

Cloud Servers™ Developer Guide

07/13/09

API v1.0 (Beta)

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

Table of Contents

Overview.....	1
Intended Audience.....	1
Concepts.....	2
Server	2
Flavor	2
Image	2
Backup Schedule.....	2
Reboot.....	2
Rebuild	3
Resize	3
Shared IP Address.....	3
Shared IP Group.....	3
General API Information.....	4
Authentication.....	4
Request/Response Types.....	5
Content Compression.....	7
Persistent Connections.....	7
Paginated Collections.....	7
Caching.....	7
Efficient Polling with the Changes-Since Parameter	8
Limits.....	8
Rate Limits.....	8
Absolute Limits.....	9
Determining Limits Programmatically	9
API Version.....	12
Faults.....	14
API Operations.....	17
Servers.....	17
List Servers.....	17
Create Server	20
Get Server Details	24
Update Server Name / Administrative Password	25

Delete Server	26
Server Addresses.....	27
List Addresses.....	27
List Public Addresses.....	28
List Private Addresses.....	28
Share an IP Address	29
Unshare an IP Address	30
Server Actions.....	31
Reboot Server	31
Rebuild Server	32
Resize Server.....	33
Confirm Resized Server	34
Revert Resized Server.....	35
Flavors.....	36
List Flavors	36
Get Flavor Details	37
Images.....	39
List Images	39
Create an Image.....	41
Get Image Details	43
Backup Schedules.....	44
List Backup Schedules	46
Create / Update Backup Schedule	47
Delete Backup Schedule	48
Shared IP Groups.....	49
List Shared IP Groups	49
Create Shared IP Group	50
Get Shared IP Group Details	52
Delete Shared IP Group	53
Appendix.....	54
HTTP/1.1 Response Codes.....	54
Cloud Servers API Roadmap.....	55
Additional Resources.....	55

Overview

Cloud Servers is a compute service that provides server capacity in the cloud. Cloud Servers come in different flavors of memory, disk space, and CPU, and can be provisioned in minutes. There are no contracts or commitments. You may use a cloud server for as long or as little as you choose. You pay only for what you use and pricing is by the hour. Interactions with Cloud Servers can occur via the Rackspace Cloud Control Panel (GUI) or programmatically via the Cloud Servers API.

We welcome feedback, comments, and bug reports at support@rackspacecloud.com.

Intended Audience

This guide is intended for software developers who want to develop applications using the Cloud Servers API. It assumes the reader has a general understanding of the Cloud Servers service and is familiar with:

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

Concepts

The Cloud Servers system has several key concepts that are important to understand:

Server

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

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.

Image

An image is a collection of files used to create or rebuild a server. Rackspace provides 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.

Backup Schedule

A backup schedule can be defined to create server images at regular intervals (daily and weekly). Backup schedules are configurable per server.

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.

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.

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.

Shared IP Address

Public IP addresses can be shared across multiple servers for use in various high availability scenarios. When an IP address is shared to another server, the cloud network restrictions are modified to allow each server to listen to and respond on that IP address (you may optionally specify that the target server network configuration be modified). Shared IP addresses can be used with many standard heartbeat facilities (e.g. keepalived) that monitor for failure and manage IP failover.

Shared IP Group

A shared IP group is a collection of servers that can share IPs with other members of the group. Any server in a group can share one or more public IPs with any other server in the group. With the exception of the first server in a shared IP group, servers must be launched into shared IP groups. A server may only be a member of one shared IP group.

General API Information

The Cloud Servers API is implemented using a RESTful web service interface. Like other products in the Rackspace Cloud suite, Cloud Servers shares a common token authentication system that allows seamless access between products and services.

Note: All requests to authenticate and operate against Cloud Servers are performed using SSL over HTTP (HTTPS) on TCP port 443.

Authentication

Each REST request against the Cloud Servers system requires the inclusion of a specific authorization token HTTP x-header defined as `X-Auth-Token`. Clients obtain this token, along with the Cloud Servers API URL, by first using the Rackspace Cloud Authentication Service and supplying a valid username and API access key.

Request

The Rackspace Cloud Authentication Service serves as the entry point to all Rackspace Cloud APIs and is itself a RESTful web service. It is accessible at <https://auth.api.rackspacecloud.com/v1.0>. In order to authenticate, you must supply your username and API access key in the `X-Auth-User` and `X-Auth-Key` x-headers, respectively. Username is your common Rackspace Cloud username and your API access key can be obtained from the Rackspace Cloud Control Panel in the *Your Account | API Access* section.

Sample Request

```
GET /v1.0 HTTP/1.1
Host: auth.api.rackspacecloud.com
X-Auth-User: jdoe
X-Auth-Key: a86850deb2742ec3cb41518e26aa2d89
```

Response

If authentication is successful, an HTTP status 204 No Content is returned with three cloud service headers, `X-Server-Management-Url`, `X-Storage-Url`, `X-CDN-Management-Url`, as well as `X-Auth-Token`. An HTTP status of 401 Unauthorized is returned if authentication fails. All operations against Cloud Servers should be performed against the URL specified in `X-Server-Management-Url` (which is dynamic and subject to change) and must include the `X-Auth-Token` header as noted above. The URLs specified in `X-Storage-Url` and `X-CDN-Management-Url` are specific to the Cloud Files product and may be ignored for purposes of interacting with Cloud Servers.

Sample Response

```
HTTP/1.1 204 No Content
Date: Mon, 12 Nov 2007 15:32:21 GMT
Server: Apache
X-Server-Management-Url: https://servers.api.rackspacecloud.com/v1.0/35428
X-Storage-Url: https://storage.clouddrive.com/v1/CloudFS_9c83b-5ed4
X-CDN-Management-Url: https://cdn.clouddrive.com/v1/CloudFS_9c83b-5ed4
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

Authentication tokens are typically valid for 24 hours. Applications should be designed to re-authenticate after receiving a 401 Unauthorized response.

Note that API operation URIs specified throughout this document are relative, this is, they should be appended to the end of the `X-Server-Management-Url` that is returned from the authentication system. For example, in the sample response above, you would list servers by performing a GET against <https://servers.api.rackspacecloud.com/v1.0/345812/servers>.

Request/Response Types

The Cloud Servers 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.

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

Sample JSON Request with Headers

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

{
  "image" : {
    "serverId" : 12,
    "name" : "Just in case"}
}
```

Sample XML Response with Headers

```
HTTP/1.1 200 OKAY
Date: Mon, 12 Nov 2007 15:55:01 GMT
Server: Apache
Content-Length: 185
Content-Type: application/xml; charset=UTF-8

<image xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  id="22" name="Just in case" serverId="12"
  created="2010-10-10T12:00:00Z"
  status="SAVING" progress="0" />
```

Notice, in the above example, that the content type is set to application/json but application/xml is requested via 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.

Sample JSON Request with Extension

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

{
  "image" : {
    "serverId" : 12,
    "name" : "Just in case"}
}
```

Content Compression

Request and response body data may be encoded with gzip compression in order to accelerate interactive performance of API calls and responses. This is controlled using the Accept-Encoding header on the request from the client and indicated by the Content-Encoding header in the server response. Unless the header is explicitly set, encoding defaults to disabled.

Header Type	Name	Value
HTTP/1.1 Request	Accept-Encoding	gzip
HTTP/1.1 Response	Content-Encoding	gzip

Persistent Connections

By default, the API supports persistent connections via HTTP/1.1 keepalives. All connections will be kept alive unless the connection header is set to close.

To prevent abuse, HTTP sessions have a timeout of 20 seconds before being closed.

Note: The server may close the connection at any time and clients should not rely on this behavior.

Paginated Collections

To reduce load on the service, list operations will return a maximum of 1,000 items at a time. To navigate the collection, the parameters limit and offset can be set in the URI (e.g. ?limit=0&offset=0). If an offset is given beyond the end of a list an empty list will be returned. Note that list operations never return itemNotFound (404) faults.

Caching

The Cloud Servers API makes extensive use of caching layers at various tiers of the system. Purging mechanisms exist to ensure that objects served out of cache are accurate and up to date. GETs returning a cached entity return a 203/Cached to signal users that the value is being served out of cache. Additionally, cached entities have the following header set:

Header	Description
Last Modified	Date and time when the entity was last updated in cache.

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 [Unix time](#) (the number of seconds since January 1, 1970, 00:00:00 UTC, not counting leap seconds). If nothing has changed since the *changes-since* time, a 304 Not Modified response 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.rackspacecloud.com/v1.0/224532/servers?changes-since=1244012982> would list all servers that have changed since Wed, 03 Jun 2009 07:09:42 UTC.

Limits

All accounts, by default, have a preconfigured 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.

Note: *If the default limits are too low for your particular application, please contact Rackspace Cloud support to request an increase. All requests require reasonable justification.*

Rate Limits

Verb	URI	RegEx	Default
POST	*	.*	10/min
POST	*/servers	^/servers	25/day
PUT	*	.*	10/min
GET	*changes-since*	changes-since	3/min
DELETE	*	.*	100/min

Notice that we specify rate limits in terms of both a human readable wild-card URI and a machine processable regular expression. The regular expression boundary matcher '^' takes affect after the root URI path. For example, the regular expression `^/servers` would match the bolded portion of the following URI: <https://servers.api.rackspacecloud.com/v1.0/3542812/servers>.

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 25 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 you exceed the thresholds established for your account, a 413 Rate Control HTTP response will be returned with a Reply-After header to notify the client when they can attempt to try again.

Absolute Limits

Limit	Default
Maximum total amount of RAM (GB)	50
Maximum number of shared IP groups	25
Maximum number of members per shared IP group	25

“Maximum total amount of RAM (GB)” limits the number of instances you can run by total aggregate RAM size. For example, with the default limit of 50GB, you may launch a max of 200 256MB cloud servers, or 100 512MB cloud servers, or 50 1GB cloud servers, or 20 2GB + 40 256MB cloud servers, etc. These limits apply to creating as well as resizing servers.

Determining Limits Programmatically

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

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

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400)

Request

This operation does not require a request body.

Sample XML Response

```
<limits xmlns="http://docs.rackspacecloud.com/servers/api/v1.0">
  <rate>
    <limit verb="POST" URI="*" regex=".*"
      value="10" remaining="2"
      unit="MINUTE" resetTime="1244425439" />

    <limit verb="POST" URI="*/servers" regex="^/servers"
      value="25" remaining="24"
      unit="DAY" resetTime="1244511839" />

    <limit verb="PUT" URI="*" regex=".*"
      value="10" remaining="2"
      unit="MINUTE" resetTime="1244425439" />

    <limit verb="GET" URI="*" regex=".*"
      value="3" remaining="3"
      unit="MINUTE" resetTime="1244425439" />

    <limit verb="DELETE" URI="*" regex=".*"
      value="100" remaining="100"
      unit="MINUTE" resetTime="1244425439" />
  </rate>
  <absolute>
    <limit name="maxTotalRAMSize" value="51200" />
    <limit name="maxIPGroups" value="25" />
    <limit name="maxIPGroupMembers" value="25" />
  </absolute>
</limits>
```

Sample JSON Response

```
{
  "limits" : {
    "rate" : [
      {
        "verb" : "POST",
        "URI" : "*",
        "regex" : ".*",
        "value" : 10,
        "remaining" : 2,
        "unit" : "MINUTE",
        "resetTime" : 1244425439
      },
      {
        "verb" : "POST",
        "URI" : "*/servers",
        "regex" : "^/servers",
        "value" : 25,
        "remaining" : 24,
        "unit" : "DAY",
        "resetTime" : 1244511839
      },
      {
        "verb" : "PUT",
        "URI" : "*",
        "regex" : ".*",
        "value" : 10,
        "remaining" : 2,
        "unit" : "MINUTE",
        "resetTime" : 1244425439
      },
      {
        "verb" : "GET",
        "URI" : "*",
        "regex" : ".*",
        "value" : 3,
        "remaining" : 3,
        "unit" : "MINUTE",
        "resetTime" : 1244425439
      },
      {
        "verb" : "DELETE",
        "URI" : "*",
        "regex" : ".*",
        "value" : 100,
        "remaining" : 100,
        "unit" : "MINUTE",
        "resetTime" : 1244425439
      }
    ],
    "absolute" : {
      "maxTotalRAMSize" : 51200,
      "maxIPGroups" : 50,
      "maxIPGroupMembers" : 25
    }
  }
}
```

API Version

The Cloud Servers API uses a URI versioning scheme. The first element of the path contains the target version identifier (e.g. `https://servers.api.rackspacecloud.com/v1.0/...`) All requests (except to query for version – see below) *must* contain a target version. New features and functionality that do not break API-compatibility will be introduced in the current version of the API and the URI will remain unchanged. Any features or functionality changes that would necessitate a break in API-compatibility will require a new version, which will result in the URI version being updated accordingly. When new API versions are released, Rackspace will work with developers and partners to ensure there is adequate time to migrate to the new version before older versions are deprecated.

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 truncated) returned from the authentication system.

```
GET HTTP/1.1
Host: servers.api.rackspacecloud.com
```

Normal Response Code(s): 200, 203

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

Request

This operation does not require a request body

Sample XML Response

```
<versions xmlns="http://docs.rackspacecloud.com/servers/api/v1.0">
  <version status="CURRENT" id="v1.0"/>
  <version status="BETA" id="v1.1"/>
</versions>
```

Sample JSON Response

```
{ "versions": [{
  "status": "CURRENT",
  "id": "v1.0"},
  {
  "status": "BETA",
  "id": "v1.1"
}] }
```

You can also obtain additional information about a specific version by performing a GET on the base version URL (e.g. <https://servers.api.rackspacecloud.com/v1.0>).

```
GET HTTP/1.1
Host: servers.api.rackspacecloud.com/v1.0
```

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400)

Request

This operation does not require a request body

Sample XML Response

```
<version xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  id="v1.0"
  status="CURRENT"
  docURL="http://docs.rackspacecloud.com/servers/api/v1.0/cs-
devguide-20090714.pdf"
  wadl="https://docs.rackspacecloud.com/servers/api/v1.0/appli
cation.wadl"/>
```

Sample JSON Response

```
{
  "version": {
    "status": "CURRENT",
    "id": "v1.0",
    "docURL" : "http://docs.rackspacecloud.com/servers/api/v1.0/cs-
devguide-20090714.pdf ",
    "wadl" : "https://docs.rackspacecloud.com/servers/api
/v1.0/application.wadl"
  }
}
```

Note that the detailed version response contains a pointer 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: While we strive to produce accurate documentation, errors may exist (please let us know if you find one!). If a discrepancy exists between the two specifications, the WADL is authoritative as it contains the most accurate and up-to-date description of the API service.

Faults

When an error occurs 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.

Sample XML Error Response

```
<cloudServersFault
  xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  code="500">
  <message>Fault!</message>
  <details>Error Details...</details>
</cloudServersFault>
```

Sample JSON Response

```
{
  "cloudServersFault" : {
    "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 useful information for tracking down an error (e.g a stack trace). The detail section may or may not be appropriate for display to an end user.

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

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

Sample XML Error Response

```
<itemNotFound
  xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  code="404">
  <message>Not Found</message>
</itemNotFound>
```

Sample JSON Response

```
{
  "itemNotFound" : {
    "code" : 404,
    "message" : "Not Found"
  }
}
```

From an XML schema perspective, all API faults are extensions of the base fault type `CloudServersAPIFault`. When working with a system that binds XML to actual classes (such as JAXB), one should be capable of using `CloudServersAPIFault` 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 `replyAfter` attribute that contains the content of the `Reply-After` header in XML Schema 1.0 date/time format.

Sample XML Error Response

```
<overLimit
  xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  code="413"
  retryAfter="2010-08-01T00:00:00Z">
  <message>OverLimit Please Retry...</message>
</overLimit>
```

Sample JSON Response

```
{
  "overLimit" : {
    "code" : 413,
    "message" : "OverLimit Please Retry...",
    "retryAfter" : "2010-08-01T00:00:00Z"
  }
}
```

API Operations

Servers

List Servers

Verb	URI	Description
GET	/servers	List all servers (IDs and names only)
GET	/servers/detail	List all servers (all details)

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), overLimit (413)

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

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

Note: *hostId* is unique PER ACCOUNT and is not globally unique.

Request

This operation does not require a request body.

Sample XML Response (detail)

```
<servers xmlns="http://docs.rackspacecloud.com/servers/api/v1.0">

  <server id="1234" name="sample-server"
    imageId="2" flavorId="1"
    status="BUILD" progress="60"
    hostId="e4d909c290d0fb1ca068ffaddf22cbd0"
    >
    <metadata>
      <meta key="Server Label">Web Head 1</meta>
      <meta key="Image Version">2.1</meta>
    </metadata>
    <addresses>
      <public>
        <ip addr="67.23.10.132"/>
        <ip addr="67.23.10.131"/>
      </public>
      <private>
        <ip addr="10.176.42.16"/>
      </private>
    </addresses>
  </server>

  <server id="5678" name="sample-server2"
    imageId="2" flavorId="1"
    status="ACTIVE"
    hostId="9e107d9d372bb6826bd81d3542a419d6">
    <metadata>
      <meta key="Server Label">DB 1</meta>
    </metadata>
    <addresses>
      <public>
        <ip addr="67.23.10.133"/>
      </public>
      <private>
        <ip addr="10.176.42.17"/>
      </private>
    </addresses>
  </server>

</servers>
```

Sample JSON Response (detail)

```
{
  "servers" : [
    {
      "id" : 1234,
      "name" : "sample-server",
      "imageId" : 2,
      "flavorId" : 1,
      "hostId" : "e4d909c290d0fb1ca068ffaddf22cbd0",
      "status" : "BUILD",
      "progress" : 60,
      "addresses" : {
        "public" : [
          "67.23.10.132",
          "67.23.10.131"
        ],
        "private" : [
          "10.176.42.16"
        ]
      },
      "metadata" : {
        "Server Label" : "Web Head 1",
        "Image Version" : "2.1"
      }
    },
    {
      "id" : 5678,
      "name" : "sample-server2",
      "imageId" : 2,
      "flavorId" : 1,
      "hostId" : "9e107d9d372bb6826bd81d3542a419d6",
      "status" : "ACTIVE",
      "addresses" : {
        "public" : [
          "67.23.10.133"
        ],
        "private" : [
          "10.176.42.17"
        ]
      },
      "metadata" : {
        "Server Label" : "DB 1"
      }
    }
  ]
}
```

Create Server

Verb	URI	Description
POST	/servers	Create a new server

Normal Response Code(s): 202

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badMediaType(415), badRequest (400), serverCapacityUnavailable (503), overLimit (413)

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 `/server/id`, which will return a progress attribute (0-100% completion).

A password will be randomly generated for you and returned in the response object. For security reasons, it will not be returned in subsequent GET calls against a given server ID.

Custom cloud server metadata can also be supplied at launch time. This metadata is stored in the API system where it is retrievable by querying the API for server status. The maximum size of the metadata key and value is each 255 bytes and the maximum number of key-value pairs that can be supplied per server is 5.

You may further customize a cloud server by injecting data into the file system of the cloud server itself. This is useful, for example, for inserting ssh keys, setting configuration files, or storing data that you want to retrieve from within the instance itself. It is intended to provide a minimal amount of launch-time personalization. If significant customization is required, a custom image should be created. The max size of the file path data is 255 bytes while the max size of the file contents is 10KB. Note that the file contents should be encoded as a Base64 string and the 10KB limit refers to the number of bytes in the decoded data not the number of characters in the encoded data. The maximum number of file path/content pairs that can be supplied is 5. Any existing files that match the specified file will be renamed to include the extension **bak** followed by a time stamp. For example, the file `/etc/passwd` will be backed up as `/etc/passwd.bak.1246036261.5785`. All files will have root and the root group as owner and group owner, respectively and will allow user and group read access only (`-r--r-----`).

Servers in the same shared IP group can share public IPs for various high availability and load balancing configurations. To launch an HA server, include the optional `sharedIpGroupId` element and the server will be launched into that shared IP group.

If you intend to use a shared IP on the server being created and have no need for a separate public IP address, you may launch the server into a shared IP group and specify an IP address from that

shared IP group to be used as its public IP. You can accomplish this by specifying the public shared IP address in your request. This is optional and is only valid if *sharedIpGroupId* is also supplied.

Note: *sharedIpGroupId* is an optional parameter and for optimal performance, should ONLY be specified when intending to share IPs between servers.

Sample XML Request

```
<server xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  name="new-server-test" imageId="2" flavorId="1">
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <personality>
    <file path="/etc/banner.txt">
      ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
      dCBtb3ZlcyBpbjBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
      IGF0IHN1Y2ggYSBzcGVlZC4uLk10IGZlZWxzIGFuIGltcHVz
      c2lvbi4uLnRoXMGaXMgdGhlIHBSYWNlIHRvIGdvIG5vdy4g
      QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
      ZSBwYXR0ZXJucyBiZWpibmQgYWxsIGNsb3VkcycwYmV5b25k
      dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmCB5b3Vy
      c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
      b25zLiINCg0KLVJpY2hhcmQgQmFjaA==
    </file>
  </personality>
</server>
```


Sample JSON Request

```
{
  "server" : {
    "name" : "new-server-test",
    "imageId" : 2,
    "flavorId" : 1,
    "metadata" : {
      "My Server Name" : "Apache1"
    },
    "personality" : [
      {
        "path" : "/etc/banner.txt",
        "contents" :
"ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBpdCBtb3ZlcyBpbiBqdXN0IHN1Y2
ggYSBkaXJlY3Rpb24gYW5kIGF0IHN1Y2ggYSBzcGVlZC4uLk10IGZlZWxzIGFuIGltcHVsc2lvb
i4uLnRoXMGaXMgdGhlIHBSYWNlIHRvIGdvIG5vdy4gQnV0IHRoZSBza3kga25vd3MgdGhlIHJl
YXNvbnMgYW5kIHRoZSBwYXR0ZXJucyBiZWwhpbmQgYWxsIGNsb3VkcycwYmV5b25kIGhvcml6b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
      }
    ]
  }
}
```

Sample XML Response

```
<server xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  id="1235" name="new-server-test"
  imageId="2" flavorId="1"
  hostId="e4d909c290d0fb1ca068ffaddf22cbd0"
  progress="0" status="BUILD"
  adminPass="GFf1j9aP">
  <metadata>
    <meta key="My Server Name">Apache1</meta>
  </metadata>
  <addresses>
    <public>
      <ip addr="67.23.10.138"/>
    </public>
    <private>
      <ip addr="10.176.42.19"/>
    </private>
  </addresses>
</server>
```

Sample JSON Response

```
{
  "server" : {
    "id" : 1235,
    "name" : "new-server-test",
    "imageId" : 2,
    "flavorId" : 1,
    "hostId" : "e4d909c290d0fb1ca068ffaddf22cbd0",
    "progress" : 0,
    "status" : "BUILD",
    "adminPass" : "GFf1j9aP",
    "metadata" : {
      "My Server Name" : "Apache1"
    },
    "addresses" : {
      "public" : [
        "67.23.10.138"
      ],
      "private" : [
        "10.176.42.19"
      ]
    }
  }
}
```

Get Server Details

Verb	URI	Description
GET	/servers/ <i>id</i>	List details of the specified server

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), itemNotFound (404), overLimit (413)

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

Request

This operation does not require a request body.

Sample XML Response

```
<server xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  id="1234" name="sample-server"
  imageId="2" flavorId="1"
  status="BUILD" progress="60"
  hostId="e4d909c290d0fb1ca068ffaddf22cbd0"
>
<metadata>
  <meta key="Server Label">Web Head 1</meta>
  <meta key="Image Version">2.1</meta>
</metadata>
<addresses>
  <public>
    <ip addr="67.23.10.132"/>
    <ip addr="67.23.10.131"/>
  </public>
  <private>
    <ip addr="10.176.42.16"/>
  </private>
</addresses>
</server>
```

Sample JSON Response

```
{
  "server" : {
    "id" : 1234,
    "name" : "sample-server",
    "imageId" : 2,
    "flavorId" : 1,
    "hostId" : "e4d909c290d0fb1ca068ffaddf22cbd0",
    "status" : "BUILD",
    "progress" : 60,
    "addresses" : {
      "public" : [
        "67.23.10.132",
        "67.23.10.131"
      ],
      "private" : [
        "10.176.42.16"
      ]
    },
    "metadata" : {
      "Server Label" : "Web Head 1",
      "Image Version" : "2.1"
    }
  }
}
```

Update Server Name / Administrative Password

Verb	URI	Description
PUT	/servers/ <i>id</i>	Update the specified server's name and/or administrative password

Normal Response Code(s): 202

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), badMediaType(415), itemNotFound (404), buildInProgress (409), overLimit (413)

Status Transition: ACTIVE → PASSWORD → ACTIVE

This operation allows you to update the name of the server and/or change the administrative password. This operation changes the name of the server in the Cloud Servers system and does not change the server host name itself.

Sample XML Request

```
<server xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  name="new-server-test" adminPass="newPassword" />
```

Sample JSON Request

```
{
  "server" :
  {
    "name" : "new-server-test",
    "adminPass" : "newPassword"
  }
}
```

Response

This operation does not contain a response body.

Delete Server

Verb	URI	Description
DELETE	/servers/ <i>id</i>	Terminate the specified server

Normal Response Code(s): 202

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), itemNotFound (404), buildInProgress (409), overLimit (413)

Status Transition: ACTIVE → DELETED
SUSPENDED → DELETED

This operation deletes a cloud server instance from the system.

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

Request

This operation does not require a request body.

Response

This operation does not contain a response body.

Server Addresses

List Addresses

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

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), overLimit (413)

Request

This operation does not require a request body.

Sample XML Response

```
<addresses xmlns="http://docs.rackspacecloud.com/servers/api/v1.0">
  <public>
    <ip addr="67.23.10.132"/>
    <ip addr="67.23.10.131"/>
  </public>
  <private>
    <ip addr="10.176.42.16"/>
  </private>
</addresses>
```

Sample JSON Response

```
{
  "addresses" : {
    "public" : [
      "67.23.10.132",
      "67.23.10.131"
    ],
    "private" : [
      "10.176.42.16"
    ]
  }
}
```

List Public Addresses

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

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), overLimit (413)

Request

This operation does not require a request body.

Sample XML Response

```
<public xmlns="http://docs.rackspacecloud.com/servers/api/v1.0">
  <ip addr="67.23.10.132"/>
  <ip addr="67.23.10.131"/>
</public>
```

Sample JSON Response

```
{
  "public" : [
    "67.23.10.132",
    "67.23.10.131"
  ]
}
```

List Private Addresses

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

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), overLimit (413)

Request

This operation does not require a request body.

Sample XML Response

```
<private xmlns="http://docs.rackspacecloud.com/servers/api/v1.0">
  <ip addr="10.176.42.16"/>
</private>
```

Sample JSON Response

```
{
  "private" : [
    "10.176.42.16"
  ]
}
```

Share an IP Address

Verb	URI	Description
PUT	/servers/id/ips/public/address	Share an IP address to the specified server

Normal Response Code(s): 202**Error Response Code(s):** cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), badMediaType(415), itemNotFound (404), overLimit (413)

Status Transition: ACTIVE → SHARE_IP → ACTIVE (if configureServer is true)
 ACTIVE → SHARE_IP_NO_CONFIG → ACTIVE

This operation shares an IP from an existing server in the specified shared IP group to another specified server in the same group. By default, the operation modifies cloud network restrictions to allow IP traffic for the given IP to/from the server specified, but does not bind the IP to the server itself. A heartbeat facility (e.g. keepalived) can then be used within the servers to perform health checks and manage IP failover. If the configureServer attribute is set to true, the server is configured with the new address, though the address is not enabled. Note that configuring the server does require a reboot.

Sample XML Request

```
<shareIp xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  sharedIpGroupId="1234" configureServer="true" />
```


Sample JSON Request

```
{
  "shareIp" : {
    "sharedIpGroupId" : 1234,
    "configureServer" : true
  }
}
```

Response

This operation does not return a response body.

Unshare an IP Address

Verb	URI	Description
DELETE	/servers/ <i>id</i> /ips/public/ <i>address</i>	Remove a shared IP address from the specified server

Normal Response Code(s): 202

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), badMediaType(415), itemNotFound (404), overLimit (413)

Status Transition: ACTIVE → DELETE_IP → ACTIVE

This operation removes a shared IP address from the specified server.

Request

This operation does not contain a request body.

Response

This operation does not return a response body.

Server Actions

Reboot Server

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

Normal Response Code(s): 202

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), badMediaType(415), itemNotFound (404), buildInProgress (409), overLimit (403)

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

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

Sample XML Request

```
<reboot xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  type="HARD"/>
```

Sample JSON Request

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

Response

This operation does not return a response body.

Rebuild Server

Verb	URI	Description
POST	/servers/ <i>id</i> /action	Rebuild the specified server

Normal Response Code(s): 202

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), badMediaType(415), itemNotFound (404), buildInProgress (409), serverCapacityUnavailable (503), overLimit (413)

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

The rebuild function removes all data on the server and replaces it with the specified image. *serverId* and IP addresses will remain the same. *imageId* is an optional argument. If it is not specified, the server is rebuilt with the original *imageId*.

Sample XML Request

```
<rebuild xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  imageId="2"/>
```

Sample JSON Request

```
{
  "rebuild" : {
    "imageId" : 2
  }
}
```

Response

This operation does not return a response body.

Resize Server

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

Normal Response Code(s): 202

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

Status Transition: ACTIVE → QUEUE_RESIZE → PREP_RESIZE → VERIFY_RESIZE
ACTIVE → QUEUE_RESIZE → ACTIVE (on error)

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

Sample XML Request

```
<resize xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
        flavorId="3"/>
```

Sample JSON Request

```
{
  "resize" : {
    "flavorId" : 3
  }
}
```

Response

This operation does not return a response body.

Confirm Resized Server

Verb	URI	Description
POST	/servers/ <i>id</i> /action	Confirm a pending resize action

Normal Response Code(s): 204

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

Status Transition: VERIFY_RESIZE → ACTIVE

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

Sample XML Request

```
<confirmResize xmlns="http://docs.rackspacecloud.com/servers/api/v1.0" />
```

Sample JSON Request

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

Response

This operation does not return a response body.

Revert Resized Server

Verb	URI	Description
POST	/servers/ <i>id</i> /action	Cancel and revert a pending resize action

Normal Response Code(s): 202

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), itemNotFound (404), buildInProgress (409), serverCapacityUnavailable (503), overLimit (413), resizeNotAllowed (403)

Status Transition: VERIFY_RESIZE → ACTIVE

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

Sample XML Request

```
<revertResize xmlns="http://docs.rackspacecloud.com/servers/api/v1.0" />
```

Sample JSON Request

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

Flavors

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.

List Flavors

Verb	URI	Description
GET	/flavors	List available flavors (IDs and names only)
GET	/flavors/detail	List available flavors (all details)

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400)

This operation will list all available flavors with details.

Request

This operation does not require a request body.

Sample XML Response (detail)

```
<flavors xmlns="http://docs.rackspacecloud.com/servers/api/v1.0">
  <flavor id="1" name="256 MB Server"
    ram="256" disk="10" />
  <flavor id="2" name="512 MB Server"
    ram="512" disk="20" />
</flavors>
```

Sample JSON Response (detail)

```
{
  "flavors" : [
    {
      "id" : 1,
      "name" : "256 MB Server",
      "ram" : 256,
      "disk" : 10
    },
    {
      "id" : 2,
      "name" : "512 MB Server",
      "ram" : 512,
      "disk" : 20
    }
  ]
}
```

Get Flavor Details

Verb	URI	Description
GET	/flavors/ <i>id</i>	List details of the specified flavor

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), itemNotFound (404)

This operation returns details of the specified image.

Request

This operation does not require a request body.

Sample XML Response

```
<flavor id="1" name="256 MB Server"
      ram="256" disk="10" />
```


Sample JSON Response

```
{
  "flavor" : {
    "id" : 1,
    "name" : "256 MB Server",
    "ram" : 256,
    "disk" : 10
  }
}
```

Images

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

List Images

Verb	URI	Description
GET	/images	List available images (IDs and names only)
GET	/images/detail	List available images (all details)

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400)

This operation will list all images visible by the account.

Status Transitions

In-flight images will have the *status* attribute set to SAVING and the conditional *progress* element (0-100% completion) will also be returned. Other possible values for the *status* attribute include: UNKNOWN, PREPARING, ACTIVE QUEUED, FAILED. Images with an ACTIVE status are available for install.

Request

This operation does not require a request body.

Sample XML Response (detail)

```
<images xmlns="http://docs.rackspacecloud.com/servers/api/v1.0">
  <image id="2" name="CentOS 5.2"
    updated="2010-10-10T12:00:00Z"
    created="2010-08-10T12:00:00Z"
    status="ACTIVE"
  />
  <image id="743" name="My Server Backup"
    serverId="12"
    updated="2010-10-10T12:00:00Z"
    created="2010-08-10T12:00:00Z"
    status="SAVING" progress="80"
  />
</images>
```

Sample JSON Response (detail)

```
{
  "images" : [
    {
      "id" : 2,
      "name" : "CentOS 5.2",
      "updated" : "2010-10-10T12:00:00Z",
      "created" : "2010-08-10T12:00:00Z",
      "status" : "ACTIVE"
    },
    {
      "id" : 743,
      "name" : "My Server Backup",
      "serverId" : 12,
      "updated" : "2010-10-10T12:00:00Z",
      "created" : "2010-08-10T12:00:00Z",
      "status" : "SAVING",
      "progress" : 80
    }
  ]
}
```

Create an Image

Verb	URI	Description
POST	/images	Create a new image

Normal Response Code(s): 202

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

Image Status Transition: QUEUED → PREPARING → SAVING → ACTIVE
QUEUED → PREPARING → SAVING → FAILED (on error)

This operation creates a new image for the given server ID. Once complete, a new image will be available that can be used to rebuild or create servers. Specifying the same image name as an existing custom image replaces the image. The image creation status can be queried by performing a GET on `/images/id` and examining the *status* and *progress* attributes.

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

Sample XML Request

```
<image xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  name="Just in case" serverId="12" />
```

Sample JSON Request

```
{
  "image" : {
    "serverId" : 12,
    "name" : "Just in case"
  }
}
```

Sample XML Response

```
<image xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  id="22" name="Just in case"
  created="2010-10-10T12:00:00Z"
  status="SAVING" progress="0" serverId="12"/>
```

Sample JSON Response

```
{
  "image" : {
    "id" : 22,
    "serverId" : 12,
    "name" : "Just in case",
    "created" : "2010-10-10T12:00:00Z",
    "status" : "SAVING",
    "progress" : 0
  }
}
```

Get Image Details

Verb	URI	Description
GET	/images/ <i>id</i>	List details of the specified image

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), itemNotFound (404)

This operation returns details of the specified image.

Request

This operation does not require a request body.

Sample XML Response

```
<image xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  id="2" name="CentOS 5.2"
  serverId="12"
  updated="2010-10-10T12:00:00Z"
  created="2010-08-10T12:00:00Z"
  status="SAVING" progress="80" />
```

Sample JSON Response

```
{
  "image" : {
    "id" : 2,
    "name" : "CentOS 5.2",
    "serverId" : 12,
    "updated" : "2010-10-10T12:00:00Z",
    "created" : "2010-08-10T12:00:00Z",
    "status" : "SAVING",
    "progress" : 80
  }
}
```

Backup Schedules

In addition to creating images on-demand, you may also schedule periodic (daily and weekly) images via a backup schedule. The daily and weekly images are triggered automatically based on the backup schedule established. The days/times specified for the backup schedule are targets and actual start and completion times may vary based on other activity in the system. All backup times are in GMT.

Weekly Backup Schedule

Value	Day
DISABLED	Weekly backup disabled
SUNDAY	Sunday
MONDAY	Monday
TUESDAY	Tuesday
WEDNESDAY	Wednesday
THURSDAY	Thursday
FRIDAY	Friday
SATURDAY	Saturday

Daily Backup Schedule

Value	Hour Range
DISABLED	Daily backups disabled
H_0000_0200	0000-0200
H_0200_0400	0200-0400
H_0400_0600	0400-0600
H_0600_0800	0600-0800
H_0800_1000	0800-1000
H_1000_1200	1000-1200
H_1200_1400	1200-1400
H_1400_1600	1400-1600
H_1600_1800	1600-1800
H_1800_2000	1800-2000
H_2000_2200	2000-2200
H_2200_0000	2200-0000

List Backup Schedules

Verb	URI	Description
GET	/servers/id/backup_schedule	List the backup schedule for the specified server

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), itemNotFound (404)

This operation lists the backup schedule for the specified server. If a server does not have a backup schedule enabled, the *status* element will be “disabled” and weekly/daily elements will not be provided in the response.

Request

This operation does not require a request body.

Sample XML Response

```
<backupSchedule xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  enabled="true" weekly="THURSDAY" daily="H_0400_0600" />
```

Sample JSON Response

```
{
  "backupSchedule" : {
    "enabled" : true,
    "weekly" : "THURSDAY",
    "daily" : "H_0400_0600"
  }
}
```

Create / Update Backup Schedule

Verb	URI	Description
POST	/servers/id/backup_schedule	Enable/update the backup schedule for the specified server

Normal Response Code(s): 204

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

This operation creates a new backup schedule or updates an existing backup schedule for the specified server.

Sample XML Request

```
<backupSchedule xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
    enabled="true" weekly="THURSDAY" daily="H_0400_0600" />
```

Sample JSON Request

```
{
  "backupSchedule" : {
    "enabled" : true,
    "weekly" : "THURSDAY",
    "daily" : "H_0400_0600"
  }
}
```

Response

This operation does not return a response body.

Delete Backup Schedule

Verb	URI	Description
DELETE	/servers/ <i>id</i> /backup_schedule	Delete backup schedule for the specified server

Normal Response Code(s): 204

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), itemNotFound (404), buildInProgress (409), serverCapacityUnavailable (503), backupOrResizeInProgress(409)

This operation deletes the backup schedule for the specified server.

Request

This operation does not require a request body.

Response

This operation does not return a response body.

Shared IP Groups

A shared IP group is a collection of servers that can share IPs with other members of the group. Any server in a group can share one or more public IPs with any other server in the group. With the exception of the first server in a shared IP group, servers must be launched into shared IP groups. A server may only be a member of one shared IP group.

List Shared IP Groups

Verb	URI	Description
GET	/shared_ip_groups	List shared ip groups (IDs and names only)
GET	/shared_ip_groups/detail	List shared ip groups (all details)

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400)

This operation provides a list of shared IP groups associated with your account.

Request

This operation does not require a request body.

Sample XML Response (detail)

```
<sharedIpGroups
  xmlns="http://docs.rackspacecloud.com/servers/api/v1.0">
  <sharedIpGroup id="1234" name="Shared IP Group 1">
    <servers>
      <server id="422" />
      <server id="3445" />
    </servers>
  </sharedIpGroup>
  <sharedIpGroup id="5678" name="Shared IP Group 2">
    <servers>
      <server id="23203"/>
      <server id="2456" />
      <server id="9891" />
    </servers>
  </sharedIpGroup>
</sharedIpGroups>
```

Sample JSON Request (detail)

```
{
  "sharedIpGroups" : [
    {
      "id" : 1234,
      "name" : "Shared IP Group 1",
      "servers" : [422, 3445]
    },
    {
      "id" : 5678,
      "name" : "Shared IP Group 2",
      "servers" : [23203, 2456, 9891]
    }
  ]
}
```

Create Shared IP Group

Verb	URI	Description
POST	/shared_ip_groups	Create a new shared ip group

Normal Response Code(s): 201

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), badMediaType(415), overLimit (413)

This operation creates a new shared IP group. Please note, all responses to requests for shared_ip_groups return an array of servers. However, on a create request, the shared IP group can be created empty or can be initially populated with a single server. Submitting a create request with a sharedIpGroup that contains an array of servers will generate a badRequest (400) fault.

Sample XML Request

```
<sharedIpGroup
  xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  name="Shared IP Group 1">
  <server id="422"/>
</sharedIpGroup>
```

Sample JSON Request

```
{
  "sharedIpGroup" : {
    "name" : "Shared IP Group 1",
    "server" : 422
  }
}
```

Sample XML Response

```
<sharedIpGroup
  xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  id="1234" name="Shared IP Group 1">
  <servers>
    <server id="422"/>
  </servers>
</sharedIpGroup>
```

Sample JSON Response

```
{
  "sharedIpGroup" : {
    "id" : 1234,
    "name" : "Shared IP Group 1",
    "servers" : [422]
  }
}
```

Get Shared IP Group Details

Verb	URI	Description
GET	/shared_ip_groups/ <i>id</i>	List details of the specified shared IP group

Normal Response Code(s): 200, 203

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), itemNotFound (404)

This operation returns details of the specified shared IP group.

Request

This operation does not require a request body.

Sample XML Response

```
<sharedIpGroup
  xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
  id="1234" name="Shared IP Group 1">
  <servers>
    <server id="422"/>
    <server id="3445"/>
  </servers>
</sharedIpGroup>
```

Sample JSON Response

```
{
  "sharedIpGroup" : {
    "id" : 1234,
    "name" : "Shared IP Group 1",
    "servers" : [
      422,
      3445
    ]
  }
}
```

Delete Shared IP Group

Verb	URI	Description
DELETE	/shared_ip_groups/ <i>id</i>	Delete the specified shared IP group

Normal Response Code(s): 204

Error Response Code(s): cloudServersFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest (400), itemNotFound (404)

This operation deletes the specified shared IP group. This operation will ONLY succeed if 1) there are no active servers in the group (i.e. they have all been terminated) or 2) no servers in the group are actively sharing IPs.

Request

This operation does not require a request body.

Response

This operation does not contain a response body.

Appendix

HTTP/1.1 Response Codes

The Cloud Servers API may return any of the HTTP/1.1 response codes defined by [RFC-2616 Section 10](#). The API uses the following return codes as described:

Code	Description
200 OK	Fresh authoritative content returned.
201 Created	A POST was accepted, and completed synchronously. The body of the response will contain the id of the created entity.
202 Accepted	An asynchronous operation was started. Where possible, the body of the response will contain an id that can be used to poll the status of the request for completion.
203 Cached	Cached content was returned. If possible, the server will return a Retry-After header indicating when fresh content will be available.
204 No Content	The request was accepted and processed, but did not need to return a response body.
304 Not Modified	Indicates that the requested resource's status has not changed since the specified If-Modified-Since date.
400 Bad Request	There was a syntax error in the request. Do not repeat the request without adjusting the input.
401 Unauthorized	The X-Auth-Token is not valid or has expired. Re-authenticate to obtain a fresh token.
403 Forbidden	Access is denied for the given request. Check your X-Auth-Token header. The token may have expired.
404 Not Found	The server has not found anything matching the Request URI.
413 Request Entity Too Large	The server is refusing to process a request because the request entity is larger than the server is willing or able to process. The server MAY close the connection to prevent the client from continuing the request. If the condition is temporary, the server will include a Retry-After header field to indicate that it is temporary and after what time the client MAY try again. _
413 Rate Control	The server is refusing to process the request because the client request rate exceeds a configured limit for the given account and action type. The server will include a Retry-After header field to indicate that it is temporary and after what time the client MAY try again.
500 Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.

Cloud Servers API Roadmap

The following capabilities are planned for the subsequent Cloud Servers API release:

- **Backup Integration with Cloud Files** - Backups are stored in your Cloud Files account. You make take unlimited on-demand images and pricing is completely utility based - you pay only for the space you consume on Cloud Files. Backups can be downloaded and are also decoupled from servers so they persist after a server is terminated.
- **Custom Image Upload** - Currently, you can customize an image by starting with a Rackspace provided default, installing applications, making desired configuration changes, and creating images. In this release, we enable you to upload your own custom images to Cloud Files and register them with Cloud Servers for deployment.
- **Image Sharing** - You can mark images as public or take advantage of image access groups to delegate image access to a specific group of Cloud Servers users.

Additional Resources

The official support channels (phone, chat, email, forums, and knowledge base articles) for Cloud Servers are available at the Rackspace Cloud website: <http://www.rackspacecloud.com>.

Interested users can also follow updates/announcements via twitter at <http://www.twitter.com/rackcloud>.