

# OpenStack Compute

## Developer Guide

API v2 (May 30, 2012)



# OpenStack Compute Developer Guide

API v2 (2012-05-30)

Copyright © 2009-2012 OpenStack LLC All rights reserved.

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

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

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

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

# Table of Contents

1. Overview .....	1
1.1. Intended Audience .....	1
1.2. Document Change History .....	2
1.3. Additional Resources .....	4
2. Concepts .....	5
2.1. Server .....	5
2.2. Flavor .....	5
2.3. Image .....	5
2.4. Reboot .....	5
2.5. Rebuild .....	5
2.6. Resize .....	5
3. General API Information .....	6
3.1. Authentication .....	6
3.2. Request/Response Types .....	6
3.3. Links and References .....	10
3.4. Paginated Collections .....	16
3.5. Efficient Polling with the <i>Changes-Since</i> Parameter .....	21
3.6. Limits .....	22
3.6.1. Rate Limits .....	22
3.6.2. Absolute Limits .....	23
3.6.3. Determining Limits Programmatically .....	24
3.7. Versions .....	26
3.8. Extensions .....	34
3.9. Faults .....	43
3.9.1. Synchronous Faults .....	43
3.9.2. Asynchronous Faults .....	45
4. API Operations .....	50
4.1. Servers .....	50
4.1.1. List Servers .....	50
4.1.2. Create Server .....	57
4.1.3. Get Server Details .....	67
4.1.4. Update Server .....	71
4.1.5. Delete Server .....	79
4.2. Server Addresses .....	80
4.2.1. List Addresses .....	80
4.2.2. List Addresses by Network .....	81
4.3. Server Actions .....	83
4.3.1. Change Administrator Password .....	83
4.3.2. Reboot Server .....	84
4.3.3. Rebuild Server .....	85
4.3.4. Resize Server .....	91
4.3.5. Confirm Resized Server .....	92
4.3.6. Revert Resized Server .....	93
4.3.7. Create Image .....	94
4.4. Flavors .....	97
4.4.1. List Flavors .....	97
4.4.2. Get Flavor Details .....	99
4.5. Images .....	101

4.5.1. List Images .....	101
4.5.2. Get Image Details .....	105
4.5.3. Delete Image .....	107
4.6. Metadata .....	108
4.6.1. List Metadata .....	108
4.6.2. Set Metadata .....	109
4.6.3. Update Metadata .....	111
4.6.4. Get Metadata Item .....	113
4.6.5. Set Metadata Item .....	114
4.6.6. Delete Metadata Item .....	116

## List of Tables

3.1. JSON and XML Response Formats .....	6
3.2. Sample Rate Limits .....	22
3.3. Sample Absolute Limits .....	23
3.4. Fault Elements and Error Codes .....	44
4.1. Create Server Request Attributes .....	57
4.2. Update Server Request Body Attributes .....	71
4.3. List Addresses by Network URI Parameters .....	82
4.4. Change Administrator Password Request Body Attributes .....	83
4.5. Reboot Server Request Attributes .....	84
4.6. Rebuild Server Request Attributes .....	85
4.7. Resize Server Request Attributes .....	91
4.8. Create Image Request Attributes .....	94
4.9. Set Metadata Request Attributes .....	109
4.10. Update Metadata Request Attributes .....	111
4.11. Set Metadata Item Request Attributes .....	114

## List of Examples

3.1. Request with Headers: JSON .....	7
3.2. Response with Headers: XML .....	8
3.3. Request with Extension: JSON .....	9
3.4. ID Image Reference: XML .....	10
3.5. ID Image Reference: JSON .....	11
3.6. Full Image Reference: XML .....	11
3.7. Full Image Reference: JSON .....	12
3.8. Server with Self Links: XML .....	13
3.9. Server with Self Links: JSON .....	14
3.10. Image with Alternate Link: XML .....	14
3.11. Server with Alternate Link: JSON .....	15
3.12. Images Collection, First Page: XML .....	17
3.13. Images Collection, First Page: JSON .....	17
3.14. Images Collection, Second Page: XML .....	18
3.15. Images Collection, Second Page: JSON .....	18
3.16. Images Collection, Last Page: XML .....	19
3.17. Images Collection, Last Page: JSON .....	19
3.18. Paginated Metadata in an Image: XML .....	20
3.19. Paginated Metadata in an Image: JSON .....	20
3.20. Limit Response: XML .....	24
3.21. Limit Response: JSON .....	25
3.22. Request with MIME type versioning .....	26
3.23. Request with URI versioning .....	26
3.24. Multiple Choices Response: XML .....	27
3.25. Multiple Choices Response: JSON .....	28
3.26. Versions List Request .....	29
3.27. Versions List Response: XML .....	29
3.28. Versions List Response: Atom .....	30
3.29. Versions List Response: JSON .....	30
3.30. Version Details Request .....	31
3.31. Version Details Response: XML .....	31
3.32. Version Details Response: Atom .....	32
3.33. Version Details Response: JSON .....	33
3.34. Extensions Response: XML .....	35
3.35. Extensions Response: JSON .....	36
3.36. Extension Response: xml .....	38
3.37. Extension Response: JSON .....	38
3.38. Extended Server Response: XML .....	39
3.39. Extended Server Response: JSON .....	40
3.40. Extended Action: XML .....	42
3.41. Extended Action: JSON .....	42
3.42. Fault Response: XML .....	43
3.43. Fault Response: JSON .....	43
3.44. Fault Response, Item Not Found: XML .....	44
3.45. Fault Response, Item Not Found: JSON .....	44
3.46. Fault Response, Over Limit: XML .....	45
3.47. Fault Response, Over Limit: JSON .....	45
3.48. Server In Error State: XML .....	46

3.49. Server In Error State: JSON .....	47
3.50. Image In Error State: XML .....	48
3.51. Image In Error State: JSON .....	49
4.1. List Servers Response: XML (detail) .....	52
4.2. List Servers Response: JSON (detail) .....	53
4.3. Server Create Request: XML .....	57
4.4. Server Create Request: JSON .....	59
4.5. Server Create Response: XML .....	60
4.6. Server Create Response: JSON .....	61
4.7. Creating a Server with a Access IP: XML .....	64
4.8. Creating a Server with a Access IP: JSON .....	64
4.9. Creating a Server with Multiple Access IPs: XML .....	65
4.10. Creating a Server with Multiple Access IPs: JSON .....	66
4.11. Server Details Response: XML .....	68
4.12. Server Details Response: JSON .....	69
4.13. Server Update Name Request: XML .....	71
4.14. Server Update Name Request: JSON .....	71
4.15. Server Update Name Response: XML .....	72
4.16. Server Update Name Response: JSON .....	73
4.17. Server Update Access Address Request: XML .....	75
4.18. Server Update Access Address Request: JSON .....	75
4.19. Server Update Access Address Response: XML .....	76
4.20. Server Update Access Address Response: JSON .....	77
4.21. Addresses List Response: XML .....	81
4.22. Addresses List Response: JSON .....	81
4.23. List Addresses by Network Response: XML .....	82
4.24. List Addresses by Network Response: JSON .....	82
4.25. Action Change Password Request: XML .....	83
4.26. Action Change Password Request: JSON .....	83
4.27. Action Reboot Request: XML .....	84
4.28. Action Reboot Request: JSON .....	84
4.29. Action Rebuild Request: XML .....	85
4.30. Action Rebuild Request: JSON .....	87
4.31. Action Rebuild Response: XML .....	88
4.32. Action Rebuild Response: JSON .....	89
4.33. Action Resize Request: XML .....	91
4.34. Action Resize Request: JSON .....	91
4.35. Action Confirm Resize Request: XML .....	92
4.36. Action Confirm Resize Request: JSON .....	92
4.37. Action Revert Resize Request: XML .....	93
4.38. Action Revert Resize Request: JSON .....	93
4.39. Action Create Image Request: XML .....	94
4.40. Action Create Image Request: JSON .....	96
4.41. Flavors List Response: XML (detail) .....	97
4.42. Flavors List Response: JSON (detail) .....	98
4.43. Flavor Details Response: XML .....	99
4.44. Flavor Details Response: JSON .....	100
4.45. Images List Response: XML (detail) .....	103
4.46. Images List Response: JSON (detail) .....	104
4.47. Image Details Response: XML .....	105
4.48. Image Details Response: JSON .....	106

4.49. Metadata List Response: XML .....	108
4.50. Metadata List Response: JSON .....	108
4.51. Set Metadata Request: XML .....	109
4.52. Set Metadata Request: JSON .....	109
4.53. Set Metadata Response: XML .....	110
4.54. Set Metadata Response: JSON .....	110
4.55. Update Metadata Request: XML .....	111
4.56. Update Metadata Request: JSON .....	111
4.57. Update Metadata Response: XML .....	112
4.58. Update Metadata Response: JSON .....	112
4.59. Metadata Item Response: XML .....	113
4.60. Metadata Item Response: JSON .....	113
4.61. Set Metadata Item Request: XML .....	114
4.62. Set Metadata Item Request: JSON .....	114
4.63. Set Metadata Item Response: XML .....	115
4.64. Set Metadata Item Response: JSON .....	115



# 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](https://bugs.launchpad.net/nova).

## 1.1. Intended Audience

This guide assists software developers who want to develop applications using the OpenStack Compute API. To use this information, you should have access to an account from an OpenStack Compute provider, and you should also be familiar with the following concepts:

- OpenStack Compute service
- 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
May 30, 2012	<ul style="list-style-type: none"> <li>Added more descriptive information about the request body attributes to the create server API operation.</li> </ul>
May 2, 2012	<ul style="list-style-type: none"> <li>Added <code>SHUTOFF</code> to the list of server status values for the list servers API operation.</li> <li>Updated the description for the <code>SUSPENDED</code> server status value for the list servers API operation.</li> </ul>
Apr 24, 2012	<ul style="list-style-type: none"> <li>Updated to use color-coded syntax formatting in request and response examples.</li> </ul>
Mar 25, 2012	<ul style="list-style-type: none"> <li>Added descriptions of URI parameters and request body attributes for API operations.</li> </ul>
Feb 14, 2012	<ul style="list-style-type: none"> <li>Updated the API version from v1.1 to v2.</li> <li>No longer use <code>mimetype</code> parameters to denote version.</li> </ul>
Nov 8, 2011	<ul style="list-style-type: none"> <li>Removed DRAFT designation.</li> </ul>
Sep 8, 2011	<ul style="list-style-type: none"> <li>Added <code>limit</code> and <code>marker</code> parameters to list operations.</li> <li>The rebuild action behaves just like create: an <code>imageRef</code> is used and a password may be specified.</li> <li>Added <code>tenant</code> and <code>user_id</code> attributes to server and image.</li> <li>Added <code>vcpus</code> attribute to flavors.</li> <li>We now use a <code>flavorRef</code> in the resize action.</li> </ul>
Jul 23, 2011	<ul style="list-style-type: none"> <li>Added missing response examples for server update.</li> <li>Ensure consistent HTTP status codes for all resources.</li> <li>Clarifications on setting and changing a server password.</li> <li>Minor updates to metadata section for clarity.</li> <li>Discuss alternate links.</li> <li>Removed version number from compute media types — use a media type parameter instead.</li> <li>Bought back the <code>flavorRef</code> and <code>imageRef</code> server attributes these are now only used when creating a server.</li> <li>Made the create image operation a server action.</li> <li>Added <code>minDisk</code> and <code>minRam</code> filters to flavor lists.</li> <li>Added <code>minDisk</code> and <code>minRam</code> attributes to images.</li> <li>Asynchronous faults may now contain a timestamp.</li> <li>Changes-since request returns an empty list rather than a 304.</li> <li>Added <code>DELETED</code> image status.</li> <li>Fix content length in <a href="#">Example 3.2, "Response with Headers: XML" [8]</a>.</li> <li>Fixed bad request error code in <a href="#">Section 4.1.2.1, "Server Passwords" [63]</a>.</li> <li>Compact image, server, and flavor lists should contain IDs, names, and links (Any kind of link may be included — not just self links).</li> <li>Changed metadata URI from <code>.../meta</code> to <code>.../metadata</code> for consistency.</li> </ul>
Jun 29, 2011	<ul style="list-style-type: none"> <li>Renamed Primary IP to Access IP.</li> </ul>
Jun 23, 2011	<ul style="list-style-type: none"> <li>Many minor updates based on community feedback.</li> <li>Removed sections on Content Compression, Persistent Connections, and Caching — these are operator specific. Added section on HTTP.</li> <li>A Location header is returned when creating servers/images.</li> <li>Added filters to collection of Image, Servers, and Flavors.</li> <li>Added asynchronous faults.</li> <li>Updates to links and references. Remove <code>serverRef</code>, <code>imageRef</code>, and <code>flavorRef</code> and instead embed one entity in another to provide links.</li> <li>Added primary IP addresses.</li> <li>Added forbidden fault.</li> <li>We now use a single bookmark link per entity regardless of mimetype.</li> <li>Collections are now sorted by create time.</li> <li>Previous links are no longer required.</li> <li>Added the ability to create or update multiple metadata items simultaneously.</li> <li>Minor cleanups to server and image state machine.</li> <li>Update to JSON collection format.</li> <li>Replace integer IDs with UUIDs.</li> </ul>

Revision Date	Summary of Changes
	<ul style="list-style-type: none"><li>• Removed affinityID, this will likely come in as an extension.</li></ul>
Apr 25, 2011	<ul style="list-style-type: none"><li>• Some minor cleanups in preparation for OpenStack Summit discussion.</li></ul>
Mar 11, 2011	<ul style="list-style-type: none"><li>• Many minor updates based on community feedback.</li><li>• Updates to resource linking and references.</li><li>• Better description of paginated collections.</li><li>• Metadata supported in servers and images.</li><li>• Dropped support for shared IP groups.</li><li>• IPs organized by network id, vs simply having public and private IPs.</li><li>• Generalized affinity id.</li></ul>
Feb 9, 2011	<ul style="list-style-type: none"><li>• Initial release.</li></ul>

## 1.3. Additional Resources

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

## 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. The virtualization platform should ensure that the reboot action has completed successfully even in cases in which the underlying domain/vm is paused or halted/stopped.

### 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.

Providers may return information identifying requests in HTTP response headers, for example, to facilitate communication between the provider and client users.

### 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 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 /v2/214412/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
dCBtb3ZlcyBpbjBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVz
c2lubi4uLnRoXMGaXMGdGhlIHBSYWNlIHRvIGdvIG5vdy4g
QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
ZSBwYXR0ZXJucyBiZWphbmQgYWxsIGNsb3VkcjYgYW5kIHlv
dSB3aWxsIGtub3csIHRvbywgZ2hlbiB5b3UgbGlmZCB5b3Vy
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"
  tenant_id="1234" user_id="5678"
  name="new-server-test"
  hostId="e4d909c290d0fblca068ffaddf22cbd0" 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/v2/1234/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/v2/1234/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/v2/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
      <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    </server>

```



&lt;/server&gt;

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 /v2/214412/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
dCBtb3ZlcyBpbjBqdXN0IHNlY2ggYSBkaXJlY3Rpb24gYW5k
IGF0IHNlY2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
c2lvbi4uLnRoaxMgaXMgdGhlIHBSYWNlIHRvIGdvIG5vdy4g
QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbzMgYmV5kIHRo
ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
c2VsZiBoawdoIGVub3VnaCB0byBzZWUgYmV5b25kIGHvcml6
b25zLiINCgOKLVJpY2hhcmQgQmFjaA=="
            }
        ]
    }
}
```

## 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
      dCBtb3ZlcyBpbjBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
      IGF0IHN1Y2ggYSBzcGVlZC4uLk10IGZlZWxzIGFuIGltcHVz
      c2lvbi4uLnRoXMGaXMgdGhlIHBSYWNlIHRvIGdvIG5vdy4g
      QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
      ZSBwYXR0ZXJucyBiZWpibmQgYWxsIGNsb3VkcycwYmV5b25k
      IGVhcnR1b3Vud3VnaCB0byBzZWUgYmV5b25kIGhvcml6
      b25zLiINCgOKLVJpY2hhcmQgQmFjaA==
    </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
IGF0IHN1Y2ggYSBzcGVlZC4uLk10IGZlZWxzIGFuIGltcHVz
c2lvbi4uLnRoXMGaXMgdGhlIHBSYWNlIHRvIGdvIG5vdy4g
QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
ZSBwYXR0ZXJucyBiZWpibmQgYWxsIGNsb3VkcycwYmV5b25k
dSB3aWxsIGtub3csIHRvbywgZ2hlbiB5b3UgbGlmZCB5b3Vy
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
IGF0IHN1Y2ggYSBzcGVlZC4uLk10IGZlZWxzIGFuIGltcHVz
c2lvbi4uLnRoXMGaXMgdGhlIHBSYWNlIHRvIGdvIG5vdy4g
QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
ZSBwYXR0ZXJucyBiZWpibmQgYWxsIGNsb3VkcycwYmV5b25k
dSB3aWxsIGtub3csIHRvbywgZ2hlbiB5b3UgbGlmZCB5b3Vy
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
dCBtb3ZlcyBpbjBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVz
c2lvbi4uLnRoXMGaXMGdGhlIHBSYWNlIHRvIGdvIG5vdy4g
QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
ZSBwYXR0ZXJucyBiZWpibmQgYWxsIGNsb3VkcycwYmV5b25kIHlv
dSB3aWxsIGtub3csIHRvbywgZ2h1biB5b3UgbGlmZCB5b3Vy
c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
      }
    ]
  }
}
```

For convenience, resources contain links to themselves. This allows a client to easily obtain rather than construct resource URIs. The following types of link relations are associated with resources:

- A `self` link contains a versioned link to the resource. Use these links when 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 might 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="52415800-8b69-11e0-9b19-734fcede0043" name="my-server">
  <atom:link
    rel="self"
    href="http://servers.api.openstack.org/v2/1234/servers/
52415800-8b69-11e0-9b19-734fcede0043"/>
  <atom:link
    rel="bookmark"
    href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734fcede0043"/>
</server>
```

### Example 3.9. Server with Self Links: JSON

```
{
  "server" : {
    "id" : "52415800-8b69-11e0-9b19-734f5736d2a2",
    "name" : "my-server",
    "links": [
      {
        "rel" : "self",
        "href" : "http://servers.api.openstack.org/v2/1234/servers/52415800-8b69-11e0-9b19-734f5736d2a2"
      },
      {
        "rel" : "bookmark",
        "href" : "http://servers.api.openstack.org/1234/servers/52415800-8b69-11e0-9b19-734f5736d2a2"
      }
    ]
  }
}
```

### 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/v2/1234/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/v2/1234/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 a *badRequest* (400) fault.

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/v2/1234/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/v2/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    </image>
    <atom:link
      rel="next"
      href="http://servers.api.openstack.org/v2/1234/images?limit=1&
amp;marker=52415800-8b69-11e0-9b19-734f6f006e54"/>
    </atom:link>
  </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/v2/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54"
        }
      ]
    }
  ],
  "images_links": [
    {
      "rel": "next",
      "href": "http://servers.api.openstack.org/v2/1234/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/v2/1234/images/
52415800-8b69-11e0-9b19-734f5736d2a2"/>
    </image>
    <atom:link rel="next"
      href="http://servers.api.openstack.org/v2/1234/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/v2/1234/
images/52415800-8b69-11e0-9b19-734f5736d2a2"
        }
      ]
    },
    {
      "images_links": [
        {
          "rel" : "next",
          "href" : "http://servers.api.openstack.org/v2/1234/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/v2/1234/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/v2/1234/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/v2/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54/meta?marker=ImageType"/>
    </metadata>
    <atom:link
      rel="self"
      href="http://servers.api.openstack.org/v2/1234/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/v2/1234/images/
52415800-8b69-11e0-9b19-734f6f006e54/meta?marker=ImageType"
      }
    ],
    "links": [
      {
        "rel": "self",
        "href": "http://servers.api.openstack.org/v2/1234/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/v2/224532/servers?changes-since=2011-01-24T17:08Z> would list all servers that have changed since Mon, 24 Jan 2011 17:08:00 UTC.

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” \[24\]](#). 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/v2/3542812/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 exceeds 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
<code>maxTotalRAMSize</code>	51200	Maximum total amount of RAM (MB)
<code>maxServerMeta</code>	5	Maximum number of metadata items associated with a server
<code>maxImageMeta</code>	5	Maximum number of metadata items associated with an Image
<code>maxPersonality</code>	5	The maximum number of file path/content pairs that can be supplied on server build
<code>maxPersonalitySize</code>	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	<code>/limits</code>	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.v2+xml`). 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.v2+xml
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

### Example 3.23. Request with URI versioning

```
GET /v2/214412/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.v1.0+xml"/>
      <media-type base="application/json"
        type="application/vnd.openstack.compute.v1.0+json"/>
    </media-types>

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

    <version id="v2" status="CURRENT">
      <media-types>
        <media-type base="application/xml"
          type="application/vnd.openstack.compute.v2+xml"/>
        <media-type base="application/json"
          type="application/vnd.openstack.compute.v2+xml"/>
      </media-types>

      <atom:link rel="self"
        href="http://servers.api.openstack.org/v2/1234/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.v1.0+xml"
        },
        {
          "base": "application/json",
          "type": "application/vnd.openstack.compute.v1.0+json"
        }
      ]
    },
    {
      "id": "v2",
      "status": "CURRENT",
      "links": [
        {
          "rel": "self",
          "href": "http://servers.api.openstack.org/v2/1234/servers/52415800-8b69-11e0-9b19-734f6af67565"
        }
      ],
      "media-types": [
        {
          "base": "application/xml",
          "type": "application/vnd.openstack.compute.v2+xml"
        },
        {
          "base": "application/json",
          "type": "application/vnd.openstack.compute.v2+json"
        }
      ]
    }
  ]
}
```

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="v2" status="CURRENT"
    updated="2010-12-12T18:30:02.25Z">
    <atom:link rel="self"
      href="http://servers.api.openstack.org/v2/" />
  </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/v2/</id>
    <title type="text">Version v2</title>
    <updated>2010-12-12T18:30:02.25Z</updated>
    <link rel="self" href="http://servers.api.openstack.org/v2/" />
    <content type="text">Version v2 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": "v2",
      "status": "CURRENT",
      "updated": "2010-12-12T18:30:02.25Z",
      "links": [
        {
          "rel": "self",
          "href": "http://servers.api.openstack.org/v2/"
        }
      ]
    }
  ]
}
```

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/v2/>). 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/v2/.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/v2/
```

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="v2" status="CURRENT" updated="2011-01-21T11:33:21-06:00">

  <media-types>
    <media-type base="application/xml"
      type="application/vnd.openstack.compute.v2+xml" />
    <media-type base="application/json"
      type="application/vnd.openstack.compute.v2+json" />
  </media-types>

  <atom:link rel="self"
    href="http://servers.api.openstack.org/v2/" />

  <atom:link rel="describedby"
    type="application/pdf"
    href="http://docs.rackspacecloud.com/servers/api/v2/cs-
devguide-20110125.pdf" />

  <atom:link rel="describedby"
    type="application/vnd.sun.wadl+xml"
    href="http://docs.rackspacecloud.com/servers/api/v2/
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 v2</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/v2/cs-
devguide-20110125.pdf"/>
    <link rel="describedby" type="application/vnd.sun.wadl+xml"
      href="http://docs.rackspacecloud.com/servers/api/v2/application.
wadl"/>
    <content type="text">Version v2 CURRENT (2011-01-21T11:33:21-06:00)</
content>
  </entry>
</feed>
```



### Example 3.33. Version Details Response: JSON

```
{
  "version" : {
    "id" : "v2",
    "status" : "CURRENT",
    "updated" : "2011-01-21T11:33:21-06:00",
    "links": [
      {
        "rel" : "self",
        "href" : "http://servers.api.openstack.org/v2/"
      },
      {
        "rel" : "describedby",
        "type" : "application/pdf",
        "href" : "http://docs.rackspacecloud.com/servers/api/v2/cs-devguide-20110125.pdf"
      },
      {
        "rel" : "describedby",
        "type" : "application/vnd.sun.wadl+xml",
        "href" : "http://docs.rackspacecloud.com/servers/api/v2/application.wadl"
      }
    ],
    "media-types": [
      {
        "base" : "application/xml",
        "type" : "application/vnd.openstack.compute.v2+xml"
      },
      {
        "base" : "application/json",
        "type" : "application/vnd.openstack.compute.v2+json"
      }
    ]
  }
}
```

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	<code>/extensions/<i>alias</i></code>	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 \[39\]](#) and [3.39 \[40\]](#) is defined in the RS-CBS namespace. Actions work in exactly the same manner as illustrated in [Examples 3.40 \[42\]](#) and [3.41 \[42\]](#). 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"
    tenant_id="1234" user_id="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/v2/1234/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/v2/1234/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/v2/1234/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",
      "tenant_id": "1234",
      "user_id": "5678",
      "name": "sample-server",
      "updated": "2010-10-10T12:00:00Z",
      "created": "2010-08-10T12:00:00Z",
      "hostId": "e4d909c290d0fblca068ffaddf22cbd0",
      "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/v2/1234/
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/v2/1234/
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/v2/1234/
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 `ERROR` 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 occurred.

### Example 3.48. Server In Error State: 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"
  tenant_id="1234" user_id="5678"
  name="sample-server" status="ERROR"
  created="2010-08-10T12:00:00Z"
  progress="66" hostId="e4d909c290d0fblca068ffaaff22cbd0">
  <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/v2/1234/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 State: JSON**

```
{
  "server": {
    "id": "52415800-8b69-11e0-9b19-734f0000ffff",
    "tenant_id": "1234",
    "user_id": "5678",
    "name": "sample-server",
    "created": "2010-08-10T12:00:00Z",
    "hostId": "e4d909c290d0fblca068ffaaff22cbd0",
    "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/v2/1234/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 State: 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 occurred</message>
    <details>Error details</details>
  </fault>
  <atom:link
    rel="self"
    href="http://servers.api.openstack.org/v2/1234/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 State: 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/v2/1234/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	<code>/servers?image=imageRef&amp;flavor=flavorRef&amp;name=serverName&amp;status=serverStatus&amp;marker=markerID&amp;limit=int&amp;changes-since=dateTime</code>	Lists IDs, names, and links for all servers.
GET	<code>/servers/detail?image=imageRef&amp;flavor=flavorRef&amp;name=serverName&amp;status=serverStatus&amp;marker=markerID&amp;limit=int&amp;changes-since=dateTime</code>	Lists all details for all servers.

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)

You can filter the list of servers by image, flavor, name, and status through the respective query parameters.

To filter the list of servers returned in the response body, specify one or more of the following optional parameters in the request URI:

image	The image reference specified as an ID or full URL.
flavor	The flavor reference specified as an ID or full URL.
server	The server name.
status	The server status. See <a href="#">Server Status Values [51]</a> .
marker	The ID of the last item in the previous list. See <a href="#">Section 3.4, "Paginated Collections" [16]</a> .
limit	The page size. See <a href="#">Section 3.4, "Paginated Collections" [16]</a> .
changes-since	The changes-since time. The list contains servers that have been deleted since the changes-since time. See <a href="#">Section 3.5, "Efficient Polling with the Changes-Since Parameter" [21]</a> .

This operation does not require a request body.

This operation lists the servers associated with your account. The list does not include servers that have been deleted.

Servers contain a status attribute that indicates the current server state. You can filter on the server status when you complete a list servers request. The server status is returned in the response body. The server status is one of the following values:

## Server Status Values

- **ACTIVE.** The server is active.
- **BUILD.** The server has not finished the original build process.
- **DELETED.** The server is deleted.
- **ERROR.** The server is in error.
- **HARD\_REBOOT.** The server is hard rebooting. This is equivalent to pulling the power plug on a physical server, plugging it back in, and rebooting it.
- **PASSWORD.** The password is being reset on the server.
- **REBOOT.** The server is in a soft reboot state. A reboot command was passed to the operating system.
- **REBUILD.** The server is currently being rebuilt from an image.
- **RESCUE.** The server is in rescue mode.
- **RESIZE.** Server is performing the differential copy of data that changed during its initial copy. Server is down for this stage.
- **REVERT\_RESIZE.** The resize or migration of a server failed for some reason. The destination server is being cleaned up and the original source server is restarting.
- **SHUTOFF.** The virtual machine (VM) was powered down by the user, but not through the OpenStack Compute API. For example, the user issued a `shutdown -h` command from within the server instance. If the OpenStack Compute manager detects that the VM was powered down, it transitions the server instance to the SHUTOFF status. If you use the OpenStack Compute API to restart the instance, the instance might be deleted first, depending on the value in the `shutdown_terminate` database field on the Instance model.
- **SUSPENDED.** The server is suspended, either by request or necessity. This status appears for only the following hypervisors: XenServer/XCP, KVM, and ESXi. Review support tickets or contact Rackspace support to determine why the server is in this state.
- **UNKNOWN.** The state of the server is unknown. Contact your cloud provider.
- **VERIFY\_RESIZE.** System is awaiting confirmation that the server is operational after a move or resize.

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



### Note

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

The following examples show an XML and JSON response for the list servers API operation:

**Example 4.1. List Servers 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"
    tenant_id="1234" user_id="5678"
    name="sample-server" status="BUILD"
    progress="60" hostId="e4d909c290d0fblca068ffaddf22cbd0"
    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/v2/1234/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/v2/1234/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/v2/1234/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"
    tenant_id="1234" user_id="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/v2/1234/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/v2/1234/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/v2/1234/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. List Servers Response: JSON (detail)

```

{
  "servers": [
    {
      "id": "52415800-8b69-11e0-9b19-734f6af67565",
      "tenant_id": "1234",
      "user_id": "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/v2/1234/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/v2/1234/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"
    }
  ],
  "private": [
    {
      "version": 4,
      "addr": "67.23.10.131"
    },
    {
      "version": 6,
      "addr": "::babe:4317:0A83"
    }
  ]
},
"metadata": {
  "Server Label": "Web Head 1",
  "Image Version": "2.1"
}
```

```

    },
    "links": [
      {
        "rel": "self",
        "href": "http://servers.api.openstack.org/v2/1234/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",
    "user_id": "5678",
    "name": "sample-server2",
    "tenant_id": "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/v2/1234/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/v2/1234/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/v2/1234/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	Creates 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” \[10\]](#)). 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 **GETs** on the server.

The following table describes the required and optional attributes that you can specify in the request body:

**Table 4.1. Create Server Request Attributes**

Name	Description	Required
imageRef	The image reference for the desired image for your server instance.  Specify as an ID or full URL. To get a list of available images, see <a href="#">Section 4.5.1, “List Images” [101]</a> .	Yes
flavorRef	The flavor reference for the desired flavor for your server instance.  Specify as an ID or full URL. To get a list of available flavors, see <a href="#">Section 4.4.1, “List Flavors” [97]</a> .	Yes
name	The server name. Choose any name you like.	Yes
metadata	Metadata key and value pairs. For information, see <a href="#">Section 4.1.2.2, “Server Metadata” [63]</a> .	No
personality	File path and contents. For information, see <a href="#">Section 4.1.2.3, “Server Personality” [63]</a> .	No

### 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
    dCBtb3ZlcyBpbiBqdXN0IHNlY2ggYSBkaXJlY3Rpb24gYW5k
    IGF0IHNlY2ggYSBzcGVlZC4uLk10IGZlZWxzIGFuIGltcHVz
    c2lvbi4uLnRoXMGaXMgdGhlIHBSYWNlIHRvIGdvIG5vdy4g
    QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
    ZSBwYXR0ZXJucyBiZWphbmQgYWxsIGNsb3VkcWwgYW5kIHlv
    dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmZCB5b3Vy
    c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
    b25zLiINCgOKLVJpY2hhcmQgQmFjaA==
  </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" : "ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBpdCBtb3ZlcyBpbjBqdXN0IHNlY2ggYSBkaXJlY3Rpb24gYW5kIGF0IHNlY2ggYSBzcGVlZC4uLk10IGZlZWxzIGFuIGltcHVsc2lubi4uLnRoaXMgaXMgdGhlIHBSYWNlIHRvIGdvIG5vdy4gQnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRoZSBwYXR0ZXJucyBiZWphbmQgYWxsIGNsb3VkcjYwY5kIHlvdSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlm dCB5b3Vy 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"
  tenant_id="1234" user_id="5678"
  name="new-server-test"
  hostId="e4d909c290d0fb1ca068ffaddf22cbd0" progress="0"
  status="BUILD" adminPass="GFflj9aP"
  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/v2/1234/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/v2/1234/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/v2/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    <atom:link
      rel="bookmark"
      href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    </server>
  </server>

```

### Example 4.6. Server Create Response: JSON

```
{
  "server": {
    "id": "52415800-8b69-11e0-9b19-734f565bc83b",
    "tenant_id": "1234",
    "user_id": "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": "GFflj9aP",
    "image" : {
      "id": "52415800-8b69-11e0-9b19-734f6f006e54",
      "name": "CentOS 5.2",
      "links": [
        {
          "rel": "self",
          "href": "http://servers.api.openstack.org/v2/1234/
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/v2/1234/
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/v2/1234/servers/  
52415800-8b69-11e0-9b19-734fcede0043"  
        },  
        {  
            "rel": "bookmark",  
            "href": "http://servers.api.openstack.org/1234/servers/  
52415800-8b69-11e0-9b19-734fcede0043"  
        }  
    ]  
    }  
}
```

### 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 `ERROR` state if the complexity requirements are not met. In this case, a client may issue a change password action to reset the server password (see [Section 4.3.1, "Change Administrator Password" \[83\]](#)).

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" \[108\]](#) 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 can customize the personality of a server instance by injecting data into its file system. For example, you might want to insert ssh keys, set configuration files, or store data that you want to retrieve from inside the instance. This feature provides a minimal amount of launch-time personalization. If you require significant customization, create a custom image.

Follow these guidelines when you inject files:

- The maximum size of the file path data is 255 bytes.
- Encode the file contents as a Base64 string. The maximum 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



#### Note

The maximum limit refers to the number of bytes in the decoded data and not the number of characters in the encoded data.

- You can inject text files only. You cannot inject binary or zip files into a new build.
- The maximum number of file path/content pairs that you can supply is also determined by the compute provider and is defined by the `maxPersonality` absolute limit.
- The absolute limit, `maxPersonalitySize`, is a byte limit that is guaranteed to apply to all images in the deployment. Providers can set additional per-image personality limits.

The file injection might not occur until after the server is built and booted.

During file injection, any existing files that match specified files are renamed to include the bak extension appended with a time stamp. For example, if the file `/etc/passwd` exists, it is backed up as `/etc/passwd.bak.1246036261.5785`.

After file injection, personality files are accessible by only system administrators. For example, on Linux, all files have root and the root group as the owner and group owner, respectively, and 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 underlying 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" \[80\]](#)). 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" \[71\]](#) 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/ <i>id</i>	Lists 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)

Specify the server ID as *id* in the URI.

This operation does not require a request body.

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

**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"
  tenant_id="1234" user_id="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/v2/1234/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/v2/1234/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/v2/1234/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",
    "tenant_id": "1234",
    "user_id": "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/v2/1234/
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/v2/1234/
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/v2/1234/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/ <i>id</i>	Updates the editable attributes for the specified server.

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 enables you to update the editable attributes of a specified server.

Specify the server ID as *id* in the URI.

The following table describes the attributes that you can specify in the request body:

**Table 4.2. Update Server Request Body Attributes**

Attribute	Description
name	The name of the server. If you edit the server name, the server host name does not change. Also, server names are not guaranteed to be unique.
accessIPv4	The IP version 4 address.
accessIPv6	The IP version 6 address.

### 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"
  tenant_id="1234" user_id="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/v2/1234/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/v2/1234/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/v2/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
      <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    </server>

```



### Example 4.16. Server Update Name Response: JSON

```
{
  "server": {
    "id": "52415800-8b69-11e0-9b19-734f565bc83b",
    "tenant_id": "1234",
    "user_id": "5678",
    "name": "new-server-test",
    "created": "2010-11-11T12:00:00Z",
    "updated": "2010-11-12T12:44:44Z",
    "hostId": "e4d909c290d0fblca068ffaddf22cbd0",
    "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/v2/1234/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/v2/1234/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/v2/1234/servers/52415800-8b69-11e0-9b19-734fcede0043"
    },
    {
      "rel": "bookmark",
      "href": "http://servers.api.openstack.org/1234/servers/52415800-8b69-11e0-9b19-734fcede0043"
    }
  ]
}
```

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"
  tenant_id="1234" user_id="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/v2/1234/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/v2/1234/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">Apachel</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/v2/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
      <atom:link
        rel="bookmark"
        href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    </server>

```

**Example 4.20. Server Update Access Address Response: JSON**

```
{
  "server": {
    "id": "52415800-8b69-11e0-9b19-734f565bc83b",
    "tenant_id": "1234",
    "user_id": "5678",
    "name": "new-server-test",
    "created": "2010-11-11T12:00:00Z",
    "updated": "2010-11-12T12:55:55Z",
    "hostId": "e4d909c290d0fblca068ffaddf22cbd0",
    "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/v2/1234/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/v2/1234/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/v2/1234/servers/52415800-8b69-11e0-9b19-734fcede0043"
    },
    {
      "rel": "bookmark",
      "href": "http://servers.api.openstack.org/1234/servers/52415800-8b69-11e0-9b19-734fcede0043"
    }
  ]
}
```

## 4.1.5. Delete Server

Verb	URI	Description
DELETE	/servers/ <i>id</i>	Deletes 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 specified cloud server instance from the system.

Specify the ID for the cloud server instance as *id* in the URI.

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

## 4.2. Server Addresses

### 4.2.1. List Addresses

Verb	URI	Description
GET	/servers/ <i>id</i> /ips	Lists all networks and addresses associated with a 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), buildInProgress (409)

This operation lists all networks and addresses associated with a specified server.

Specify the server ID as *id* in the URI.

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/ <i>id</i> /ips/ <i>networkID</i>	Lists addresses associated with a specified server and network.

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 lists all addresses associated with a specified server and network.

The following table describes the required parameters that you specify in the request URI:

**Table 4.3. List Addresses by Network URI Parameters**

Name	Description
id	The server ID.
networkID	The network ID.

This operation does not require a request body.

#### Example 4.23. List Addresses by Network Response: 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 Response: 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 Administrator Password

Verb	URI	Description
POST	/servers/ <i>id</i> /action	Changes the administrator password for a 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 → PASSWORD → ACTIVE

ACTIVE → PASSWORD → ERROR (on error)

ERROR → PASSWORD → ACTIVE (password reset after error)

This operation changes the administrator password for a specified server.

Specify the server ID as *id* in the URI.

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

In the request body, specify the *changePassword* action followed by attributes. The following table describes the attributes that you specify in the request body:

**Table 4.4. Change Administrator Password Request Body Attributes**

Attribute	Description	Required
adminPass	The administrator password.	Yes

This operation does not return a response body.

#### Example 4.25. Action Change Password Request: XML

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

#### Example 4.26. Action Change Password Request: JSON

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

## 4.3.2. Reboot Server

Verb	URI	Description
POST	/servers/ <i>id</i> /action	Reboots 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)

This operation enables you to complete either a soft or hard reboot of a specified 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.

Specify the server ID as *id* in the URI.

In the request body, specify the *reboot* action followed by attributes.

The following table describes the attributes that you specify in the request body:

**Table 4.5. Reboot Server Request Attributes**

Attribute	Description	Required
type	The type of reboot: <ul style="list-style-type: none"><li><i>SOFT</i>. The operating system is signaled to restart, which allows for a graceful shutdown of all processes.</li><li><i>HARD</i>. Equivalent to power cycling the server.</li></ul>	No. Default is <i>SOFT</i> .

This operation does not return a response body.

### Example 4.27. Action Reboot Request: 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 Request: JSON

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

### 4.3.3. Rebuild Server

Verb	URI	Description
POST	/servers/ <i>id</i> /action	Rebuilds 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 operation 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, metadata, accessIPv4, or accessIPv6 are specified, they will replace existing values, otherwise they do not change.

A rebuild operation always removes data injected into the file system through server personality. You can reinsert data into the file system during the rebuild. The full URL to the rebuilt server is returned through the `Location` header.

Specify the server ID as *id* in the URI.

In the request body, specify the *rebuild* action followed by attributes.

The following table describes the attributes that you specify in the request body:

**Table 4.6. Rebuild Server Request Attributes**

Attribute	Description	Required
name	The new name for the server.	Yes
imageRef	The image reference. Specify as an ID or full URL.	Yes
accessIPv4	The IP version 4 address.	No
accessIPv6	The IP version 6 address.	No
adminPass	The administrator password.	Yes
metadata	A metadata key and value pair.	No
personality	The file path and file contents.	No

#### 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/v2/32278/images/
52415800-8b69-11e0-9b19-734f6f006e54"
  accessIPv4="172.19.0.1"
  accessIPv6="fe80::100"
  adminPass="GFf1j9aP">
  <metadata>
```

```
<meta key="My Server Name">Apache1</meta>  
</metadata>  
<personality>  
  <file path="/etc/banner.txt">  
    ICAgICAgDQoiQSBjbG9lZCBkb2VzIG5vdCBrbm93IHdoeSBp  
    dCBtb3ZlcYBpbjBqdXN0IHNlY2ggYSBkaXJlY3Rpb24gYW5k  
    IGF0IHNlY2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs  
    c2lvbi4uLnRoZXMGaXMgdGhlIHBSYWNlIHhvIGdvIG5vdy4g  
    QnV0IHROZSBza3kga25vd3MgdGhlIHJlYXNvbGNgYW5kIHRO  
    ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv  
    dSB3aWxsIGtub3csIHRvbywgd2h1biB5b3UgbGlmdCB5b3Vy  
    c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGHvcml6  
    b25zLiINCgOKLVJPY2hhcmQgQmFjaA==  
  </file>  
</personality>  
</rebuild>
```

**Example 4.30. Action Rebuild Request: JSON**

```
{
  "rebuild" : {
    "imageRef" : "https://servers.api.rackspacecloud.com/v2/32278/images/52415800-8b69-11e0-9b19-734f6f006e54",
    "name" : "newName",
    "adminPass" : "GFf1j9aP",
    "accessIPv4" : "172.19.0.1",
    "accessIPv6" : "fe80::100",
    "metadata" : {
      "My Server Name" : "Apache1"
    },
    "personality" : [
      {
        "path" : "/etc/banner.txt",
        "contents" : "ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBpdCBtb3ZlcyBpbIBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5kIGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltchVsY2lubi4uLnRoXMGaXMgdGhlIHBSYWNlIHRvIGdvIG5vdy4gQnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRoZSBwYXR0ZXJucyBiZWpibmQgYWxsIGNsb3VkcycwYW5kIHlvdSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3VyY2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
      }
    ]
  }
}
```

**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"
  tenant_id="1234" user_id="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/v2/1234/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/v2/1234/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/v2/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    <atom:link
      rel="bookmark"
      href="http://servers.api.openstack.org/1234/servers/
52415800-8b69-11e0-9b19-734f6f006e54"/>
    </server>
  </server>

```



**Example 4.32. Action Rebuild Response: JSON**

```
{
  "server": {
    "id": "52415800-8b69-11e0-9b19-734f565bc83b",
    "tenant_id": "1234",
    "user_id": "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": "GFflj9aP",
    "image" : {
      "id": "52415800-8b69-11e0-9b19-734f6f006e54",
      "name": "CentOS 5.2",
      "links": [
        {
          "rel": "self",
          "href": "http://servers.api.openstack.org/v2/1234/
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/v2/1234/
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/v2/1234/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/ <i>id</i> /action	Resizes 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 operation 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.

Specify the server ID as *id* in the URI.

In the request body, specify the *resize* action followed by attributes. The following table describes the attributes that you specify in the request body:

**Table 4.7. Resize Server Request Attributes**

Attribute	Description	Required
name	The name for the resized server.	Yes
flavorRef	The flavor ID : <code>flavor_id</code>	Yes

This operation does not return a response body.

### Example 4.33. Action Resize Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<resize
  xmlns="http://docs.openstack.org/compute/api/v1.1"
  flavorRef="4" />
```

### Example 4.34. Action Resize Request: JSON

```
{
  "resize" : {
    "flavorRef" : "4"
  }
}
```

## 4.3.5. Confirm Resized Server

Verb	URI	Description
POST	/servers/ <i>id</i> /action	Confirms 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 a problem exists. 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.

Specify the server ID as *id* in the URI.

In the request body, specify the *confirmResize* action.

This operation does not return a response body.

### Example 4.35. Action Confirm Resize Request: XML

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

### Example 4.36. Action Confirm Resize Request: JSON

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

## 4.3.6. Revert Resized Server

Verb	URI	Description
POST	/servers/ <i>id</i> /action	Cancels and reverts 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 → REVERT\_RESIZE → ACTIVE  
                                VERIFY\_RESIZE → REVERT\_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 the revert resize 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.

Specify the server ID as *id* in the URI.

In the request body, specify the *revertResize* action followed by attributes.

This operation does not return a response body.

### Example 4.37. Action Revert Resize Request: XML

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

### Example 4.38. Action Revert Resize Request: JSON

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

This operation does not return a response body.

## 4.3.7. Create Image

Verb	URI	Description
POST	/servers/ <i>id</i> /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 operation 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" \[105\]](#) for details.

Custom image metadata can also be supplied when creating an image. See [Section 4.6, "Metadata" \[108\]](#) 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 through the `maxImageMeta` absolute limit.

Specify the server ID as *id* in the URI.

In the request body, specify the *createImage* action followed by attributes. The following table describes the attributes that you specify in the request body:

**Table 4.8. Create Image Request Attributes**

Attribute	Description	Required
name	The name for the new image.	Yes
metadata	Key and value pairs for metadata.	No

This operation does not return a response body.



### Note

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

### Example 4.39. Action Create Image Request: 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 Request: 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	<code>/flavors?minDisk=minDiskInGB&amp;minRam=minRamInMB&amp;marker=markerID&amp;limit=int</code>	Lists IDs, names, and links for all available flavors.
GET	<code>/flavors/detail?minDisk=minDiskInGB&amp;minRam=minRamInMB&amp;marker=markerID&amp;limit=int</code>	Lists all details for all available flavors.

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 lists information for all available flavors.

To filter the list of flavors returned in the response body, you can specify the following optional parameters in the URI:

<code>minDisk=minDiskInGB&amp;</code>	Filters the list of flavors to those with the specified minimum number of gigabytes of disk storage.
<code>minRam=minRamInMB&amp;</code>	Filters the list of flavors to those with the specified minimum amount of RAM in megabytes.
<code>marker=markerID&amp;</code>	The ID of the last item in the previous list. See <a href="#">Section 3.4, "Paginated Collections"</a> [16].
<code>limit=int</code>	Sets the page size. See <a href="#">Section 3.4, "Paginated Collections"</a> [16].

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/v2/1234/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/v2/1234/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/v2/1234/
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/v2/1234/
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/ <i>id</i>	Lists 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)

Specify the flavor ID as *id* in the URI.

This operation does not require a request body.

This operation returns details of the specified flavor in the response 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/v2/1234/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/v2/1234/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	<code>/images?server=serverRef&amp;name=imageName&amp;status=imageStatus&amp;changes-since=dateTime&amp;marker=markerID&amp;limit=int&amp;type=(BASE SERVER)</code>	Lists IDs, names, and links for all available images.
GET	<code>/images/detail?server=serverRef&amp;name=imageName&amp;status=imageStatus&amp;changes-since=dateTime&amp;marker=markerID&amp;limit=int&amp;type=(BASE SERVER)</code>	Lists all details for all available images.

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 lists all images visible by the account.

To filter the list of images returned in the response body, you can specify the following optional parameters in the URI:

<code>server=serverRef&amp;</code>	Filters the list of images by server. Specify the server reference by ID or by full URL.
<code>name=imageName&amp;</code>	Filters the list of images by image name.
<code>status=imageStatus&amp;</code>	Filters the list of images by status. In-flight images will have the status attribute set to <code>SAVING</code> and the conditional progress element (0-100% completion) will also be returned. Other possible values for the status attribute include: <code>UNKNOWN</code> , <code>ACTIVE</code> , <code>SAVING</code> , <code>ERROR</code> , and <code>DELETED</code> . Images with an <code>ACTIVE</code> status are available for install.
<code>changes-since=dateTime&amp;</code>	Filters the list of images to those that have changed since the changes-since time. See <a href="#">Section 3.5, "Efficient Polling with the Changes-Since Parameter" [21]</a> for details.
<code>marker=markerID&amp;</code>	The ID of the last item in the previous list. See <a href="#">Section 3.4, "Paginated Collections" [16]</a> .
<code>limit=int</code>	Sets the page size. See <a href="#">Section 3.4, "Paginated Collections" [16]</a> .

*type*={BASE|SERVER}

Filters base images or server backups.

This operation does not require a request body. The optional `minDisk` and `minRam` attributes set the minimum disk and RAM requirements needed to create a server with the image.

**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"
    tenant_id="12345"
    user_id="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/v2/1234/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"
    tenant_id="12345"
    user_id="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/v2/1234/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/v2/1234/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",
      "tenant_id": "12345",
      "user_id": "joe",
      "status": "ACTIVE",
      "metadata": {
        "ImageType": "Gold",
        "ImageVersion": "1.5"
      },
      "links": [
        {
          "rel": "self",
          "href": "http://servers.api.openstack.org/v2/1234/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",
      "tenant_id": "12345",
      "user_id": "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/v2/1234/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/v2/1234/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/ <i>id</i>	Lists 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)

Specify the image ID as *id* in the URI.

This operation does not require a request body.

This operation returns details of the specified image in the response 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"
  tenant_id="12345"
  user_id="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/v2/1234/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/v2/1234/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",
    "tenant_id" : "12345",
    "user_id" : "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/v2/1234/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/v2/1234/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/ <i>id</i>	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 the specified image from the system.

Specify the image ID as *id* in the URI.

Images are immediately removed.

This operation does not require a request body.

This operation does not return 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/ <i>id</i> /metadata	Lists metadata associated with a server.
GET	/images/ <i>id</i> /metadata	Lists 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.

Specify the server or image ID as *id* in the URI.

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="Label">Web</meta>
  <meta key="Version">2.1</meta>
</metadata>
```

#### Example 4.50. Metadata List Response: JSON

```
{
  "metadata" : {
    "Label" : "Web",
    "Version" : "2.1"
  }
}
```

## 4.6.2. Set Metadata

Verb	URI	Description
PUT	/servers/ <i>id</i> /metadata	Sets metadata for a specified server.
PUT	/images/ <i>id</i> /metadata	Sets metadata for a specified image.

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 specified image or server.

Specify the image or server ID as *id* in the URI.

In the request body, specify the *metadata* element followed by attributes. The following table describes the attributes that you specify in the request body:

**Table 4.9. Set Metadata Request Attributes**

Attribute	Description	Required
meta	Key and value pairs for metadata.	Yes

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. Set Metadata Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata xmlns="http://docs.openstack.org/compute/api/v1.1">
  <meta key="Label">Web</meta>
  <meta key="Version">2.1</meta>
</metadata>
```

### Example 4.52. Set Metadata Request: JSON

```
{
  "metadata" : {
    "Label" : "Web",
    "Version" : "2.1"
  }
}
```

**Example 4.53. Set Metadata Response: XML**

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

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

**Example 4.54. Set Metadata Response: JSON**

```
{
  "metadata" : {
    "Label" : "Web",
    "Version" : "2.1"
  }
}
```

## 4.6.3. Update Metadata

Verb	URI	Description
POST	/servers/ <i>id</i> /metadata	Updates metadata items for the specified server.
POST	/images/ <i>id</i> /metadata	Updates metadata items for the specified image.

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.

Specify the image or server ID as *id* in the URI.

In the request body, specify the *metadata* element followed by attributes. The following table describes the attributes that you specify in the request body:

**Table 4.10. Update Metadata Request Attributes**

Attribute	Description	Required
meta	Key and value pairs for metadata.	Yes

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. Update Metadata Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata xmlns="http://docs.openstack.org/compute/api/v1.1">
  <meta key="Label">Web2</meta>
</metadata>
```

### Example 4.56. Update Metadata Request: JSON

```
{
  "metadata" : {
    "Label" : "Web2"
  }
}
```

**Example 4.57. Update Metadata Response: XML**

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

<metadata xmlns="http://docs.openstack.org/compute/api/v1.1">
  <meta key="Label">Web2</meta>
  <meta key="Version">2.1</meta>
</metadata>
```

**Example 4.58. Update Metadata Response: JSON**

```
{
  "metadata" : {
    "Label" : "Web2",
    "Version" : "2.1"
  }
}
```



## 4.6.4. Get Metadata Item

Verb	URI	Description
GET	/servers/ <i>id</i> /metadata/ <i>key</i>	Gets a metadata item associated with a server.
GET	/images/ <i>id</i> /metadata/ <i>key</i>	Gets a 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.

In the URI, specify the image or server ID as *id* and the key as *key*.

This 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="Label">Web</meta>
```

### Example 4.60. Metadata Item Response: JSON

```
{
  "meta" : {
    "Label" : "Web"
  }
}
```

## 4.6.5. Set Metadata Item

Verb	URI	Description
PUT	/servers/ <i>id</i> /metadata/ <i>key</i>	Sets a metadata item for a specified server.
PUT	/images/ <i>id</i> /metadata/ <i>key</i>	Sets a metadata item for a specified image.

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.

In the URI, specify the image or server ID as *id* and the key as *key*.

In the request body, specify the *metadata* element followed by attributes. The following table describes the attributes that you specify in the request body:

**Table 4.11. Set Metadata Item Request Attributes**

Attribute	Description	Required
meta	Key and value pairs for metadata.	Yes

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. Set Metadata Item Request: XML

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

### Example 4.62. Set Metadata Item Request: JSON

```
{
  "meta" : {
    "Label" : "Web"
  }
}
```

**Example 4.63. Set Metadata Item Response: XML**

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

**Example 4.64. Set Metadata Item Response: JSON**

```
{
  "meta" : {
    "Label" : "Web"
  }
}
```

## 4.6.6. Delete Metadata Item

Verb	URI	Description
DELETE	/servers/ <i>id</i> /metadata/ <i>key</i>	Deletes a metadata item for the specified server.
DELETE	/images/ <i>id</i> /metadata/ <i>key</i>	Deletes a metadata item for 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), buildInProgress (409)

Deletes a metadata item.

In the URI, specify the image or server ID as *id* and the key as *key*.

The operation does not require a request body.

The operation does not return a response body.