links. The last page in the list will not contain a "next" link. The following examples illustrate three pages in a collection of images. The first page was retrieved via a **GET** to http://servers.api.openstack.org/v1.1/1234/images?limit=1. In these examples, the *limit* parameter sets the page size to a single item. Subsequent links will honor the initial page size. Thus, a client may follow links to traverse a paginated collection without having to input the *marker* parameter.

Example 3.12. Images Collection, First Page: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<images xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom">
  <image id="52415800-8b69-11e0-9b19-734f6f006e54"
         name="CentOS 5.2">
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
  </image>
  <atom:link
      rel="next"
      href="http://servers.api.openstack.org/v1.1/1234/images?limit=1&
amp;marker=52415800-8b69-11e0-9b19-734f6f006e54"/>
</images>
```

Example 3.13. Images Collection, First Page: JSON

```
{
    "images": [
        {
            "id": "52415800-8b69-11e0-9b19-734f6f006e54",
            "name": "CentOS 5.2",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
images/52415800-8b69-11e0-9b19-734f6f006e54"
                }
            ]
        }
    ],
    "images_links" : [
        {
            "rel": "next",
            "href": "http://servers.api.openstack.org/v1.1/1234/images?limit=
1&marker=52415800-8b69-11e0-9b19-734f6f006e54"
        }
    ]
}
```

Example 3.14. Images Collection, Second Page: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<images xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom">
  <image id="52415800-8b69-11e0-9b19-734f5736d2a2" name="My Server Backup">
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
  </image>
  <atom:link rel="next"
             href="http://servers.api.openstack.org/v1.1/1234/images?limit=1&
amp;marker=52415800-8b69-11e0-9b19-734f5736d2a2"/>
</images>
```

Example 3.15. Images Collection, Second Page: JSON

```
{
    "images" :  [
            {
                "id" : "52415800-8b69-11e0-9b19-734f5736d2a2",
                "name" : "My Server Backup",
                "links": [
                    {
                        "rel" : "self",
                        "href" : "http://servers.api.openstack.org/v1.1/1234/
images/52415800-8b69-11e0-9b19-734f5736d2a2"
                    }
                ]
            }
        ],
    "images_links": [
        {
            "rel" : "next",
            "href" : "http://servers.api.openstack.org/v1.1/1234/images?limit=
1&marker=52415800-8b69-11e0-9b19-734f5736d2a2"
        }
    ]
}
```

Example 3.16. Images Collection, Last Page: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<images xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom">
  <image id="52415800-8b69-11e0-9b19-734f6ff7c475"
         name="Backup 2">
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6ff7c475"/>
  </image>
</images>
```

Example 3.17. Images Collection, Last Page: JSON

```
{
    "images": [
        {
            "id": "52415800-8b69-11e0-9b19-734f6ff7c475",
            "name": "Backup 2",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
images/52415800-8b69-11e0-9b19-734f6ff7c475"
                }
            ]
        }
    ]
}
```

In JSON, members in a paginated collection are stored in a JSON array named after the collection. A JSON object may also be used to hold members in cases where using an associative array is more practical. Properties about the collection itself, including links, are contained in an array with the name of the entity an underscore (_) and `links`. The combination of the objects and arrays that start with the name of the collection and an underscore represent the collection in JSON. The approach allows for extensibility of paginated collections by allowing them to be associated with arbitrary properties. It also allows collections to be embedded in other objects as illustrated below. Here, a subset of metadata items are presented within the image. Clients must follow the "next" link to retrieve the full set of metadata.

Example 3.18. Paginated Metadata in an Image: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<image
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    xmlns:atom="http://www.w3.org/2005/Atom"
    id="52415800-8b69-11e0-9b19-734f6f006e54"
    name="CentOS 5.2">
    <metadata>
        <meta key="ImageVersion">1.5</meta>
        <meta key="ImageType">Gold</meta>
        <atom:link
            rel="next"
            href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54/meta?marker=ImageType"/>
    </metadata>
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
</image>
```

Example 3.19. Paginated Metadata in an Image: JSON

```
{
    "image": {
        "id": "52415800-8b69-11e0-9b19-734f6f006e54",
        "name": "CentOS 5.2",
        "metadata": {
            "ImageVersion": "1.5",
            "ImageType": "Gold"
        },
        "metadata_links": [
            {
                "rel": "next",
                "href": "http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54/meta?marker=ImageType"
            }
        ],
        "links": [
            {
                "rel": "self",
                "href": "http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
            }
        ]
    }
}
```

3.5. Efficient Polling with the *changes-since* Parameter

The ReST API allows you to poll for the status of certain operations by performing a **GET** on various elements. Rather than re-downloading and re-parsing the full status at each polling interval, your ReST client may use the *changes-since* parameter to check for changes since a previous request. The *changes-since* time is specified as an ISO 8601 dateTime (2011-01-24T17:08Z). The form for the timestamp is CCYY-MM-DDThh:mm:ss. An optional time zone may be written in by appending the form ±hh:mm which describes the timezone as an offset from UTC. When the timezone is not specified (2011-01-24T17:08), the UTC timezone will be assumed. If nothing has changed since the *changes-since* time, an empty list will be returned. If data has changed, only the items changed since the specified time will be returned in the response. For example, performing a **GET** against https://api.servers.openstack.org/v1.1/224532/servers?*changes-since*=2011-01-24T17:08Z would list all servers that have changed since Mon, 24 Jan 2011 17:08:00 UTC.

In order to allow clients to keep track of changes, the changes-since filter displays items that have been *recently* deleted. Both images and servers contain a DELETED status that indicates that the resource has been removed. Implementations are not required to keep track of deleted resources indefinitely, so sending a changes since time in the distant past may miss deletions.

3.6. Limits

Accounts may be pre-configured with a set of thresholds (or limits) to manage capacity and prevent abuse of the system. The system recognizes two kinds of limits: *rate limits* and *absolute limits*. Rate limits are thresholds that are reset after a certain amount of time passes. Absolute limits are fixed. Limits are configured by operators and may differ from one deployment of the OpenStack Compute service to another. Please contact your provider to determine the limits that apply to your account or see Section 3.6.3, “Determining Limits Programmatically”. Your provider may be able to adjust your account's limits if they are too low.

3.6.1. Rate Limits

Rate limits are specified in terms of both a human-readable wild-card URI and a machine-processable regular expression. The human-readable limit is intended for displaying in graphical user interfaces. The machine-processable form is intended to be used directly by client applications.

The regular expression boundary matcher "^" for the rate limit takes effect after the root URI path. For example, the regular expression ^/servers would match the bolded portion of the following URI: https://servers.api.openstack.org/v1.1/3542812**/servers**.

Table 3.2. Sample Rate Limits

Verb	URI	RegEx	Default
POST	*	.*	10/min
POST	*/servers	^/servers	50/day
PUT	*	.*	10/min
GET	*changes-since*	changes-since	3/min
DELETE	*	.*	100/min

Rate limits are applied in order relative to the verb, going from least to most specific. For example, although the threshold for **POST** to */servers is 50 per day, one cannot **POST** to */servers more than 10 times within a single minute because the rate limits for any **POST** is 10/min.

In the event a request exceed the thresholds established for your account, a 413 HTTP response will be returned with a `Retry-After` header to notify the client when they can attempt to try again.

3.6.2. Absolute Limits

Absolute limits are specified as name/value pairs. The name of the absolute limit uniquely identifies the limit within a deployment. Please consult your provider for an exhaustive list of absolute value names. An absolute limit value is always specified as an integer. The name of the absolute limit determines the unit type of the integer value. For example, the name maxServerMeta implies that the value is in terms of server metadata items.

Table 3.3. Sample Absolute Limits

Name	Value	Description
maxTotalRAMSize	51200	Maximum total amount of RAM (MB)
maxServerMeta	5	Maximum number of metadata items associated with a server
maxImageMeta	5	Maximum number of metadata items associated with an Image
maxPersonality	5	The maximum number of file path/content pairs that can be supplied on server build
maxPersonalitySize	10240	The maximum size, in bytes, for each personality file

3.6.3. Determining Limits Programmatically

Applications can programmatically determine current account limits using the /limits URI as follows:

Verb	URI	Description
GET	/limits	Returns the current limits for your account

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413)

This operation does not require a request body.

Example 3.20. Limit Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<limits xmlns="http://docs.openstack.org/common/api/v1.0">
  <rates>
      <rate uri="*" regex=".*">
          <limit value="10" verb="POST" remaining="2"
                 unit="MINUTE"
                 next-available="2011-12-15T22:42:45Z"/>
          <limit value="10" verb="PUT" remaining="2"
                 unit="MINUTE"
                 next-available="2011-12-15T22:42:45Z"/>
          <limit value="100" verb="DELETE" remaining="100"
                 unit="MINUTE"
                 next-available="2011-12-15T22:42:45Z"/>
      </rate>
      <rate uri="*changes-since*" regex="changes-since">
          <limit value="3" verb="GET" remaining="3"
                 unit="MINUTE"
                 next-available="2011-12-15T22:42:45Z"/>
      </rate>
      <rate uri="*/servers" regex="^/servers">
          <limit verb="POST" value="25" remaining="24"
                 unit="DAY"
                 next-available="2011-12-15T22:42:45Z"/>
      </rate>
  </rates>
  <absolute>
    <limit name="maxTotalRAMSize" value="51200"/>
    <limit name="maxServerMeta" value="5"/>
    <limit name="maxImageMeta" value="5"/>
    <limit name="maxPersonality" value="5"/>
    <limit name="maxPersonalitySize" value="10240"/>
  </absolute>
</limits>
```

Example 3.21. Limit Response: JSON

```
{
    "limits": {
        "rate": [
            {
                "uri": "*",
                "regex": ".*",
                "limit": [
                    {
                        "value": 10,
                        "verb": "POST",
                        "remaining": 2,
                        "unit": "MINUTE",
                        "next-available": "2011-12-15T22:42:45Z"
                    },
                    {
                        "value": 10,
                        "verb": "PUT",
                        "remaining": 2,
                        "unit": "MINUTE",
                        "next-available": "2011-12-15T22:42:45Z"
                    },
                    {
                        "value": 100,
                        "verb": "DELETE",
                        "remaining": 100,
                        "unit": "MINUTE",
                        "next-available": "2011-12-15T22:42:45Z"
                    }
                ]
            },
            {
                "uri": "*changes-since*",
                "regex": "changes-since",
                "limit": [
                    {
                        "value": 3,
                        "verb": "GET",
                        "remaining": 3,
                        "unit": "MINUTE",
                        "next-available": "2011-12-15T22:42:45Z"
                    }
                ]
            },
            {
                "uri": "*/servers",
                "regex": "^/servers",
                "limit": [
                    {
                        "verb": "POST",
                        "value": 25,
                        "remaining": 24,
                        "unit": "DAY",
                        "next-available": "2011-12-15T22:42:45Z"
                    }
                ]
            }
        ],
        "absolute": {
            "maxTotalRAMSize": 51200,
            "maxServerMeta": 5,
            "maxImageMeta": 5,
            "maxPersonality": 5,
            "maxPersonalitySize": 10240
        }
    }
}
```

3.7. Versions

The OpenStack Compute API uses both a URI and a MIME type versioning scheme. In the URI scheme, the first element of the path contains the target version identifier (e.g. https://servers.api.openstack.org/ v1.0/...). The MIME type versioning scheme uses HTTP content negotiation where the `Accept` or `Content-Type` headers contains a MIME type that identifies the version (application/vnd.openstack.compute+xml;version=1.1). A version MIME type is always linked to a base MIME type (application/xml or application/json). If conflicting versions are specified using both an HTTP header and a URI, the URI takes precedence.

Example 3.22. Request with MIME type versioning

```
GET /214412/images HTTP/1.1
Host: servers.api.openstack.org
Accept: application/vnd.openstack.compute+xml;version=1.1
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

Example 3.23. Request with URI versioning

```
GET /v1.1/214412/images HTTP/1.1
Host: servers.api.openstack.org
Accept: application/xml
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

Note

The MIME type versioning approach allows for the creating of permanent links, because the version scheme is not specified in the URI path: https://api.servers.openstack.org/224532/servers/123.

If a request is made without a version specified in the URI or via HTTP headers, then a multiple-choices response (300) will follow providing links and MIME types to available versions.

Example 3.24. Multiple Choices Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>

<choices xmlns="http://docs.openstack.org/common/api/v1.0"
         xmlns:atom="http://www.w3.org/2005/Atom">
 <version id="v1.0" status="DEPRECATED">
     <media-types>
         <media-type base="application/xml"
            type="application/vnd.openstack.compute+xml;version=1.0"/>
         <media-type base="application/json"
            type="application/vnd.openstack.compute+json;version=1.0"/>
     </media-types>

     <atom:link rel="self"
                href="http://servers.api.openstack.org/v1.0/1234/servers/
52415800-8b69-11e0-9b19-734f6af67565"/>
    </version>

    <version id="v1.1" status="CURRENT">
     <media-types>
         <media-type base="application/xml"
            type="application/vnd.openstack.compute+xml;version=1.1"/>
         <media-type base="application/json"
            type="application/vnd.openstack.compute+json;version=1.1"/>
     </media-types>

     <atom:link rel="self"
                href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f6af67565"/>
    </version>
</choices>
```

Example 3.25. Multiple Choices Response: JSON

```
{
    "choices": [
        {
            "id": "v1.0",
            "status": "DEPRECATED",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.0/1234/
servers/52415800-8b69-11e0-9b19-734f6af67565"
                }
            ],
            "media-types": [
                {
                    "base": "application/xml",
                    "type": "application/vnd.openstack.compute
+xml;version=1.0"
                },
                {
                    "base": "application/json",
                    "type": "application/vnd.openstack.compute
+json;version=1.0"
                }
            ]
        },
        {
            "id": "v1.1",
            "status": "CURRENT",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
servers/52415800-8b69-11e0-9b19-734f6af67565"
                }
            ],
            "media-types": [
                {
                    "base": "application/xml",
                    "type": "application/vnd.openstack.compute
+xml;version=1.0"
                },
                {
                    "base": "application/json",
                    "type": "application/vnd.openstack.compute
+json;version=1.0"
                }
            ]
        }
    ]
}
```

New features and functionality that do not break API-compatibility will be introduced in the current version of the API as extensions (see below) and the URI and MIME types will remain unchanged. Features or functionality changes that would necessitate a break in API-compatibility will require a new version, which will result in URI and MIME type version being updated accordingly. When new API versions are released, older versions will be marked as DEPRECATED. Providers should work with developers and partners to ensure there is adequate time to migrate to the new version before deprecated versions are discontinued.

Your application can programmatically determine available API versions by performing a **GET** on the root URL (i.e. with the version and everything to the right of it truncated) returned from the authentication system. Note that an Atom representation of the versions resources is supported when issuing a request with the `Accept` header containing application/atom+xml or by adding a .atom to the request URI. This allows standard Atom clients to track version changes.

Example 3.26. Versions List Request

```
GET HTTP/1.1
Host: servers.api.openstack.org
```

Normal Response Code(s): 200, 203

Error Response Code(s): 400, 413, 500, 503

This operation does not require a request body.

Example 3.27. Versions List Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>

<versions xmlns="http://docs.openstack.org/common/api/v1.0"
          xmlns:atom="http://www.w3.org/2005/Atom">

 <version id="v1.0" status="DEPRECATED"
          updated="2009-10-09T11:30:00Z">
     <atom:link rel="self"
                href="http://servers.api.openstack.org/v1.0/"/>
 </version>

 <version id="v1.1" status="CURRENT"
          updated="2010-12-12T18:30:02.25Z">
     <atom:link rel="self"
                href="http://servers.api.openstack.org/v1.1/"/>
 </version>

</versions>
```

Example 3.28. Versions List Response: Atom

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <title type="text">Available API Versions</title>
    <updated>2010-12-12T18:30:02.25Z</updated>
    <id>http://servers.api.openstack.org/</id>
    <author><name>Rackspace</name><uri>http://www.rackspace.com/</uri></
author>
    <link rel="self" href="http://servers.api.openstack.org/"/>
    <entry>
       <id>http://servers.api.openstack.org/v1.1/</id>
       <title type="text">Version v1.1</title>
       <updated>2010-12-12T18:30:02.25Z</updated>
       <link rel="self" href="http://servers.api.openstack.org/v1.1/"/>
       <content type="text">Version v1.1 CURRENT (2010-12-12T18:30:02.25Z)</
content>
    </entry>
    <entry>
       <id>http://servers.api.openstack.org/v1.0/</id>
       <title type="text">Version v1.0</title>
       <updated>2009-10-09T11:30:00Z</updated>
       <link rel="self" href="http://servers.api.openstack.org/v1.0/"/>
       <content type="text">Version v1.0 DEPRECATED (2009-10-09T11:30:00Z)</
content>
    </entry>
</feed>
```

Example 3.29. Versions List Response: JSON

```
{
    "versions": [
        {
            "id": "v1.0",
            "status": "DEPRECATED",
            "updated": "2009-10-09T11:30:00Z",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.0/"
                }
            ]
        },
        {
            "id": "v1.1",
            "status": "CURRENT",
            "updated": "2010-12-12T18:30:02.25Z",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/"
                }
            ]
        }
    ]
}
```

You can also obtain additional information about a specific version by performing a **GET** on the base version URL (e.g. https://servers.api.openstack.org/v1.1/). Version request URLs should always end with a trailing slash (/). If the slash is omitted, the server may respond with a 302 redirection request. Format extensions may be placed after the slash (e.g. https://servers.api.openstack.org/v1.1/.xml). Note that this is a special case that does not hold true for other API requests. In general, requests such as /servers.xml and /servers/.xml are handled equivalently.

Example 3.30. Version Details Request

```
GET HTTP/1.1
Host: servers.api.openstack.org/v1.1/
```

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413)

This operation does not require a request body

Example 3.31. Version Details Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<version xmlns="http://docs.openstack.org/common/api/v1.0"
         xmlns:atom="http://www.w3.org/2005/Atom"
         id="v1.1" status="CURRENT" updated="2011-01-21T11:33:21-06:00">

     <media-types>
         <media-type base="application/xml"
            type="application/vnd.openstack.compute+xml;version=1.1"/>
         <media-type base="application/json"
            type="application/vnd.openstack.compute+json;version=1.1"/>
     </media-types>

     <atom:link rel="self"
                href="http://servers.api.openstack.org/v1.0/"/>

    <atom:link rel="describedby"
               type="application/pdf"
               href="http://docs.rackspacecloud.com/servers/api/v1.1/cs-
devguide-20110125.pdf" />

    <atom:link rel="describedby"
               type="application/vnd.sun.wadl+xml"
               href="http://docs.rackspacecloud.com/servers/api/v1.1/
application.wadl" />
</version>
```

Example 3.32. Version Details Response: Atom

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text">About This Version</title>
  <updated>2011-01-21T11:33:21-06:00</updated>
  <id>http://servers.api.openstack.org/v1.0/</id>
   <author><name>Rackspace</name><uri>http://www.rackspace.com/</uri></author>
   <link rel="self" href="http://servers.api.openstack.org/v1.0/"/>
   <entry>
      <id>http://servers.api.openstack.org/v1.0/</id>
      <title type="text">Version v1.1</title>
      <updated>2011-01-21T11:33:21-06:00</updated>
      <link rel="self" href="http://servers.api.openstack.org/v1.0/"/>
      <link rel="describedby" type="application/pdf"
           href="http://docs.rackspacecloud.com/servers/api/v1.1/cs-
devguide-20110125.pdf"/>
      <link rel="describedby" type="application/vnd.sun.wadl+xml"
           href="http://docs.rackspacecloud.com/servers/api/v1.1/application.
wadl"/>
      <content type="text">Version v1.1 CURRENT (2011-01-21T11:33:21-06:00)</
content>
   </entry>
</feed>
```

Example 3.33. Version Details Response: JSON

```
{
    "version" : {
        "id" : "v1.1",
        "status" : "CURRENT",
        "updated" : "2011-01-21T11:33:21-06:00",
        "links": [
            {
                "rel" : "self",
                "href" : "http://servers.api.openstack.org/v1.0/"
            },
            {
                "rel" : "describedby",
                "type" : "application/pdf",
                "href" : "http://docs.rackspacecloud.com/servers/api/v1.1/
cs-devguide-20110125.pdf"
            },
            {
                "rel" : "describedby",
                "type" : "application/vnd.sun.wadl+xml",
                "href" : "http://docs.rackspacecloud.com/servers/api/v1.1/
application.wadl"
            }
        ],
        "media-types": [
            {
                "base" : "application/xml",
                "type" : "application/vnd.openstack.compute+xml;version=1.
1"
            },
            {
                "base" : "application/json",
                "type" : "application/vnd.openstack.compute+json;version=
1.1"
            }
        ]
    }
}
```

The detailed version response contains pointers to both a human-readable and a machine-processable description of the API service. The machine-processable description is written in the Web Application Description Language (WADL).

Note

If there is a discrepancy between the two specifications, the WADL is authoritative as it contains the most accurate and up-to-date description of the API service.

3.8. Extensions

The OpenStack Compute API is extensible. Extensions serve two purposes: They allow the introduction of new features in the API without requiring a version change and they allow the introduction of vendor specific niche functionality. Applications can programmatically determine what extensions are available by performing a **GET** on the /extensions URI. Note that this is a versioned request — that is, an extension available in one API version may not be available in another.

Verb	URI	Description
GET	/extensions	List all available extensions

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413)

This operation does not require a request body. Each extension is identified by two unique identifiers, a namespace and an alias. Additionally an extension contains documentation links in various formats.

Example 3.34. Extensions Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>

<extensions xmlns="http://docs.openstack.org/common/api/v1.0"
            xmlns:atom="http://www.w3.org/2005/Atom">

    <extension
        name="Public Image Extension"
        namespace="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.
0"
        alias="RAX-PIE"
        updated="2011-01-22T13:25:27-06:00">

        <description>
            Adds the capability to share an
            image with other users.
        </description>

        <atom:link rel="describedby"
                   type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/
cs-pie-20111111.pdf"/>
        <atom:link rel="describedby"
                   type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/
cs-pie.wadl"/>
    </extension>

    <extension
        name="Cloud Block Storage"
        namespace="http://docs.rackspacecloud.com/servers/api/ext/cbs/v1.
0"
        alias="RAX-CBS"
        updated="2011-01-12T11:22:33-06:00"
               >
        <description>
            Allows mounting cloud block
            storage volumes.
        </description>

        <atom:link rel="describedby"
                   type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/
cs-cbs-20111201.pdf"/>
        <atom:link rel="describedby"
                   type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/
cs-cbs.wadl"/>
    </extension>
</extensions>
```

Example 3.35. Extensions Response: JSON

```
{
    "extensions": [
        {
            "name": "Public Image Extension",
            "namespace": "http://docs.rackspacecloud.com/servers/api/ext/
pie/v1.0",
            "alias": "RAX-PIE",
            "updated": "2011-01-22T13:25:27-06:00",
            "description": "Adds the capability to share an image with
 other users.",
            "links": [
                {
                    "rel": "describedby",
                    "type": "application/pdf",
                    "href": "http://docs.rackspacecloud.com/servers/api/
ext/cs-pie-20111111.pdf"
                },
                {
                    "rel": "describedby",
                    "type": "application/vnd.sun.wadl+xml",
                    "href": "http://docs.rackspacecloud.com/servers/api/
ext/cs-pie.wadl"
                }
            ]
        },
        {
            "name": "Cloud Block Storage",
            "namespace": "http://docs.rackspacecloud.com/servers/api/ext/
cbs/v1.0",
            "alias": "RAX-CBS",
            "updated": "2011-01-12T11:22:33-06:00",
            "description": "Allows mounting cloud block storage volumes.",
            "links": [
                {
                    "rel": "describedby",
                    "type": "application/pdf",
                    "href": "http://docs.rackspacecloud.com/servers/api/
ext/cs-cbs-20111201.pdf"
                },
                {
                    "rel": "describedby",
                    "type": "application/vnd.sun.wadl+xml",
                    "href": "http://docs.rackspacecloud.com/servers/api/
ext/cs-cbs.wadl"
                }
            ]
        }
    ]
}
```

Extensions may also be queried individually by their unique alias. This provides the simplest method of checking if an extension is available as an unavailable extension will issue an itemNotFound (404) response.

Verb	URI	Description
GET	/extensions/*alias*	Get details about a specific extension

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404)

Example 3.36. Extension Response: xml

```
<?xml version="1.0" encoding="UTF-8"?>

<extension
    xmlns="http://docs.openstack.org/common/api/v1.0"
    xmlns:atom="http://www.w3.org/2005/Atom"
    name="Public Image Extension"
    namespace="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0"
    alias="RS-PIE"
    updated="2011-01-22T13:25:27-06:00">

    <description>
        Adds the capability to share an image with other users.
    </description>

    <atom:link
        rel="describedby"
        type="application/pdf"
        href="http://docs.rackspacecloud.com/servers/api/ext/cs-
pie-20111111.pdf"/>

    <atom:link
        rel="describedby"
        type="application/vnd.sun.wadl+xml"
        href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie.wadl"/
>
</extension>
```

Example 3.37. Extension Response: JSON

```
{
    "extension" : {
        "name" : "Public Image Extension",
        "namespace" : "http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0",
        "alias" : "RS-PIE",
        "updated" : "2011-01-22T13:25:27-06:00",
        "description" : "Adds the capability to share an image with other users.",
        "links" : [
            {
                "rel" : "describedby",
                "type" : "application/pdf",
                "href" : "http://docs.rackspacecloud.com/servers/api/ext/cs-
pie-20111111.pdf"
            },
            {
                "rel" : "describedby",
                "type" : "application/vnd.sun.wadl+xml",
                "href" : "http://docs.rackspacecloud.com/servers/api/ext/cs-pie.
wadl"
            }
        ]
    }
}
```

Extensions may define new data types, parameters, actions, headers, states, and resources. In XML, additional elements and attributes may be defined. These elements must be

defined in the extension's namespace. In JSON, the alias must be used. The volumes element in the Examples 3.38 and 3.39 is defined in the RS-CBS namespace. Actions work in exactly the same manner as illustrated in Examples 3.40 and 3.41. Extended headers are always prefixed with X- followed by the alias and a dash: (X-RS-CBS-HEADER1). States and parameters must be prefixed with the extension alias followed by a colon. For example, an image may be in the RS-PIE:PrepareShare state.

Important

Applications should be prepared to ignore response data that contains extension elements. An extended state should always be treated as an UNKNOWN state if the application does not support the extension. Applications should also verify that an extension is available before submitting an extended request.

Example 3.38. Extended Server Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<servers xmlns="http://docs.openstack.org/compute/api/v1.1"
         xmlns:atom="http://www.w3.org/2005/Atom">
  <server id="52415800-8b69-11e0-9b19-734f6af67565"
          tenantId="1234" userId="5678"
          name="sample-server" status="BUILD"
          progress="60" hostId="e4d909c290d0fb1ca068ffaddf22cbd0"
          updated="2010-10-10T12:00:00Z"
          created="2010-08-10T12:00:00Z"
          accessIPv4="67.23.10.132"
          accessIPv6="::babe:67.23.10.132">
    <image id="52415800-8b69-11e0-9b19-734f6f006e54">
        <atom:link
            rel="self"
            href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
        <atom:link
            rel="bookmark"
            href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    </image>
    <flavor id="52415800-8b69-11e0-9b19-734f216543fd">
        <atom:link
            rel="self"
            href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
        <atom:link
            rel="bookmark"
            href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
    </flavor>
    <metadata>
      <meta key="Server Label">Web Head 1</meta>
      <meta key="Image Version">2.1</meta>
    </metadata>
    <addresses>
        <network id="public">
```

```
            <ip version="4" addr="67.23.10.132"/>
            <ip version="6" addr="::babe:67.23.10.132"/>
            <ip version="4" addr="67.23.10.131"/>
            <ip version="6" addr="::babe:4317:0A83"/>
        </network>
        <network id="private">
            <ip version="4" addr="10.176.42.16"/>
            <ip version="6" addr="::babe:10.176.42.16"/>
        </network>
    </addresses>
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f6af67565"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f6af67565"/>
    <volumes
        xmlns="http://docs.rackspacecloud.com/servers/api/ext/cbs/v1.0">
        <volume name="OS"
                href="https://cbs.api.rackspacecloud.com/12934/volumes/
19"/>
        <volume name="Work"
                href="https://cbs.api.rackspacecloud.com/12934/volumes/
23"/>
    </volumes>
  </server>
</servers>
```

Example 3.39. Extended Server Response: JSON

```
{
    "servers": [
        {
            "id": "52415800-8b69-11e0-9b19-734f6af67565",
            "tenantId": "1234",
            "userId": "5678",
            "name": "sample-server",
            "updated": "2010-10-10T12:00:00Z",
            "created": "2010-08-10T12:00:00Z",
            "hostId": "e4d909c290d0fb1ca068ffaddf22cbd0",
            "status": "BUILD",
            "progress": 60,
            "accessIPv4" : "67.23.10.132",
            "accessIPv6" : "::babe:67.23.10.132",
            "image" : {
                "id": "52415800-8b69-11e0-9b19-734f6f006e54",
                "links": [
                    {
                        "rel": "self",
                        "href": "http://servers.api.openstack.org/v1.1/
1234/images/52415800-8b69-11e0-9b19-734f6f006e54"
                    },
                    {
```

```
                        "rel": "bookmark",
                        "href": "http://servers.api.openstack.org/1234/
images/52415800-8b69-11e0-9b19-734f6f006e54"
                    }
                ]
            },
            "flavor" : {
                "id": "52415800-8b69-11e0-9b19-734f216543fd",
                "links": [
                    {
                        "rel": "self",
                        "href": "http://servers.api.openstack.org/v1.1/
1234/flavors/52415800-8b69-11e0-9b19-734f216543fd"
                    },
                    {
                        "rel": "bookmark",
                        "href": "http://servers.api.openstack.org/1234/
flavors/52415800-8b69-11e0-9b19-734f216543fd"
                    }
                ]
            },
            "addresses": {
                "public" : [
                    {
                        "version": 4,
                        "addr": "67.23.10.132"
                    },
                    {
                        "version": 6,
                        "addr": "::babe:67.23.10.132"
                    },
                    {
                        "version": 4,
                        "addr": "67.23.10.131"
                    },
                    {
                        "version": 6,
                        "addr": "::babe:4317:0A83"
                    }
                ],
                "private" : [
                    {
                        "version": 4,
                        "addr": "10.176.42.16"
                    },
                    {
                        "version": 6,
                        "addr": "::babe:10.176.42.16"
                    }
                ]
            },
            "metadata": {
                "Server Label": "Web Head 1",
                "Image Version": "2.1"
            },
            "links": [
                {
```

```
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
servers/52415800-8b69-11e0-9b19-734f6af67565"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/
servers/52415800-8b69-11e0-9b19-734f6af67565"
                }
            ],
            "RS-CBS:volumes": [
                {
                    "name": "OS",
                    "href": "https://cbs.api.rackspacecloud.com/12934/
volumes/19"
                },
                {
                    "name": "Work",
                    "href": "https://cbs.api.rackspacecloud.com/12934/
volumes/23"
                }
            ]
        }
    ]
}
```

Example 3.40. Extended Action: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<attach-volume
    xmlns="http://docs.rackspacecloud.com/servers/api/ext/cbs/v1.0"
    href="https://cbs.api.rackspacecloud.com/12934/volumes/19"/>
```

Example 3.41. Extended Action: JSON

```
{
    "RS-CBS:attach-volume" {
        "href" : "https://cbs.api.rackspacecloud.com/12934/volumes/19"
    }
}
```

3.9. Faults

3.9.1. Synchronous Faults

When an error occurs at request time, the system will return an HTTP error response code denoting the type of error. The system will also return additional information about the fault in the body of the response.

Example 3.42. Fault Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<computeFault
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    code="500">
  <message>Fault!</message>
  <details>Error Details...</details>
</computeFault>
```

Example 3.43. Fault Response: JSON

```
{
    "computeFault" : {
        "code" : 500,
        "message" : "Fault!",
        "details" : "Error Details..."
    }
}
```

The error code is returned in the body of the response for convenience. The message section returns a human-readable message that is appropriate for display to the end user. The details section is optional and may contain information—for example, a stack trace—to assist in tracking down an error. The detail section may or may not be appropriate for display to an end user.

The root element of the fault (e.g. computeFault) may change depending on the type of error. The following is a list of possible elements along with their associated error codes.

Table 3.4. Fault Elements and Error Codes

Fault Element	Associated Error Codes	Expected in All Requests?
computeFault	500, 400, other codes possible	✓
serviceUnavailable	503	✓
unauthorized	401	✓
forbidden	403	✓
badRequest	400	✓
overLimit	413	✓
badMediaType	415	
badMethod	405	
itemNotFound	404	
buildInProgress	409	
serverCapacityUnavailable	503	
backupOrResizeInProgress	409	
resizeNotAllowed	403	
notImplemented	501	

Example 3.44. Fault Response, Item Not Found: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<itemNotFound
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    code="404">
  <message>Not Found</message>
  <details>Error Details...</details>
</itemNotFound>
```

Example 3.45. Fault Response, Item Not Found: JSON

```
{
    "itemNotFound" : {
        "code" : 404,
        "message" : "Not Found",
        "details" : "Error Details..."
    }
}
```

From an XML schema perspective, all API faults are extensions of the base fault type ComputeAPIFault. When working with a system that binds XML to actual classes (such as JAXB), one should be capable of using ComputeAPIFault as a “catch-all” if there's no interest in distinguishing between individual fault types.

The OverLimit fault is generated when a rate limit threshold is exceeded. For convenience, the fault adds a retryAt attribute that contains the content of the Retry-After header in XML Schema 1.0 date/time format.

Example 3.46. Fault Response, Over Limit: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<overLimit
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    code="413"
    retryAt="2010-08-01T00:00:00Z">
  <message>OverLimit Retry...</message>
  <details>Error Details...</details>
</overLimit>
```

Example 3.47. Fault Response, Over Limit: JSON

```
{
    "overLimit" : {
        "code" : 413,
        "message" : "OverLimit Retry...",
        "details" : "Error Details...",
     "retryAt" : "2010-08-01T00:00:00Z"
    }
}
```

3.9.2. Asynchronous Faults

An error may occur in the background while a server or image is being built or while a server is executing an action. In these cases, the server or image is placed in an ERR0R state and the fault is embedded in the offending server or image. Note that these asynchronous faults follow the same format as the synchronous ones. The fault contains an error code, a human readable message, and optional details about the error. Additionally, asynchronous faults may also contain a created timestamp that specify when the fault occured.

Example 3.48. Server In Error Sate: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom"
        id="52415800-8b69-11e0-9b19-734f0000ffff"
        tenantId="1234" userId="5678"
        name="sample-server" status="ERROR"
        created="2010-08-10T12:00:00Z"
        progress="66" hostId="e4d909c290d0fb1ca068ffafff22cbd0">
    <image id="52415800-8b69-11e0-9b19-734f6f007777" />
    <flavor id="52415800-8b69-11e0-9b19-734f216543fd" />
    <fault code="404" created="2010-08-10T11:59:59Z">
        <message>Could not find image 52415800-8b69-11e0-9b19-734f6f007777</
message>
        <details>Fault details</details>
    </fault>
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f000004d2"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f000004d2"/>
</server>
```

Example 3.49. Server In Error Sate: JSON

```
{
    "server": {
        "id": "52415800-8b69-11e0-9b19-734f0000ffff",
        "tenantId": "1234",
        "userId": "5678",
        "name": "sample-server",
        "created": "2010-08-10T12:00:00Z",
        "hostId": "e4d909c290d0fb1ca068ffafff22cbd0",
        "status": "ERROR",
        "progress": 66,
        "image" : {
            "id": "52415800-8b69-11e0-9b19-734f6f007777"
        },
        "flavor" : {
            "id": "52415800-8b69-11e0-9b19-734f216543fd"
        },
        "fault" : {
            "code" : 404,
            "created": "2010-08-10T11:59:59Z",
            "message" : "Could not find image
 52415800-8b69-11e0-9b19-734f6f007777",
            "details" : "Fault details"
        },
        "links": [
            {
                "rel": "self",
                "href": "http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f000004d2"
            },
            {
                "rel": "bookmark",
                "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f000004d2"
            }
        ]
    }
}
```

Example 3.50. Image In Error Sate: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<image
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    xmlns:atom="http://www.w3.org/2005/Atom"
    id="52415800-8b69-11e0-9b19-734f5736d2a2"
    name="My Server Backup"
    created="2010-08-10T12:00:00Z"
    status="ERROR" progress="89">
    <server id="52415800-8b69-11e0-9b19-734f335aa7b3" />
    <fault code="500">
        <message>An internal error occured</message>
        <details>Error details</details>
    </fault>
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
</image>
```

Example 3.51. Image In Error Sate: JSON

```
{
    "image" : {
        "id" : "52415800-8b69-11e0-9b19-734f5736d2a2",
        "name" : "My Server Backup",
        "created" : "2010-08-10T12:00:00Z",
        "status" : "SAVING",
        "progress" : 89,
        "server" : {
            "id": "52415800-8b69-11e0-9b19-734f335aa7b3"
        },
        "fault" : {
            "code" : 500,
            "message" : "An internal error occured",
            "details" : "Error details"
        },
        "links": [
            {
                "rel" : "self",
                "href" : "http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
            },
            {
                "rel" : "bookmark",
                "href" : "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
            }
        ]
    }
}
```

4. API Operations

4.1. Servers

4.1.1. List Servers

Verb	URI	Description
GET	/servers?*image*=imageRef&*flavor*=flavorRef&*name*=serverName&*status*=serverStatus&*marker*=markerID&*limit*=int&*changes-since*=dateTime	List all servers (IDs, names, links)
GET	/servers/detail?*image*=imageRef&*flavor*=flavorRef&*name*=serverName&*status*=serverStatus&*marker*=markerID&*limit*=int&*changes-since*=dateTime	List all servers (all details)

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413)

This operation provides a list of servers associated with your account. Servers that have been deleted are not included in this list. Servers contain a status attribute that can be used as an indication of the current server state. Servers with an ACTIVE status are available for use. Other possible values for the status attribute include: BUILD, REBUILD, SUSPENDED, RESIZE, VERIFY_RESIZE, PASSWORD, REBOOT, HARD_REBOOT, DELETED, UNKNOWN, and ERROR.

The list of servers may be filtered by image, flavor, name, and status via the respective query parameters. Image and flavor references may be IDs or full URLs. When retrieving a list of servers via the changes-since parameter, the list will contain servers that have been deleted since the changes-since time (see Section 3.5, "Efficient Polling with the *Changes-Since* Parameter").

The compute provisioning algorithm has an anti-affinity property that attempts to spread out customer VMs across hosts. Under certain situations, VMs from the same customer may be placed on the same host. hostId represents the host your server runs on and can be used to determine this scenario if it's relevant to your application.

Note

HostId is unique *per account* and is not globally unique.

This operation does not require a request body.

Example 4.1. Servers List Response: XML (detail)

```
<?xml version="1.0" encoding="UTF-8"?>
<servers xmlns="http://docs.openstack.org/compute/api/v1.1"
         xmlns:atom="http://www.w3.org/2005/Atom">
  <server id="52415800-8b69-11e0-9b19-734f6af67565"
          tenantId="1234" userId="5678"
          name="sample-server" status="BUILD"
          progress="60" hostId="e4d909c290d0fb1ca068ffaddf22cbd0"
          updated="2010-10-10T12:00:00Z"
          created="2010-08-10T12:00:00Z"
          accessIPv4="67.23.10.132"
          accessIPv6="::babe:67.23.10.132"
          >
    <image id="52415800-8b69-11e0-9b19-734f6f006e54">
        <atom:link
            rel="self"
            href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
        <atom:link
            rel="bookmark"
            href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    </image>
    <flavor id="52415800-8b69-11e0-9b19-734f216543fd">
        <atom:link
            rel="self"
            href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
        <atom:link
            rel="bookmark"
            href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
    </flavor>
    <metadata>
      <meta key="Server Label">Web Head 1</meta>
      <meta key="Image Version">2.1</meta>
    </metadata>
    <addresses>
        <network id="public">
            <ip version="4" addr="67.23.10.132"/>
            <ip version="6" addr="::babe:67.23.10.132"/>
            <ip version="4" addr="67.23.10.131"/>
            <ip version="6" addr="::babe:4317:0A83"/>
        </network>
        <network id="private">
            <ip version="4" addr="10.176.42.16"/>
            <ip version="6" addr="::babe:10.176.42.16"/>
        </network>
    </addresses>
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f6af67565"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f6af67565"/>
  </server>
  <server id="52415800-8b69-11e0-9b19-734f1f1350e5"
          tenantId="1234" userId="5678"
          name="sample-server2"
          status="ACTIVE" hostId="9e107d9d372bb6826bd81d3542a419d6"
```

```
          updated="2010-10-10T12:00:00Z"
          created="2010-08-10T12:00:00Z"
          accessIPv4="67.23.10.133"
          accessIPv6="::babe:67.23.10.133"
          >
    <image id="52415800-8b69-11e0-9b19-734f5736d2a2">
        <atom:link
            rel="self"
            href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
        <atom:link
            rel="bookmark"
            href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
    </image>
    <flavor id="52415800-8b69-11e0-9b19-734f216543fd">
        <atom:link
            rel="self"
            href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
        <atom:link
            rel="bookmark"
            href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
    </flavor>
    <metadata>
      <meta key="Server Label">DB 1</meta>
    </metadata>
    <addresses>
      <network id="public">
          <ip version="4" addr="67.23.10.133"/>
          <ip version="6" addr="::babe:67.23.10.133"/>
      </network>
      <network id="private">
          <ip version="4" addr="10.176.42.17"/>
          <ip version="6" addr="::babe:10.176.42.17"/>
      </network>
    </addresses>
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f1f1350e5"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f1f1350e5"/>
  </server>

</servers>
```

Example 4.2. Servers List Response: JSON (detail)

```
{
    "servers": [
        {
            "id": "52415800-8b69-11e0-9b19-734f6af67565",
            "tenantId": "1234",
            "userId": "5678",
            "name": "sample-server",
            "updated": "2010-10-10T12:00:00Z",
            "created": "2010-08-10T12:00:00Z",
            "hostId": "e4d909c290d0fb1ca068ffaddf22cbd0",
            "status": "BUILD",
            "progress": 60,
            "accessIPv4" : "67.23.10.132",
            "accessIPv6" : "::babe:67.23.10.132",
            "image" : {
                "id": "52415800-8b69-11e0-9b19-734f6f006e54",
                "links": [
                    {
                        "rel": "self",
                        "href": "http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
                    },
                    {
                        "rel": "bookmark",
                        "href": "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
                    }
                ]
            },
            "flavor" : {
                "id": "52415800-8b69-11e0-9b19-734f216543fd",
                "links": [
                    {
                        "rel": "self",
                        "href": "http://servers.api.openstack.org/v1.1/1234/
flavors/52415800-8b69-11e0-9b19-734f216543fd"
                    },
                    {
                        "rel": "bookmark",
                        "href": "http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"
                    }
                ]
            },
            "addresses": {
                "public" : [
                    {
                        "version": 4,
                        "addr": "67.23.10.132"
                    },
                    {
                        "version": 6,
                        "addr": "::babe:67.23.10.132"
                    },
                    {
                        "version": 4,
                        "addr": "67.23.10.131"
                    },
                    {
                        "version": 6,
                        "addr": "::babe:4317:0A83"
```

```
                    }
                ],
                "private" : [
                    {
                        "version": 4,
                        "addr": "10.176.42.16"
                    },
                    {
                        "version": 6,
                        "addr": "::babe:10.176.42.16"
                    }
                ]
            },
            "metadata": {
                "Server Label": "Web Head 1",
                "Image Version": "2.1"
            },
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f6af67565"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f6af67565"
                }
            ]
        },
        {
            "id": "52415800-8b69-11e0-9b19-734f1f1350e5",
            "userId": "5678",
            "name": "sample-server2",
            "tenantId": "1234",
            "updated": "2010-10-10T12:00:00Z",
            "created": "2010-08-10T12:00:00Z",
            "hostId": "9e107d9d372bb6826bd81d3542a419d6",
            "status": "ACTIVE",
            "accessIPv4" : "67.23.10.133",
            "accessIPv6" : "::babe:67.23.10.133",
            "image" : {
                "id": "52415800-8b69-11e0-9b19-734f5736d2a2",
                "links": [
                    {
                        "rel": "self",
                        "href": "http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
                    },
                    {
                        "rel": "bookmark",
                        "href": "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
                    }
                ]
            },
            "flavor" : {
                "id": "52415800-8b69-11e0-9b19-734f216543fd",
                "links": [
                    {
                        "rel": "self",
                        "href": "http://servers.api.openstack.org/v1.1/1234/
flavors/52415800-8b69-11e0-9b19-734f216543fd"
                    },
```

```
                    {
                        "rel": "bookmark",
                        "href": "http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"
                    }
                ]
            },
            "addresses": {
                "public" : [
                    {
                        "version": 4,
                        "addr": "67.23.10.133"
                    },
                    {
                        "version": 6,
                        "addr": "::babe:67.23.10.133"
                    }
                ],
                "private" : [
                    {
                            "version": 4,
                        "addr": "10.176.42.17"
                    },
                    {
                        "version": 6,
                            "addr": "::babe:10.176.42.17"
                        }
                    ]
            },
            "metadata": {
                "Server Label": "DB 1"
            },
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f1f1350e5"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f1f1350e5"
                }
            ]
        }
    ]
}
```

4.1.2. Create Server

Verb	URI	Description
POST	/servers	Create a new server

Normal Response Code(s): 202

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), serverCapacityUnavailable (503)

Status Transition: `BUILD` → `ACTIVE`
`BUILD` → `ERROR` (on error)

This operation asynchronously provisions a new server. The progress of this operation depends on several factors including location of the requested image, network i/o, host load, and the selected flavor. The progress of the request can be checked by performing a **GET** on /servers/*id*, which will return a progress attribute (0-100% completion). The full URL to the newly created server is returned via the `Location` header and is available as a `self` and `bookmark` link in the server representation (See Section 3.3, "Links and References"). Note that when creating a server only the server ID, its links, and the admin password are guaranteed to be returned in the request. Additional attributes may be retrieved by performing subsequent **GET**s on the server.

Example 4.3. Server Create Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        imageRef="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
        flavorRef="52415800-8b69-11e0-9b19-734f1195ff37"
        name="new-server-test"
        >
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <personality>
    <file path="/etc/banner.txt">
        ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
        dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
        IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
        c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
        QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
        ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
        dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
        c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
        b25zLiINCg0KLVJpY2hhcmQgQmFjaA==
    </file>
  </personality>
</server>
```

Example 4.4. Server Create Request: JSON

```
{
    "server" : {
        "name" : "new-server-test",
        "imageRef" : "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54",
        "flavorRef" : "52415800-8b69-11e0-9b19-734f1195ff37",
        "metadata" : {
            "My Server Name" : "Apache1"
        },
        "personality" : [
            {
                "path" : "/etc/banner.txt",
                "contents" : "ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
 dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
 IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
 c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
 QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
 ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
 dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
 c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
 b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
            }
        ]
    }
}
```

Example 4.5. Server Create Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom"
        id="52415800-8b69-11e0-9b19-734f565bc83b"
        tenantId="1234" userId="5678"
        name="new-server-test"
        hostId="e4d909c290d0fb1ca068ffaddf22cbd0" progress="0"
        status="BUILD" adminPass="GFf1j9aP"
        created="2010-11-11T12:00:00Z"
        accessIPv4="67.23.10.138"
        accessIPv6="::babe:67.23.10.138">
  <image id="52415800-8b69-11e0-9b19-734f6f006e54"
         name="CentOS 5.2">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
  </image>
  <flavor id="52415800-8b69-11e0-9b19-734f1195ff37"
          name="256 MB Server">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
  </flavor>
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <addresses>
    <network id="public">
      <ip version="4" addr="67.23.10.138"/>
      <ip version="6" addr="::babe:67.23.10.138"/>
    </network>
    <network id="private">
      <ip version="4" addr="10.176.42.19"/>
      <ip version="6" addr="::babe:10.176.42.19"/>
    </network>
  </addresses>
  <atom:link
      rel="self"
      href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
  <atom:link
      rel="bookmark"
      href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
</server>
```

Example 4.6. Server Create Response: JSON

```
{
    "server": {
        "id": "52415800-8b69-11e0-9b19-734f565bc83b",
        "tenantId": "1234",
        "userId": "5678",
        "name": "new-server-test",
        "created": "2010-11-11T12:00:00Z",
        "hostId": "e4d909c290d0fb1ca068ffaddf22cbd0",
        "accessIPv4" : "67.23.10.138",
        "accessIPv6" : "::babe:67.23.10.138",
        "progress": 0,
        "status": "BUILD",
        "adminPass": "GFf1j9aP",
        "image" : {
            "id": "52415800-8b69-11e0-9b19-734f6f006e54",
            "name": "CentOS 5.2",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
images/52415800-8b69-11e0-9b19-734f6f006e54"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
                }
            ]
        },
        "flavor" : {
            "id": "52415800-8b69-11e0-9b19-734f1195ff37",
            "name": "256 MB Server",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
flavors/52415800-8b69-11e0-9b19-734f1195ff37"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/
flavors/52415800-8b69-11e0-9b19-734f1195ff37"
                }
            ]
        },
        "metadata": {
            "My Server Name": "Apache1"
        },
        "addresses": {
            "public" : [
                {
                    "version": 4,
                    "addr": "67.23.10.138"
                },
                {
```

```
                    "version": 6,
                    "addr": "::babe:67.23.10.138"
                }
            ],
            "private" : [
                {
                    "version": 4,
                    "addr": "10.176.42.19"
                },
                {
                    "version": 6,
                    "addr": "::babe:10.176.42.19"
                }
            ]
        },
        "links": [
            {
                "rel": "self",
                "href": "http://servers.api.openstack.org/v1.1/1234/
servers/52415800-8b69-11e0-9b19-734fcece0043"
            },
            {
                "rel": "bookmark",
                "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"
            }
        ]
    }
}
```

4.1.2.1. Server Passwords

A password may be specified when creating the server via the optional adminPass attribute. The specified password must meet the complexity requirements set by your OpenStack Compute provider. The server may enter an ERR0R state if the complexity requirements are not met. In this case, an client may issue a change password action to reset the server password (see Section 4.3.1, "Change Password").

If a password is not specified, a randomly generated password will be assigned and returned in the response object. This password is guaranteed to meet the security requirements set by the compute provider. For security reasons, the password will not be returned in subsequent **GET** calls.

4.1.2.2. Server Metadata

Custom server metadata can also be supplied at launch time. See Section 4.6, "Metadata" for details on working with metadata. The maximum size of the metadata key and value is 255 bytes each. The maximum number of key-value pairs that can be supplied per server is determined by the compute provider and may be queried via the maxServerMeta absolute limit.

4.1.2.3. Server Personality

You may further customize a server by injecting data into the file system of the server itself. This is useful, for example, for inserting ssh keys, setting configuration files, or storing data that you want to retrieve from within the instance itself. It is intended to provide a minimal amount of launch-time personalization. If significant customization is required, a custom image should be created. The max size of the file path data is 255 bytes. The max size of the file contents is determined by the compute provider and may vary based on the image that is used to create the server. The absolute limit maxPersonalitySize is a byte limit that is guaranteed to apply to all images in the deployment. Providers may set additional per-image personality limits. Note that file contents should be encoded as a Base64 string and these limits refer to the number of bytes in the decoded data not the number of characters in the encoded data. The maximum number of file path/content pairs that can be supplied is also determined by the compute provider and is denoted by the maxPersonality absolute limit.

Any existing files that match the specified file will be renamed to include the extension bak followed by a time stamp. For example, the file /etc/passwd will be backed up as /etc/passwd.bak.1246036261.5785. Personality files will be accessible only to system administrators. For example, in Linux files will have root and the root group as owner and group owner, respectively and will allow user and group read access only (`-r--r-----`).

4.1.2.4. Server Access Addresses

In a hybrid environment, the IP address of a server may not be controlled by the underling implementation. Instead, the access IP address may be part of the dedicated hardware; for example, a router/NAT device. In this case, the addresses provided by the implementation cannot actually be used to access the server (from outside the local LAN). Here, a separate *access address* may be assigned at creation time to provide access to the server. This address may not be directly bound to a network interface on the server and may

not necessarily appear when a server's addresses are queried (see Section 4.2, “Server Addresses”). Nonetheless, clients which need to access the server directly are encouraged to do so via an access address. In the example below, an IPv4 address is assigned at creation time.

Example 4.7. Creating a Server with a Access IP: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        name="test" accessIPv4="67.23.10.132"
        imageRef="52415800-8b69-11e0-9b19-734f6f006e54"
        flavorRef="52415800-8b69-11e0-9b19-734f1195ff37" />
```

Example 4.8. Creating a Server with a Access IP: JSON

```
{
    "server" : {
        "name" : "new-server-test",
        "imageRef" : "52415800-8b69-11e0-9b19-734f6f006e54",
        "flavorRef" : "52415800-8b69-11e0-9b19-734f1195ff37",
        "accessIPv4" : "67.23.10.132"
    }
}
```

Note that both IPv4 and IPv6 addresses may be used as access addresses and both addresses may be assigned simultaneously as illustrated below. Access addresses may be updated after a server has been created. See Section 4.1.4, “Update Server” for more details.

Example 4.9. Creating a Server with Multiple Access IPs: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        name="test"
        accessIPv4="67.23.10.132"
        accessIPv6="::babe:67.23.10.132"
        imageRef="52415800-8b69-11e0-9b19-734f6f006e54"
        flavorRef="52415800-8b69-11e0-9b19-734f1195ff37" />
```

Example 4.10. Creating a Server with Multiple Access IPs: JSON

```
{
    "server" : {
        "name" : "new-server-test",
        "imageRef" : "52415800-8b69-11e0-9b19-734f6f006e54",
        "flavorRef" : "52415800-8b69-11e0-9b19-734f1195ff37",
        "accessIPv4" : "67.23.10.132",
        "accessIPv6" : "::babe:67.23.10.132"
    }
}
```

4.1.3. Get Server Details

Verb	URI	Description
GET	/servers/*id*	List details of the specified server

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404)

This operation returns the details of a specific server by its ID.

This operation does not require a request body.

Example 4.11. Server Details Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom"
        id="52415800-8b69-11e0-9b19-734f000004d2"
        tenantId="1234" userId="5678"
        name="sample-server" status="BUILD"
        updated="2010-10-10T12:00:00Z" created="2010-08-10T12:00:00Z"
        progress="60" hostId="e4d909c290d0fb1ca068ffaddf22cbd0"
        accessIPv4="67.23.10.132"
        accessIPv6="::babe:67.23.10.132">
    <image id="52415800-8b69-11e0-9b19-734f6f006e54">
        <atom:link
            rel="self"
            href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
        <atom:link
            rel="bookmark"
            href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    </image>
    <flavor id="52415800-8b69-11e0-9b19-734f216543fd">
        <atom:link
            rel="self"
            href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
        <atom:link
            rel="bookmark"
            href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
    </flavor>
    <metadata>
        <meta key="Server Label">Web Head 1</meta>
        <meta key="Image Version">2.1</meta>
    </metadata>
    <addresses>
        <network id="public">
            <ip version="4" addr="67.23.10.132"/>
            <ip version="6" addr="::babe:67.23.10.132"/>
            <ip version="4" addr="67.23.10.131"/>
            <ip version="6" addr="::babe:4317:0A83"/>
        </network>
        <network id="private">
            <ip version="4" addr="10.176.42.16"/>
            <ip version="6" addr="::babe:10.176.42.16"/>
        </network>
    </addresses>
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f000004d2"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f000004d2"/>
</server>
```

Example 4.12. Server Details Response: JSON

```
{
    "server": {
        "id": "52415800-8b69-11e0-9b19-734f000004d2",
        "tenantId": "1234",
        "userId": "5678",
        "name": "sample-server",
        "updated": "2010-10-10T12:00:00Z",
        "created": "2010-08-10T12:00:00Z",
        "hostId": "e4d909c290d0fb1ca068ffaddf22cbd0",
        "accessIPv4" : "67.23.10.132",
        "accessIPv6" : "::babe:67.23.10.132",
        "status": "BUILD",
        "progress": 60,
        "image" : {
            "id": "52415800-8b69-11e0-9b19-734f6f006e54",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
images/52415800-8b69-11e0-9b19-734f6f006e54"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
                }
            ]
        },
        "flavor" : {
            "id": "52415800-8b69-11e0-9b19-734f216543fd",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
flavors/52415800-8b69-11e0-9b19-734f216543fd"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/
flavors/52415800-8b69-11e0-9b19-734f216543fd"
                }
            ]
        },
        "addresses":  {
            "public" : [
                {
                    "version": 4,
                    "addr": "67.23.10.132"
                },
                {
                    "version": 6,
                    "addr": "::babe:67.23.10.132"
                },
                {
                    "version": 4,
```

```
                    "addr": "67.23.10.131"
                },
                {
                    "version": 6,
                    "addr": "::babe:4317:0A83"
                }
            ],
            "private" : [
                {
                    "version": 4,
                    "addr": "10.176.42.16"
                },
                {
                    "version": 6,
                    "addr": "::babe:10.176.42.16"
                }
            ]
        },
        "metadata": {
            "Server Label": "Web Head 1",
            "Image Version": "2.1"
        },
        "links": [
            {
                "rel": "self",
                "href": "http://servers.api.openstack.org/v1.1/1234/
servers/52415800-8b69-11e0-9b19-734f000004d2"
            },
            {
                "rel": "bookmark",
                "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f000004d2"
            }
        ]
    }
}
```

4.1.4. Update Server

Verb	URI	Description
PUT	/servers/*id*	Update the specified server's editable attributes

Normal Response Code(s): 200

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), buildInProgress (409)

Status Transition: ACTIVE → ACTIVE

This operation updates the editable attributes of a server: the name of the server and the IPv4 and IPv6 access addresses. Note that while the server name is editable, the operation does not change the server host name. Note also that server names are not guaranteed to be unique.

Example 4.13. Server Update Name Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    name="new-server-test"/>
```

Example 4.14. Server Update Name Request: JSON

```
{
  "server" :
    {
        "name" : "new-server-test"
    }
}
```

Example 4.15. Server Update Name Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom"
        id="52415800-8b69-11e0-9b19-734f565bc83b"
        tenantId="1234" userId="5678"
        name="new-server-test"
        hostId="e4d909c290d0fb1ca068ffaddf22cbd0" progress="0"
        status="ACTIVE"
        created="2010-11-11T12:00:00Z"
        updated="2010-11-12T12:44:44Z"
        accessIPv4="67.23.10.138"
        accessIPv6="::babe:67.23.10.138">
  <image id="52415800-8b69-11e0-9b19-734f6f006e54"
         name="CentOS 5.2">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
  </image>
  <flavor id="52415800-8b69-11e0-9b19-734f1195ff37"
          name="256 MB Server">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
  </flavor>
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <addresses>
    <network id="public">
      <ip version="4" addr="67.23.10.138"/>
      <ip version="6" addr="::babe:67.23.10.138"/>
    </network>
    <network id="private">
      <ip version="4" addr="10.176.42.19"/>
      <ip version="6" addr="::babe:10.176.42.19"/>
    </network>
  </addresses>
  <atom:link
      rel="self"
      href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
  <atom:link
      rel="bookmark"
      href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
</server>
```

Example 4.16. Server Update Name Response: JSON

```
{
    "server": {
        "id": "52415800-8b69-11e0-9b19-734f565bc83b",
        "tenantId": "1234",
        "userId": "5678",
        "name": "new-server-test",
        "created": "2010-11-11T12:00:00Z",
        "updated": "2010-11-12T12:44:44Z",
        "hostId": "e4d909c290d0fb1ca068ffaddf22cbd0",
        "accessIPv4" : "67.23.10.138",
        "accessIPv6" : "::babe:67.23.10.138",
        "progress": 0,
        "status": "ACTIVE",
        "image" : {
            "id": "52415800-8b69-11e0-9b19-734f6f006e54",
            "name": "CentOS 5.2",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
images/52415800-8b69-11e0-9b19-734f6f006e54"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
                }
            ]
        },
        "flavor" : {
            "id": "52415800-8b69-11e0-9b19-734f1195ff37",
            "name": "256 MB Server",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
flavors/52415800-8b69-11e0-9b19-734f1195ff37"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"
                }
            ]
        },
        "metadata": {
            "My Server Name": "Apache1"
        },
        "addresses": {
            "public" : [
                {
                    "version": 4,
                    "addr": "67.23.10.138"
                },
                {
                    "version": 6,
                    "addr": "::babe:67.23.10.138"
```

```
                }
            ],
            "private" : [
                {
                    "version": 4,
                    "addr": "10.176.42.19"
                },
                {
                    "version": 6,
                    "addr": "::babe:10.176.42.19"
                }
            ]
        },
        "links": [
            {
                "rel": "self",
                "href": "http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"
            },
            {
                "rel": "bookmark",
                "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"
            }
        ]
    }
}
```

Access addresses may simultaneously be updated as illustrated below.

Example 4.17. Server Update Access Address Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    accessIPv4="67.23.10.132"
    accessIPv6="::babe:67.23.10.132"
/>
```

Example 4.18. Server Update Access Address Request: JSON

```
{
  "server" :
    {
        "accessIPv4" : "67.23.10.132",
        "accessIPv6" : "::babe:67.23.10.132"
    }
}
```

Example 4.19. Server Update Access Address Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom"
        id="52415800-8b69-11e0-9b19-734f565bc83b"
        tenantId="1234" userId="5678"
        name="new-server-test"
        hostId="e4d909c290d0fb1ca068ffaddf22cbd0" progress="0"
        status="ACTIVE"
        created="2010-11-11T12:00:00Z"
        updated="2010-11-12T12:55:55Z"
        accessIPv4="67.23.10.132"
        accessIPv6="::babe:67.23.10.132">
  <image id="52415800-8b69-11e0-9b19-734f6f006e54"
         name="CentOS 5.2">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
  </image>
  <flavor id="52415800-8b69-11e0-9b19-734f1195ff37"
          name="256 MB Server">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
  </flavor>
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <addresses>
    <network id="public">
      <ip version="4" addr="67.23.10.138"/>
      <ip version="6" addr="::babe:67.23.10.138"/>
    </network>
    <network id="private">
      <ip version="4" addr="10.176.42.19"/>
      <ip version="6" addr="::babe:10.176.42.19"/>
    </network>
  </addresses>
  <atom:link
      rel="self"
      href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
  <atom:link
      rel="bookmark"
      href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
</server>
```

Example 4.20. Server Update Access Address Response: JSON

```
{
    "server": {
        "id": "52415800-8b69-11e0-9b19-734f565bc83b",
        "tenantId": "1234",
        "userId": "5678",
        "name": "new-server-test",
        "created": "2010-11-11T12:00:00Z",
        "updated": "2010-11-12T12:55:55Z",
        "hostId": "e4d909c290d0fb1ca068ffaddf22cbd0",
        "accessIPv4" : "67.23.10.132",
        "accessIPv6" : "::babe:67.23.10.132",
        "progress": 0,
        "status": "ACTIVE",
        "image" : {
            "id": "52415800-8b69-11e0-9b19-734f6f006e54",
            "name": "CentOS 5.2",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
images/52415800-8b69-11e0-9b19-734f6f006e54"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
                }
            ]
        },
        "flavor" : {
            "id": "52415800-8b69-11e0-9b19-734f1195ff37",
            "name": "256 MB Server",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
flavors/52415800-8b69-11e0-9b19-734f1195ff37"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"
                }
            ]
        },
        "metadata": {
            "My Server Name": "Apache1"
        },
        "addresses": {
            "public" : [
                {
                    "version": 4,
                    "addr": "67.23.10.138"
                },
                {
                    "version": 6,
                    "addr": "::babe:67.23.10.138"
```

```
                }
            ],
            "private" : [
                {
                    "version": 4,
                    "addr": "10.176.42.19"
                },
                {
                    "version": 6,
                    "addr": "::babe:10.176.42.19"
                }
            ]
        },
        "links": [
            {
                "rel": "self",
                "href": "http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"
            },
            {
                "rel": "bookmark",
                "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"
            }
        ]
    }
}
```

4.1.5. Delete Server

Verb	URI	Description
DELETE	/servers/*id*	Terminate the specified server

Normal Response Code(s): 204

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), buildInProgress (409)

Status Transition:	ACTIVE → DELETED
	ERROR → DELETED

This operation deletes a cloud server instance from the system.

Note

When a server is deleted, all images created from that server are also removed.

This operation does not require a request or a response body.

4.2. Server Addresses

4.2.1. List Addresses

Verb	URI	Description
GET	/servers/*id*/ips	List all server addresses

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), buildInProgress (409)

This operation does not require a request body.

Example 4.21. Addresses List Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<addresses xmlns="http://docs.openstack.org/compute/api/v1.1">
    <network id="public">
        <ip version="4" addr="67.23.10.132"/>
        <ip version="6" addr="::babe:67.23.10.132"/>
        <ip version="4" addr="67.23.10.131"/>
        <ip version="6" addr="::babe:4317:0A83"/>
    </network>
    <network id="private">
        <ip version="4" addr="10.176.42.16"/>
        <ip version="6" addr="::babe:10.176.42.16"/>
    </network>
</addresses>
```

Example 4.22. Addresses List Response: JSON

```
{
  "addresses": {
      "public" : [
          {
              "version": 4,
              "addr": "67.23.10.132"
          },
          {
              "version": 6,
              "addr": "::babe:67.23.10.132"
          },
          {
              "version": 4,
              "addr": "67.23.10.131"
          },
          {
              "version": 6,
              "addr": "::babe:4317:0A83"
          }
      ],
      "private" : [
          {
              "version": 4,
              "addr": "10.176.42.16"
          },
          {
              "version": 6,
              "addr": "::babe:10.176.42.16"
          }
      ]
  }
}
```

4.2.2. List Addresses by Network

Verb	URI	Description
GET	/servers/*id*/ips/*networkID*	List addresses by Network ID

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), buildInProgress (409)

This operation does not require a request body.

Example 4.23. List Addresses by Network: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns="http://docs.openstack.org/compute/api/v1.1"
         id="public">
    <ip version="4" addr="67.23.10.132"/>
    <ip version="6" addr="::babe:67.23.10.132"/>
    <ip version="4" addr="67.23.10.131"/>
    <ip version="6" addr="::babe:4317:0A83"/>
</network>
```

Example 4.24. List Addresses by Network: JSON

```
{
    "network" {
        "id" : "public",
        "ip" : [
            {"version" : 4, "addr" : "67.23.10.132"},
            {"version" : 6, "addr" : "::babe:67.23.10.132"},
            {"version" : 4, "addr" : "67.23.10.131"},
            {"version" : 6, "addr" : "::babe:4317:0A83"}
        ]
    }
}
```

4.3. Server Actions

4.3.1. Change Password

Verb	URI	Description
POST	/servers/*id*/action	Change a server's password

Normal Response Code(s): 202

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), buildInProgress (409)

Status Transition:

ACTIVE → PASSWORD → ACTIVE

ACTIVE → PASSWORD → ERROR (on error)

ERROR → PASSWORD → ACTIVE (password reset after error)

This operation changes the server's administrator password. The specified password must meet the complexity requirements set by your OpenStack Compute provider. The server may enter an ERROR state if the complexity requirements are not met. In this case, a client may reissue the change password action.

Example 4.25. Server Update Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<changePassword
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    adminPass="ss1293837$%^"/>
```

Example 4.26. Server Update Request: JSON

```
{
    "changePassword" : {
        "adminPass" : "ss1293837$%^"
    }
}
```

This operation does not contain a response body.

4.3.2. Reboot Server

Verb	URI	Description
POST	/servers/*id*/action	Reboot the specified server

Normal Response Code(s): 202

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), buildInProgress (409)

Status Transition:	ACTIVE → REBOOT → ACTIVE (soft reboot)
	ACTIVE → HARD_REBOOT → ACTIVE (hard reboot)

The reboot function allows for either a soft or hard reboot of a server. With a soft reboot (SOFT), the operating system is signaled to restart, which allows for a graceful shutdown of all processes. A hard reboot (HARD) is the equivalent of power cycling the server.

Example 4.27. Action Reboot: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<reboot
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    type="HARD"/>
```

Example 4.28. Action Reboot: JSON

```
{
    "reboot" : {
        "type" : "HARD"
    }
}
```

This operation does not return a response body.

4.3.3. Rebuild Server

Verb	URI	Description
POST	/servers/*id*/action	Rebuild the specified server

Normal Response Code(s): 202

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), serverCapacityUnavailable (503), buildInProgress (409)

Status Transition: ACTIVE → REBUILD → ACTIVE
ACTIVE → REBUILD → ERROR (on error)

The rebuild function removes all data on the server and replaces it with the specified image. The serverRef and all IP addresses will remain the same. If name and metadata are specified, they will replace existing values, otherwise they will not change. A rebuild operation always removes data injected into the file system via server personality. You may reinsert data into the filesystem during the rebuild. The full URL to the rebuild server is returned via the `Location` header.

Example 4.29. Action Rebuild Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rebuild
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    name="newName"
    imageRef="https://servers.api.rackspacecloud.com/v1.1/32278/images/
52415800-8b69-11e0-9b19-734f6f006e54"
    adminPass="GFf1j9aP">
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <personality>
    <file path="/etc/banner.txt">
        ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
        dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
        IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
        c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
        QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
        ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
        dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
        c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
        b25zLiINCg0KLVJpY2hhcmQgQmFjaA==
    </file>
  </personality>
</rebuild>
```

Example 4.30. Action Rebuild Request: JSON

```
{
    "rebuild" : {
        "imageRef" : "https://servers.api.rackspacecloud.com/v1.1/32278/
images/52415800-8b69-11e0-9b19-734f6f006e54",
        "name" : "newName",
        "adminPass" : "GFf1j9aP",
        "metadata" : {
            "My Server Name" : "Apache1"
        },
        "personality" : [
            {
                "path" : "/etc/banner.txt",
                "contents" : "ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
 dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
 IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
 c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
 QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
 ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
 dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
 c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
 b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
            }
        ]
    }
}
```

Example 4.31. Action Rebuild Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom"
        id="52415800-8b69-11e0-9b19-734f565bc83b"
        tenantId="1234" userId="5678"
        name="newName"
        hostId="e4d909c290d0fb1ca068ffaddf22cbd0" progress="0"
        status="REBUILD" adminPass="GFf1j9aP"
        created="2010-11-11T12:00:00Z"
        accessIPv4="67.23.10.138"
        accessIPv6="::babe:67.23.10.138">
  <image id="52415800-8b69-11e0-9b19-734f6f006e54"
         name="CentOS 5.2">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
  </image>
  <flavor id="52415800-8b69-11e0-9b19-734f1195ff37"
          name="256 MB Server">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
  </flavor>
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <addresses>
    <network id="public">
      <ip version="4" addr="67.23.10.138"/>
      <ip version="6" addr="::babe:67.23.10.138"/>
    </network>
    <network id="private">
      <ip version="4" addr="10.176.42.19"/>
      <ip version="6" addr="::babe:10.176.42.19"/>
    </network>
  </addresses>
  <atom:link
      rel="self"
      href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
  <atom:link
      rel="bookmark"
      href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"/>
</server>
```

Example 4.32. Action Rebuild Response: JSON

```
{
    "server": {
        "id": "52415800-8b69-11e0-9b19-734f565bc83b",
        "tenantId": "1234",
        "userId": "5678",
        "name": "newName",
        "created": "2010-11-11T12:00:00Z",
        "hostId": "e4d909c290d0fb1ca068ffaddf22cbd0",
        "accessIPv4" : "67.23.10.138",
        "accessIPv6" : "::babe:67.23.10.138",
        "progress": 0,
        "status": "REBUILD",
        "adminPass": "GFf1j9aP",
        "image" : {
            "id": "52415800-8b69-11e0-9b19-734f6f006e54",
            "name": "CentOS 5.2",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
images/52415800-8b69-11e0-9b19-734f6f006e54"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
                }
            ]
        },
        "flavor" : {
            "id": "52415800-8b69-11e0-9b19-734f1195ff37",
            "name": "256 MB Server",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
flavors/52415800-8b69-11e0-9b19-734f1195ff37"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/
flavors/52415800-8b69-11e0-9b19-734f1195ff37"
                }
            ]
        },
        "metadata": {
            "My Server Name": "Apache1"
        },
        "addresses": {
            "public" : [
                {
                    "version": 4,
                    "addr": "67.23.10.138"
                },
                {
```

```
                    "version": 6,
                    "addr": "::babe:67.23.10.138"
                }
            ],
            "private" : [
                {
                    "version": 4,
                    "addr": "10.176.42.19"
                },
                {
                    "version": 6,
                    "addr": "::babe:10.176.42.19"
                }
            ]
        },
        "links": [
            {
                "rel": "self",
                "href": "http://servers.api.openstack.org/v1.1/1234/
servers/52415800-8b69-11e0-9b19-734fcece0043"
            },
            {
                "rel": "bookmark",
                "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcece0043"
            }
        ]
    }
}
```

4.3.4. Resize Server

Verb	URI	Description
POST	/servers/*id*/action	Resize the specified server

Normal Response Code(s): 202

Error Response Code(s): computeFault (400, 500, ...), resizeNotAllowed (403), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), serverCapacityUnavailable (503), buildInProgress (409)

Status Transition: ACTIVE → RESIZE → VERIFY_RESIZE
ACTIVE → RESIZE → ACTIVE (on error)

The resize function converts an existing server to a different flavor, in essence, scaling the server up or down. The original server is saved for a period of time to allow rollback if there is a problem. All resizes should be tested and explicitly confirmed, at which time the original server is removed. All resizes are automatically confirmed after 24 hours if they are not explicitly confirmed or reverted.

Example 4.33. Action Resize: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<resize
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    flavorRef="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
```

Example 4.34. Action Resize: JSON

```
{
    "resize" : {
        "flavorRef" : "http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"
    }
}
```

This operation does not return a response body.

4.3.5. Confirm Resized Server

Verb	URI	Description
POST	/servers/*id*/action	Confirm a pending resize action

Normal Response Code(s): 204

Error Response Code(s): computeFault (400, 500, ...), resizeNotAllowed (403), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), serverCapacityUnavailable (503), buildInProgress (409)

Status Transition: VERIFY_RESIZE → ACTIVE
VERIFY_RESIZE → ERROR (on error)

During a resize operation, the original server is saved for a period of time to allow roll back if there is a problem. Once the newly resized server is tested and has been confirmed to be functioning properly, use this operation to confirm the resize. After confirmation, the original server is removed and cannot be rolled back to. All resizes are automatically confirmed after 24 hours if they are not explicitly confirmed or reverted.

Example 4.35. Action Confirm Resize: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<confirmResize
    xmlns="http://docs.openstack.org/compute/api/v1.1"/>
```

Example 4.36. Action Confirm Resize: JSON

```
{
    "confirmResize" : null
}
```

This operation does not return a response body.

4.3.6. Revert Resized Server

Verb	URI	Description
POST	/servers/*id*/action	Cancel and revert a pending resize action

Normal Response Code(s): 202

Error Response Code(s): computeFault (400, 500, ...), resizeNotAllowed (403), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), serverCapacityUnavailable (503), buildInProgress (409)

Status Transition: VERIFY_RESIZE → ACTIVE
VERIFY_RESIZE → ERROR (on error)

During a resize operation, the original server is saved for a period of time to allow for roll back if there is a problem. If you determine there is a problem with a newly resized server, use this operation to revert the resize and roll back to the original server. All resizes are automatically confirmed after 24 hours if they have not already been confirmed explicitly or reverted.

Example 4.37. Action Revert Resize: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<revertResize
    xmlns="http://docs.openstack.org/compute/api/v1.1"/>
```

Example 4.38. Action Revert Resize: JSON

```
{
    "revertResize" : null
}
```

This operation does not return a response body.

4.3.7. Create Image

Verb	URI	Description
POST	/servers/*id*/action	Creates a new image.

Normal Response Code(s): 202

Error Response Code(s): computeFault (400, 500, ...), backupOrResizeInProgress (409), resizeNotAllowed (403), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), serverCapacityUnavailable (503), buildInProgress (409)

Image Status Transition:	`SAVING` → `ACTIVE`
	`SAVING` → `ERROR` (on error)

This action creates a new image for the given server. Once complete, a new image will be available that can be used to rebuild or create servers. The full URL to the newly created image is returned via the `Location` header, additional attributes for the image including creation status may be retrieved by performing a subsequent **GET** on that URL. See Section 4.5.2, "Get Image Details" for details.

Custom image metadata can also be supplied when creating an image. See Section 4.6, "Metadata" for details on working with metadata. The maximum size of the metadata key and value is 255 bytes each. The maximum number of key-value pairs that can be supplied per image is determined by the compute provider and may be queried via the maxImageMeta absolute limit.

This operation does not return a response body.

Note

At present, image creation is an asynchronous operation, so coordinating the creation with data quiescence, etc. is currently not possible.

Example 4.39. Action Create Image: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<createImage
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    name="new-image">
    <metadata>
        <meta key="ImageType">Gold</meta>
        <meta key="ImageVersion">2.0</meta>
    </metadata>
</createImage>
```

Example 4.40. Action Create Image: JSON

```
{
    "createImage" : {
        "name" : "new-image",
        "metadata": {
            "ImageType": "Gold",
            "ImageVersion": "2.0"
        }
    }
}
```

4.4. Flavors

A flavor is an available hardware configuration for a server. Each flavor has a unique combination of disk space and memory capacity.

4.4.1. List Flavors

Verb	URI	Description
GET	/flavors?*minDisk*=minDiskInGB&*minRam*=minRamInMB&*marker*=markerID&*limit*=int	List available flavors (IDs, names, links)
GET	/flavors/detail?*minDisk*=minDiskInGB&*minRam*=minRamInMB&*marker*=markerID&*limit*=int	List available flavors (all details)

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413)

This operation will list all available flavors. The *minDisk* parameter can narrow the list of flavors to those which contain at least the specified number of gigabytes of disk storage. The *minRam* parameter narrows the list of flavors to those that contain at least the specified amount of RAM in megabytes.

This operation does not require a request body.

Example 4.41. Flavors List Response: XML (detail)

```
<?xml version="1.0" encoding="UTF-8"?>
<flavors xmlns="http://docs.openstack.org/compute/api/v1.1"
         xmlns:atom="http://www.w3.org/2005/Atom">
  <flavor id="52415800-8b69-11e0-9b19-734f1195ff37"
          name="256 MB Server" ram="256" disk="10" vcpus="1">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
  </flavor>
  <flavor id="52415800-8b69-11e0-9b19-734f216543fd"
          name="512 MB Server" ram="512" disk="20" vcpus="2">
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f216543fd"/>
  </flavor>
</flavors>
```

Example 4.42. Flavors List Response: JSON (detail)

```
{
    "flavors": [
        {
            "id": "52415800-8b69-11e0-9b19-734f1195ff37",
            "name": "256 MB Server",
            "ram": 256,
            "disk": 10,
            "vcpus": 1,
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
flavors/52415800-8b69-11e0-9b19-734f1195ff37"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/
flavors/52415800-8b69-11e0-9b19-734f1195ff37"
                }
            ]
        },
        {
            "id": "52415800-8b69-11e0-9b19-734f216543fd",
            "name": "512 MB Server",
            "ram": 512,
            "disk": 20,
            "vcpus": 2,
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
flavors/52415800-8b69-11e0-9b19-734f216543fd"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/
flavors/52415800-8b69-11e0-9b19-734f216543fd"
                }
            ]
        }
    ]
}
```

4.4.2. Get Flavor Details

Verb	URI	Description
GET	/flavors/*id*	List details of the specified flavor

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404)

This operation returns details of the specified flavor.

This operation does not require a request body.

Example 4.43. Flavor Details Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<flavor
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    xmlns:atom="http://www.w3.org/2005/Atom"
    id="52415800-8b69-11e0-9b19-734f1195ff37"
    name="256 MB Server" ram="256" disk="10" vcpus="1">
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"/>
</flavor>
```

Example 4.44. Flavor Details Response: JSON

```
{
    "flavor" : {
        "id" : "52415800-8b69-11e0-9b19-734f1195ff37",
        "name" : "256 MB Server",
        "ram" : 256,
        "disk" : 10,
        "vcpus" : 1,
        "links": [
            {
                "rel" : "self",
                "href" : "http://servers.api.openstack.org/v1.1/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"
            },
            {
                "rel" : "bookmark",
                "href" : "http://servers.api.openstack.org/1234/flavors/
52415800-8b69-11e0-9b19-734f1195ff37"
            }
        ]
    }
}
```

4.5. Images

An image is a collection of files you use to create or rebuild a server. Operators provide pre-built OS images by default. You may also create custom images.

4.5.1. List Images

Verb	URI	Description
GET	/images?*server*=serverRef&*name*=imageName&*status*=imageStatus&*changes-since*=dateTime&*marker*=markerID&*limit*=int&*type*=(BASE|SERVER)	List available images (IDs, names, links)
GET	/images/detail?*server*=serverRef&*name*=imageName&*status*=imageStatus&*changes-since*=dateTime&*marker*=markerID&*limit*=int&*type*=(BASE|SERVER)	List available images (all details)

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413)

This operation will list all images visible by the account.

In-flight images will have the status attribute set to SAVING and the conditional progress element (0-100% completion) will also be returned. Other possible values for the status attribute include: UNKNOWN, ACTIVE, SAVING, ERROR, and DELETED. Images with an ACTIVE status are available for install. The optional minDisk and minRam attributes set the minimum disk and RAM requirements needed to create a server with the image.

The list of images may be filtered by server, name, and status via the respective query parameters. A server reference may be specified by ID or with a full URL. The *type* parameter will select only base images (BASE) or server backups (SERVER). When using the changes-since parameter the list of images will contain images that have changed since the changes-since time. See Section 3.5, "Efficient Polling with the *Changes-Since* Parameter" for details.

This operation does not require a request body.

Example 4.45. Images List Response: XML (detail)

```
<?xml version="1.0" encoding="UTF-8"?>
<images xmlns="http://docs.openstack.org/compute/api/v1.1"
        xmlns:atom="http://www.w3.org/2005/Atom">
  <image id="52415800-8b69-11e0-9b19-734f6f006e54"
         name="CentOS 5.2"
         updated="2010-10-10T12:00:00Z"
         created="2010-08-10T12:00:00Z"
         tenantId="12345"
         userId="joe"
         status="ACTIVE">
    <metadata>
        <meta key="ImageType">Gold</meta>
        <meta key="ImageVersion">1.5</meta>
    </metadata>
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
  </image>
  <image
      id="52415800-8b69-11e0-9b19-734f5736d2a2"
      name="My Server Backup"
      updated="2010-10-10T12:00:00Z"
      created="2010-08-10T12:00:00Z"
      tenantId="12345"
      userId="joe"
      status="SAVING" progress="80"
      minDisk="5" minRam="256">
      <server id="52415800-8b69-11e0-9b19-734f335aa7b3">
          <atom:link
              rel="self"
              href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f335aa7b3"/>
          <atom:link
              rel="bookmark"
              href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f335aa7b3"/>
      </server>
      <atom:link
          rel="self"
          href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
      <atom:link
          rel="bookmark"
          href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
  </image>
</images>
```

Example 4.46. Images List Response: JSON (detail)

```
{
    "images": [
        {
            "id": "52415800-8b69-11e0-9b19-734f6f006e54",
            "name": "CentOS 5.2",
            "updated": "2010-10-10T12:00:00Z",
            "created": "2010-08-10T12:00:00Z",
            "tenantId" : "12345",
            "userId" : "joe",
            "status": "ACTIVE",
            "metadata": {
                "ImageType": "Gold",
                "ImageVersion": "1.5"
            },
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
                }
            ]
        },
        {
            "id" : "52415800-8b69-11e0-9b19-734f5736d2a2",
            "name" : "My Server Backup",
            "updated" : "2010-10-10T12:00:00Z",
            "created" : "2010-08-10T12:00:00Z",
            "tenantId" : "12345",
            "userId" : "joe",
            "status" : "SAVING",
            "progress" : 80,
            "minDisk" : 5,
            "minRam" : 256,
            "server" : {
                "id": "52415800-8b69-11e0-9b19-734f335aa7b3",
                "links": [
                    {
                        "rel": "self",
                        "href": "http://servers.api.openstack.org/v1.1/1234/
servers/52415800-8b69-11e0-9b19-734f335aa7b3"
                    },
                    {
                        "rel": "bookmark",
                        "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f335aa7b3"
                    }
                ]
            },
            "links": [
                {
                    "rel" : "self",
                    "href" : "http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
                },
                {
                    "rel" : "bookmark",
```

```
                    "href" : "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
                }
            ]}]}
```

4.5.2. Get Image Details

Verb	URI	Description
GET	/images/*id*	List details of the specified image

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404)

This operation returns details of the specified image.

This operation does not require a request body.

Example 4.47. Image Details Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<image
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    xmlns:atom="http://www.w3.org/2005/Atom"
    id="52415800-8b69-11e0-9b19-734f5736d2a2"
    name="My Server Backup"
    updated="2010-10-10T12:00:00Z"
    created="2010-08-10T12:00:00Z"
    tenantId="12345"
    userId="joe"
    status="SAVING" progress="80"
    minDisk="5" minRam="256">
    <server id="52415800-8b69-11e0-9b19-734f335aa7b3">
        <atom:link
            rel="self"
            href="http://servers.api.openstack.org/v1.1/1234/servers/
52415800-8b69-11e0-9b19-734f335aa7b3"/>
        <atom:link
            rel="bookmark"
            href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f335aa7b3"/>
    </server>
    <atom:link
        rel="self"
        href="http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
    <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
</image>
```

Example 4.48. Image Details Response: JSON

```
{
    "image" : {
        "id" : "52415800-8b69-11e0-9b19-734f5736d2a2",
        "name" : "My Server Backup",
        "updated" : "2010-10-10T12:00:00Z",
        "created" : "2010-08-10T12:00:00Z",
        "tenantId" : "12345",
        "userId" : "joe",
        "status" : "SAVING",
        "progress" : 80,
        "minDisk" : 5,
        "minRam" : 256,
        "server" : {
            "id": "52415800-8b69-11e0-9b19-734f335aa7b3",
            "links": [
                {
                    "rel": "self",
                    "href": "http://servers.api.openstack.org/v1.1/1234/
servers/52415800-8b69-11e0-9b19-734f335aa7b3"
                },
                {
                    "rel": "bookmark",
                    "href": "http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f335aa7b3"
                }
            ]
        },
        "links": [
            {
                "rel" : "self",
                "href" : "http://servers.api.openstack.org/v1.1/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
            },
            {
                "rel" : "bookmark",
                "href" : "http://servers.api.openstack.org/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"
            }
        ]
    }
}
```

4.5.3. Delete Image

Verb	URI	Description
DELETE	/images/*id*	Deletes the specified image.

Normal Response Code(s): 204

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404)

Status Transition:	ACTIVE → DELETED
	ERROR → DELETED

This operation deletes an image from the system.

Images are immediately removed. Currently, there are no state transitions to track the delete operation.

This operation does not require a request body.

This operation does not contain a response body.

4.6. Metadata

The following operations allow access to metadata after a server or image has been created.

4.6.1. List Metadata

Verb	URI	Description
GET	/servers/*id*/metadata	List metadata associated with a server
GET	/images/*id*/metadata	List metadata associated with an image

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404)

Lists all metadata. This operation does not require a request body.

Example 4.49. Metadata List Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>

<metadata xmlns="http://docs.openstack.org/compute/api/v1.1">
    <meta key="Server Label">Web Head 1</meta>
    <meta key="Image Version">2.1</meta>
</metadata>
```

Example 4.50. Metadata List Response: JSON

```
{
    "metadata" :  {
            "Server Label" : "Web Head 1",
            "Image Version" : "2.1"
        }
}
```

4.6.2. Set Metadata

Verb	URI	Description
PUT	/servers/*id*/metadata	Set metadata
PUT	/images/*id*/metadata	Set metadata

Normal Response Code(s): 200

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), buildInProgress (409)

Sets metadata for the resource. Existing metadata items are replaced with the ones provided in the request. An overLimit (413) fault may be thrown if the maximum number of metadata items is exceeded. The maximum number of key-value pairs that can be supplied per server is determined by the compute provider and may be queried via the maxServerMeta absolute limit. The maximum number of key-value pairs for an image may be queried via the maxImageMeta absolute limit.

Example 4.51. Metadata Reset Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>

<metadata xmlns="http://docs.openstack.org/compute/api/v1.1">
    <meta key="Server Label">Web Head 1</meta>
    <meta key="Image Version">2.1</meta>
</metadata>
```

Example 4.52. Metadata Reset Request: JSON

```
{
    "metadata" :  {
            "Server Label" : "Web Head 1",
            "Image Version" : "2.1"
        }
}
```

Example 4.53. Metadata Reset Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>

<metadata xmlns="http://docs.openstack.org/compute/api/v1.1">
    <meta key="Server Label">Web Head 1</meta>
    <meta key="Image Version">2.1</meta>
</metadata>
```

Example 4.54. Metadata Reset Response: JSON

```
{
    "metadata" :  {
            "Server Label" : "Web Head 1",
            "Image Version" : "2.1"
        }
}
```

4.6.3. Update Metadata

Verb	URI	Description
POST	/servers/*id*/metadata	Update metadata items
POST	/images/*id*/metadata	Update metadata items

Normal Response Code(s): 200

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), buildInProgress (409)

Updates resource metadata. Updates will replace existing metadata items with the same key. Items not explicitly mentioned in the request will not be modified. An overLimit (413) fault may be thrown if the operation causes the maximum number of metadata items to be exceeded. The maximum number of key-value pairs that can be supplied per server is determined by the compute provider and may be queried via the maxServerMeta absolute limit. The maximum number of key-value pairs for an image may be queried via the maxImageMeta absolute limit.

Example 4.55. Metadata Update Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>

<metadata xmlns="http://docs.openstack.org/compute/api/v1.1">
    <meta key="Server Label">Web Head 2</meta>
</metadata>
```

Example 4.56. Metadata Update Request: JSON

```
{
    "metadata" : {
            "Server Label" : "Web Head 2"
        }
}
```

Example 4.57. Metadata Update Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>

<metadata xmlns="http://docs.openstack.org/compute/api/v1.1">
    <meta key="Server Label">Web Head 2</meta>
    <meta key="Image Version">2.1</meta>
</metadata>
```

Example 4.58. Metadata Update Response: JSON

```
{
    "metadata" : {
            "Server Label" : "Web Head 2",
            "Image Version" : "2.1"
        }
}
```

4.6.4. Get Metadata Item

Verb	URI	Description
GET	/servers/*id*/metadata/*key*	Get metadata item associated with a server
GET	/images/*id*/metadata/*key*	Get metadata item associated with an image

Normal Response Code(s): 200, 203

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404)

Retrieves a single metadata item by key. The operation does not require a request body.

Example 4.59. Metadata Item Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<meta
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    key="Server Label">Web Head 1</meta>
```

Example 4.60. Metadata Item Response: JSON

```
{
    "meta" : {
            "Server Label" : "Web Head 1"
        }
}
```

4.6.5. Set Metadata Item

Verb	URI	Description
PUT	/servers/*id*/metadata/*key*	Set a metadata item
PUT	/images/*id*/metadata/*key*	Set a metadata item

Normal Response Code(s): 200

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), badMediaType (415), buildInProgress (409)

Sets a metadata item by its key. An overLimit (413) fault may be thrown if the operation causes the maximum number of metadata items to be exceeded. The maximum number of key-value pairs that can be supplied per server is determined by the compute provider and may be queried via the maxServerMeta absolute limit. The maximum number of key-value pairs for an image may be queried via the maxImageMeta absolute limit.

Example 4.61. Metadata Item Update Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<meta
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    key="Server Label">Web Head 1</meta>
```

Example 4.62. Metadata Item Update Request: JSON

```
{
    "meta" : {
            "Server Label" : "Web Head 1"
        }
}
```

Example 4.63. Metadata Item Update Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<meta
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    key="Server Label">Web Head 1</meta>
```

Example 4.64. Metadata Item Update Response: JSON

```
{
    "meta" : {
            "Server Label" : "Web Head 1"
        }
}
```

4.6.6. Delete Metadata Item

Verb	URI	Description
DELETE	/servers/*id*/metadata/*key*	Deletes a metadata item
DELETE	/images/*id*/metadata/*key*	Deletes a metadata item

Normal Response Code(s): 204

Error Response Code(s): computeFault (400, 500, ...), serviceUnavailable (503), unauthorized (401), forbidden (403), badRequest (400), badMethod (405), overLimit (413), itemNotFound (404), buildInProgress (409)

Deletes a metadata item. The operation does not require a request body and does not contain a response body.