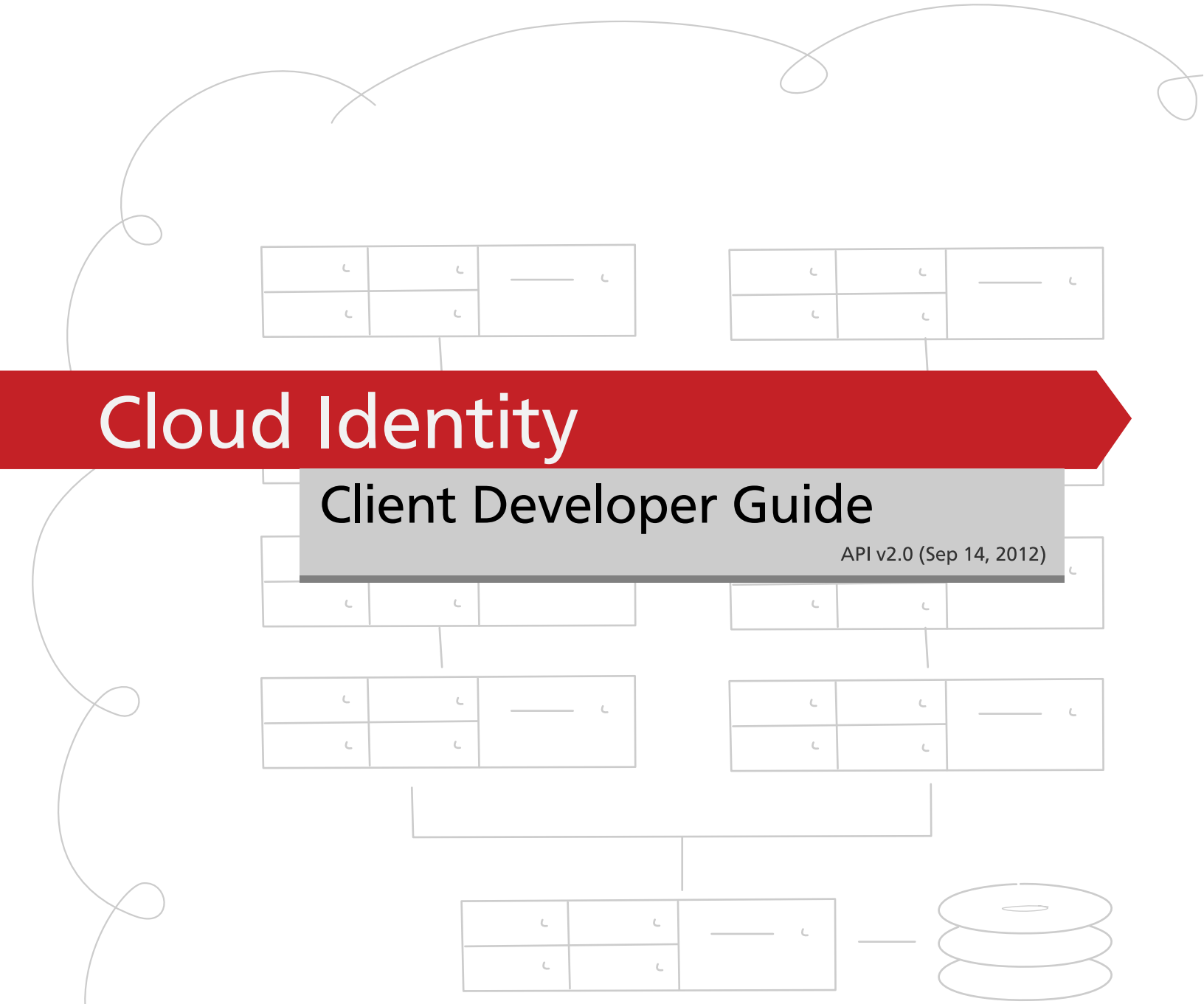


Cloud Identity

Client Developer Guide

API v2.0 (Sep 14, 2012)

rackspace HOSTING

docs.rackspace.com/api

Cloud Identity Client Developer Guide

API v2.0 (2012-09-14)

This document is intended for software developers interested in developing applications that utilize Rackspace's implementation of OpenStack's Keystone Identity Service for authentication. This document also includes details on how to integrate services with Rackspace's implementation of OpenStack's Keystone Identity Service. The document is for informational purposes only and is provided "AS IS."

Table of Contents

List of Figures

List of Tables

List of Examples

1. Overview

This document describes Rackspace's implementation of the OpenStack Keystone Identity Service v2.0. We have named our Keystone implementation the Rackspace Cloud Identity Service v2.0; the identity service performs authentication, generating a token in response to valid credentials, and identification, matching users and their roles. Earlier versions of this API, v1.0 and v1.1, support only authentication and are named Rackspace Cloud Authentication Service; all versions of the API allow clients to obtain tokens that can be used to access resources in the Rackspace Cloud, but only v2.0 is an implementation of OpenStack's Keystone Identity Service.

While each version of the API implements different features, tokens created by any version can be recognized as valid by any other version. For example, if you use v1.1 to generate a token, you can use that token in calls to v2.0.

1.1. QuickStart

To authenticate, submit a `POST /v2.0/tokens` request, presenting valid Rackspace customer *credentials* in the message body to a Rackspace *authentication endpoint*.

GET YOUR CREDENTIALS

You can use either of two sets of credentials:

- your `username` and `password`
- your `username` and `API key`

Your `username` and `password` are the ones you use to login to the Rackspace Cloud Control Panel at http://mycloud.rackspace.com/. Once you are logged in, you can use the Cloud Control Panel to obtain your `API key`.

Important

Multiple users (sub-users) can be created for an account but, as explained in Section 4.1.4, "Add User" [45], users created via the API do not have access to the control panel. Control panel access for these users is coming soon.

CHOOSE YOUR AUTHENTICATION ENDPOINT

Use the authentication endpoint for the region in which your account is based:

- US-based accounts authenticate through https://identity.api.rackspacecloud.com/v2.0/.
- UK-based accounts authenticate through https://lon.identity.api.rackspacecloud.com/v2.0/.

SEND YOUR CREDENTIALS TO YOUR AUTHENTICATION ENDPOINT

If you know your credentials and your authentication endpoint, and you can issue a `POST /v2.0/tokens` request in an API call, you have all the basic information you need to use the Rackspace Cloud Identity Service.

You can use cURL to try the authentication process in two steps: get a token; send the token to a service.

1. Get an authentication token by providing your username and either your API key or your password. Here are examples of both approaches:

   You can request a token by providing your username and your API key.

   ```
   curl -X POST https://identity.api.rackspacecloud.com/v2.0/tokens -d
   '{ "auth":{ "RAX-KSKEY:apiKeyCredentials":{ "username":"theUserName",
    "apiKey":"00a00000a000a0000000a000a00aaa0a" } } }' -H "Content-type:
    application/json"
   ```

   You can request a token by providing your username and your password.

   ```
   curl -X POST https://identity.api.rackspacecloud.com/v2.0/tokens -d
   '{"auth":{"passwordCredentials":
   {"username":"theUserName","password":"thePassword"}}}' -H "Content-type:
    application/json"
   ```

   Successful authentication returns a token which you can use as evidence that your identity has already been authenticated. To use the token, pass it to other services as an `X-Auth-Token` header.

   Authentication also returns a service catalog, listing the endpoints you can use for Cloud services.

2. Use the authentication token to send a GET to a service you would like to use. Here is an example of passing an authentication token to the Cloud Files service, using the Cloud Files service catalog endpoint that was returned along with the token.

   You can use a token and a service endpoint to tell a service that your credentials are valid.

   ```
   curl -X GET https://storage101.dfw1.clouddrive.com/v1/MossoCloudFS_aaaaaaaa-
   bbbb-cccc-dddd-eeeeeeee
   -H 'X-Auth-Token:11111111-aaaa-2222-bbbb-3333cccc4444' --verbose
   ```

The rest of this API Developer Guide provides reference and background information, including sample requests and responses. To learn more about the Cloud Identity Service and this API, two good places to begin are with the fundamental ideas explained in Chapter 2, *Concepts* [7] and the annotated request and response in Section 3.2, “Sample Authentication Request and Response” [10]. To see examples of authentication using several kinds of credentials, read Section 4.3.1, “Authenticate” [57].

Tip

For links to language binding examples you can adapt to work with the Cloud Identity service, visit https://github.com/rackspace.

1.2. Intended Audience

This document is intended for software developers interested in developing applications that utilize the Keystone Identity Service API for authentication. The Rackspace Cloud Identity Service, also known as the authentication service or "Auth", allows these developers

to obtain an authentication token and a list of regional service endpoints to the various services available in the cloud.

To use this Developer Guide most effectively, readers should be familiar with ReSTful web services, HTTP/1.1, and JSON and/or XML serialization formats.

1.3. Document Change History

This version of the Developer Guide applies only to v2.0 of the API. The most recent changes are described in the table below:

Revision Date	Summary of Changes
Sep 14, 2012	• Repaired broken link from Section 4.3.1, "Authenticate" [57] to Section 3.2, "Sample Authentication Request and Response" [10]. • Corrected endpoint used in samples in Section 1.1, "QuickStart" [1]. • Updated samples to show `defaultRegion` in Section 3.2, "Sample Authentication Request and Response" [10] and Section 4.3.1, "Authenticate" [57].
Jul 31, 2012	• Added information about `defaultRegion` for Get User by Name [41], Get User by ID [43], and Update User [47]. • Updated Section 3.2.1, "Service Types in the Service Catalog" [19] and supporting examples to show `rax:dns` as the service type for Cloud DNS. • Simplified references to the Cloud Control Panel, showing that all customers have access to the Cloud Control Panel at http://mycloud.rackspace.com/.
Jul 11, 2012	• Removed "BETA" designation. • Updated illustration in Chapter 2, *Concepts* [7]. • Restructured and expanded Section 1.4, "Additional Resources" [4], adding links to Rackspace Knowledge Center and customer feedback site. • Added Section 3.8, "Statuses" [34], listing all possible account statuses. • Added Section 1.5, "Alternate Authentication Endpoints" [5], showing recommended and legacy authentication endpoints. • Updated Section 3.10, "Faults" [35] to show that the overLimit fault can occur with any request. • Standardized user roles as `user-admin` and `user-default`. • Standardized references to "user" and "customer", since it is possible to have a user, such as a Racker, who is not a customer. • Repaired broken link to Section 3.2, "Sample Authentication Request and Response" [10]. • Updated Section 3.2.1, "Service Types in the Service Catalog" [19] to provide link to publicly-available documentation on Cloud Databases.
Jun 16, 2012	• Added details to Section 3.6, "Namespaces" [32], describing new standards and defaults.
May 16, 2012	• For "Update User" and "Delete User" operations, added suggestion to use Get User by ID [43] to confirm that the intended user will be changed. • Added Section 3.7, "Sub-Users" [34]; moved most of the details from the Sub-User glossary item into that section, along with additional details on sub-users' inheritance of groups from parents. • Added Section 3.9, "Limits" [35], introducing rate limiting. • Corrected malformed cURL example in Section 1.1, "QuickStart" [1]. In the same section, added an introduction to API extensions.
Apr 12, 2012	• Updated twenty code samples, primarily to demonstrate use of namespaces. • Added Section 3.6, "Namespaces" [32], introducing the idea of namespaces and relating it to the idea of API extensions. • Linked Section 1.1, "QuickStart" [1] to Section 4.1.4, "Add User" [45] with note that sub-users added via API cannot use the control panel. • Updated Section 4.1.7, "List Credentials" [51] and **Sub-User** [9] to state that sub-users must authenticate with their passwords. • Added an annotated JSON example in Section 3.2, "Sample Authentication Request and Response" [10]. In the same section, updated the annotated XML example to show new service types. • Updated Section 3.2.1, "Service Types in the Service Catalog" [19] to show new service types.

Revision Date	Summary of Changes
Mar 30, 2012	• Updated authentication endpoints to https://identity.api.rackspacecloud.com/v2.0/ (for US-based accounts) and https://lon.identity.api.rackspacecloud.com/v2.0/ (for UK-based accounts). • Added Section 3.2, "Sample Authentication Request and Response" [10], focused on interpreting authentication responses. This section includes annotated examples relocated from the "Request/Response Types" section and adds details on service types and recommended workflow. In the same section, added a note suggesting service type as the primary value for locating a service. • Expanded the the list of operations described in Chapter 4, *API Operations for Client Developers* [37] and simplified its structure. In the same chapter, added notes on functionality provided by extensions to the core Identity API. • Added examples of creating a user with a password in Section 4.1.4, "Add User" [45]. In the same section, added information that an account can have up to 100 sub-users. • Added examples of changing a user's password in Section 4.1.5, "Update User" [47]. • Added information about sub-users (also known as sub-accounts), including an entry in Glossary [8]. • Clarified Section 4.1.7, "List Credentials" [51] to say that user passwords cannot be retrieved.
Feb 10, 2012	• Updated multiple examples to show cloudServersOpenStack (Nova) in addition to cloudServers in service catalog responses. • Added note in Section 3.1, "Request/Response Types" [10] that service type is stable across all releases and should be parsed for rather than service name. • Updated "Get Tenants Request with X-Auth-Token" example to show correct endpoint. • Added link to language binding examples in Section 1.1, "QuickStart" [1].
Dec 8, 2011	• Added examples of authentication with an API key in Section 4.3.1, "Authenticate" [57]. • Added annotations explaining key points in authentication request and response examples in Section 3.1, "Request/Response Types" [10]. • Added QuickStart section.
Nov 21, 2011	• Initial release for API v2.0.

1.4. Additional Resources

Descriptive information about this service is also published in its Web Application Description Language (WADL) and XML Schema Definition (XSD). You are welcome to read this information, but please use the appropriate US-based or UK-based version.

Your account may be based in either the US or the UK; this is not determined by your physical location but by the location of the Rackspace retail site which was used to create your account:

- If your account was created via http://www.rackspacecloud.com, it is a US-based account.
  - The US-based WADL is http://docs.rackspacecloud.com/auth/api/v2.0/auth.wadl.
  - The US-based XSDs are http://docs.rackspacecloud.com/auth/api/v2.0/common/xsd/api.xsd and http://docs.rackspacecloud.com/auth/api/v2.0/common/xsd/api-common.xsd.
- If your account was created via http:/www.rackspace.co.uk, it is a UK-based account.
  - The UK-based WADL is http://docs.rackspacecloud.com/auth/api/uk/v2.0/auth.wadl.
  - The UK-based XSDs are http://docs.rackspacecloud.com/auth/api/uk/v2.0/common/xsd/api.xsd and http://docs.rackspacecloud.com/auth/api/uk/v2.0/common/xsd/api-common.xsd.

If you are unsure how your account was created, use the Rackspace contact information at either site to ask for help.

In addition to this API documentation and its related WADLs and XSDs, you might find relevant information in several other places:

- To see how the Cloud Authentication Service is used by another Rackspace service, read that service's API documentation at docs.rackspace.com. You might also find useful articles and tutorials in the Rackspace Knowledge Center at rackspace.com/knowledge_center/.
- To learn about v1.1 of the Cloud Authentication Service, which supports Rackspace authentication but does not implement the OpenStack Keystone Identity Service, read the Rackspace Cloud Authentication Client Developer Guide for API v1.1 at http://docs.rackspace.com/auth/api/v1.1/auth-client-devguide-latest.pdf.
- To learn about Keystone, read OpenStack's Keystone documentation at keystone.openstack.org.

If you have a suggestion about any Rackspace product or service, including this one, please share it at feedback.rackspacecloud.com.

1.5. Alternate Authentication Endpoints

You can use the current Cloud Identity authentication endpoints with earlier versions of the API. All versions of the API, including legacy Identity contracts, are available if you authenticate at a current endpoint and specify an API version. Whichever version of the API you choose, you must use the endpoint designated for your US-based or UK-based account.

Rackspace recommends that you authenticate at one of the following Cloud Identity endpoints:

Table 1.1. Recommended Authentication Endpoints

Resource	v1	v1.1	v2.0
US-Based Endpoint	https://identity.api.rackspacecloud.com/v1	https://identity.api.rackspacecloud.com/v1.1	https://identity.api.rackspacecloud.com/v2.0
UK-Based Endpoint	https://lon.identity.api.rackspacecloud.com/v1	https://lon.identity.api.rackspacecloud.com/v1.1	https://lon.identity.api.rackspacecloud.com/v2.0
Documentation	No standalone documentation is available for Auth 1.0, but you can see Auth1.0 in use in First Generation Cloud Servers™ Developer Guide - API v1.0: Section 3.1, Authentication	Cloud Authentication Client Developer Guide - API v1.1	Cloud Identity Client Developer Guide - API v2.0

Alternatively, versions 1.0 and 1.1 are supported by the following legacy endpoints:

Table 1.2. Legacy Authentication Endpoints

Resource	v1	v1.1
US-Based Endpoint	https://auth.api.rackspacecloud.com/v1.0	https://auth.api.rackspacecloud.com/v1.1

Resource	v1	v1.1
UK-Based Endpoint	https://lon.auth.api.rackspacecloud.com/v1.0	https://lon.auth.api.rackspacecloud.com/v1.1
Documentation	No standalone documentation is available for Auth 1.0, but you can see Auth1.0 in use in First Generation Cloud Servers™ Developer Guide - API v1.0: Section 3.1, Authentication	Cloud Authentication Client Developer Guide - API v1.1

1.6. Release Notes

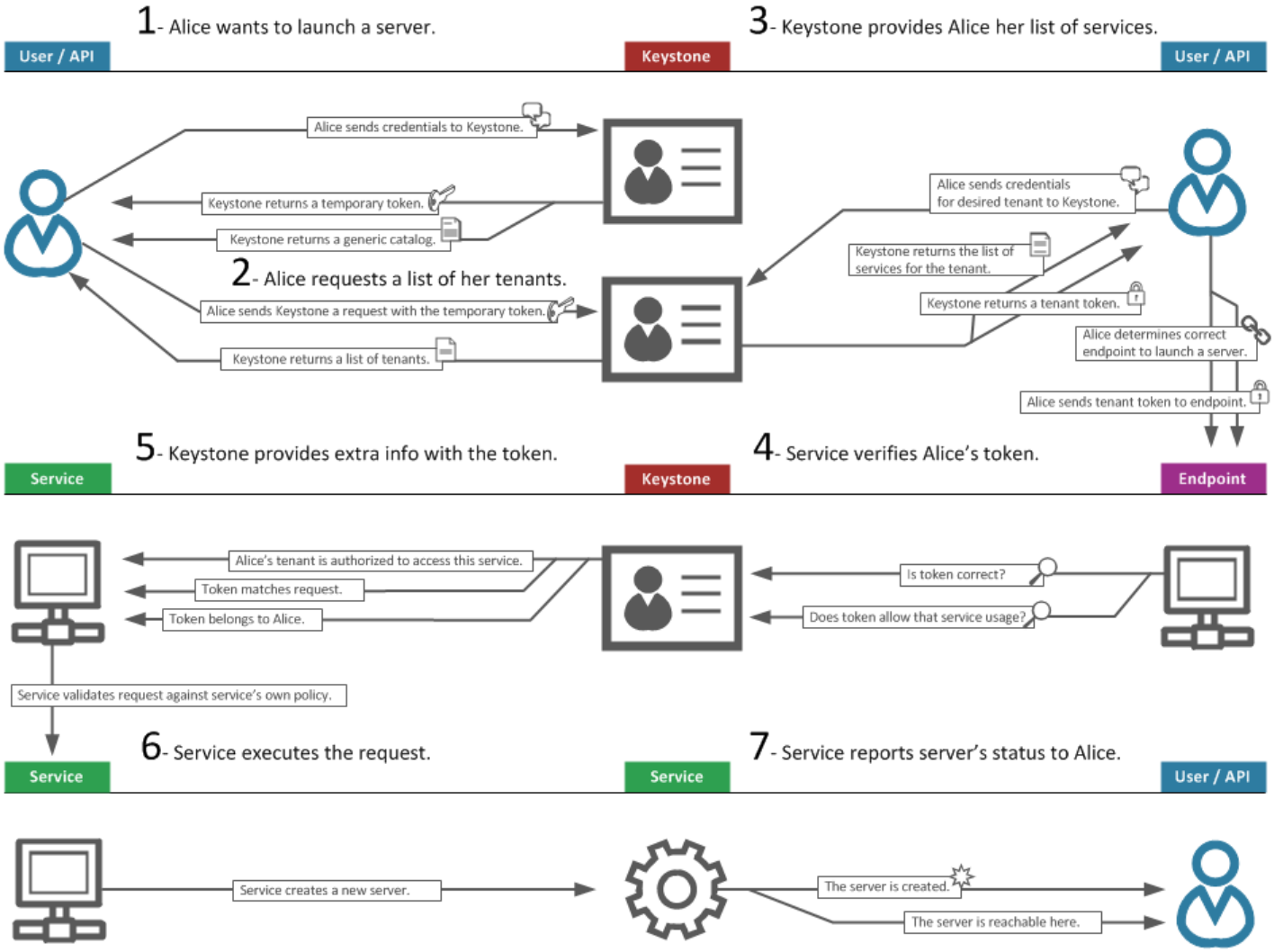
Version 2.0 of the Auth API differs from Version 1.1 in several important ways:

- **API CHANGES:**
  - API implements OpenStack's Keystone v2.0 API Specification.
- **NEW FEATURES:**
  - Introduction of tenants and roles.
  - Service catalog includes service types such as `compute` and `object-store`.
  - Service catalog includes version information and version list links.
  - Upon authentication, service catalog returns a separate `tenantID` attribute by endpoint.
  - Ability to add multiple users (sub-users) to an account. This functionality is available in the US and is coming soon to the UK.
  - Introduction of rate limiting.

2. Concepts

The basic function of the authentication service is to validate a client's credentials. If a client offers valid credentials, Auth returns a token and a service catalog. The client can use the service catalog to find endpoints for the services it can use. To any of those services, the client can show the token as evidence of having been authenticated.

Figure 2.1. A client sends credentials to the Auth service; if they are valid, the Auth service returns a token which can be used to identify the client to other services.



You can see some OpenStack-oriented illustrations of this idea, showing the OpenStack Keystone identity service authenticating for the OpenStack Nova compute service, at http://docs.openstack.org/trunk/openstack-identity/admin/content/example-flows.html.

Some of the OpenStack functions illustrated may not be consistent with the privileges available to you as a Rackspace customer; you are very welcome to examine this and other OpenStack publications as general background material, but please rely on Rackspace sources for specific information about what you can do and how you can do it.

If the client's authentication attempt fails, the authentication service returns 401 unAuth.

To use the Cloud Authentication Service effectively, you should understand several key concepts:

Glossary

API Key
Your API Key is a unique alphanumeric identifier associated with your account. You can use your API key as an authentication credential, paired with your username, to generate an authentication token which will be recognized by the services in your service catalog. To find or generate your API key, login to the Rackspace Cloud Control Panel at http://mycloud.rackspace.com/.

Authentication
Authentication is the act or process of confirming the identity of a user or the truth of a claim. The authentication service confirms that an incoming request is being made by the user who claims to be making the request. It does this by validating a set of claims that the user makes. These claims are initially in the form of a set of credentials. After initial confirmation based on credentials, the authentication service issues a token to the user; when making subsequent requests, the user can provide the token as evidence that the user's identity has already been authenticated.

Credentials
Credentials are data that belong to and identify a specific user. Because credentials are assumed to be known by only one user, users who present valid credentials are assumed to have proven that they are who they say they are. Examples of credentials include:

- a matching username and password
- a matching username and API key
- a unique token

Endpoint
An endpoint is an entry point to an API. The endpoint is defined as a set of base URLs. API operations are defined relative to these URLs. An API may offer several regional endpoints for a single API. Rackspace provides two authentication endpoints: one for US-based accounts and one for UK-based accounts. To use v2.0 of the API for authentication, specify the API version as shown below:

- US-based accounts authenticate through https://identity.api.rackspacecloud.com/v2.0/.
- UK-based accounts authenticate through https://lon.identity.api.rackspacecloud.com/v2.0/.

Your account may be based in either the US or the UK; this is not determined by your physical location but by the location of the Rackspace retail site which was used to create your account:

- If your account was created via http://www.rackspacecloud.com, it is a US-based account.
- If your account was created via http:/www.rackspace.co.uk, it is a UK-based account.

If you are unsure how your account was created, use the Rackspace contact information at either site to ask for help.

Role
A role is a personality that a user assumes when performing a specific set of operations. A role includes a set of rights and privileges. A user assuming a role inherits the rights and privileges associated with the role. A token that is issued to a user includes the list of roles the user can assume. When a user calls a service, that service determines how to interpret a user's roles. A role that grants access to a list of operations or resources within one service may grant access to a completely different list when interpreted by a different service.

Service
A service provides one or more endpoints through which users can access resources and perform operations. Examples of OpenStack services include Compute (Nova), Object Storage (Swift), and Image Service (Glance).

Service Catalog
Your service catalog is the list of services available to you, as returned along with your authentication token and an expiration date for that token. All the services in your service catalog should recognize your token as valid until it expires.

The catalog listing for each service provides at least one endpoint URL for that service. Other information, such as regions and versions and tenants, is provided if it's relevant to your access to this service.

Sub-User
A sub-user, also called a sub-account, is a child of the account's fully-privileged administrative user. Each account has exactly one administrative user, holding the `identity:user-admin` role; optionally, one or multiple sub-users can be created, each holding the `identity:default` role. Each sub-user shares its parent's tenant information, group memberships, and endpoints.

Tenant
A tenant is a container used to group or isolate resources and/or identity objects. Depending on the service operator, a tenant may map to a customer, account, organization, or project.

Token
A token is an opaque string that represents an authorization to access cloud resources. Tokens may be revoked at any time and are valid for a finite duration.

User
A user is a digital representation of a person, system, or service who consumes cloud services. Users have credentials and may be assigned tokens; based on these credentials and tokens, the authentication service validates that incoming requests are being made by the user who claims to be making the request, and that the user has the right to access the requested resources. Users may be directly assigned to a particular tenant and behave as if they are contained within that tenant.

3. General API Information

The authentication service allows clients to obtain tokens that can be used to access cloud services such as those provided by Rackspace and OpenStack.

The authentication service API is implemented using a ReSTful web service interface. All requests to authenticate and operate against the authentication service should be performed using SSL over HTTP (HTTPS) on TCP port 443.

3.1. Request/Response Types

The Keystone 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 by using the `Accept` header or by adding an `.xml` or `.json` extension to the request URI. It is possible for a response to be serialized using a format different from the request. 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. Response Types

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

3.2. Sample Authentication Request and Response

The sample request and response in this section illustrate a general case. In your authentication request, use your own credentials rather than the sample values shown here for `username`, `password`. and `tenantId`. When you authenticate successfully, the response to your authentication request will include a catalog of the services to which you have subscribed rather than the sample values shown here.

Example 3.1. Authentication Request with Headers: JSON

```
POST /v2.0/tokens HTTP/1.1
Host: identity.api.rackspacecloud.com
Content-Type: application/json
Accept: application/xml
```

```
{
  "auth": {
    "passwordCredentials": {
      "username": "demoauthor",❶
      "password": "mypass"❷
    },
    "tenantId": "1234"❸
  }
}
```

❶ This is the username you use to login to the Rackspace Cloud Control Panel at http://mycloud.rackspace.com/.

❷ This is the password you use to login to the Rackspace Cloud Control Panel.

❸ Tenant is an optional specification. Some services use multi-level authentication, with service-specific credentials in addition to vendor-specific credentials. In such cases, associating a user with a tenant can be a method of passing that additional level of identifying information to the service.

A successful authentication response includes a token and a service catalog. For each service in the service catalog, only relevant details are returned, so the description of one service may include a different set of details than the description of another service. For example, some services' endpoints may be associated with a default region; if there is no default region for that endpoint, `region` is omitted.

Example 3.2. Service Catalog in Authentication Response with Headers: XML

```
HTTP/1.1 200 OKAY
Date: Mon, 12 Nov 2010 15:55:01 GMT
Content-Length:
Content-Type: application/xml; charset=UTF-8
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<access❶
    xmlns:os-ksadm="http://docs.openstack.org/identity/api/ext/OS-
KSADM/v1.0"
    xmlns="http://docs.openstack.org/identity/api/v2.0"
    xmlns:rax-kskey="http://docs.rackspace.com/identity/api/ext/RAX-
KSKEY/v1.0"
    xmlns:rax-ksqa="http://docs.rackspace.com/identity/api/ext/RAX-
KSQA/v1.0"
    xmlns:common="http://docs.openstack.org/common/api/v1.0"
    xmlns:ksgrp="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/
v1.0"
    xmlns:rax-kscatalog="http://docs.openstack.org/identity/api/ext/
OS-KSCATALOG/v1.0"
    xmlns:atom="http://www.w3.org/2005/Atom">
```

```
    <token❷
        id="aaaaa-bbbbb-ccccc-dddd"
        expires="2012-04-13T13:15:00.000-05:00"/>
    <user
        xmlns:rax-auth="http://docs.rackspace.com/identity/api/ext/
RAX-AUTH/v1.0"
        id="161418" name="demoauthor" rax-auth:defaultRegion="DF❸W">
        <roles>❹
            <role id="3" name="identity:user-admin"
                description="User Admin Role."/>
        </roles>
    </user>
    <serviceCatalog>❺
        <service type="rax:database" name="cloudDatabases">
            <endpoint region="DFW" tenantId="12345"
                publicURL="https://dfw.databases.api.rackspacecloud.
com/v1.0/12345"/>
            <endpoint region="ORD" tenantId="12345"
                publicURL="https://ord.databases.api.rackspacecloud.
com/v1.0/12345"/>
        </service>
        <service type="rax:load-balancer" name="cloudLoadBalancers">
            <endpoint region="ORD" tenantId="12345"
                publicURL="https://ord.loadbalancers.api.
rackspacecloud.com/v1.0/12345"/>
            <endpoint region="DFW" tenantId="12345"
                publicURL="https://dfw.loadbalancers.api.
rackspacecloud.com/v1.0/12345"/>
        </service>
        <service type="rax:object-cdn" name="cloudFilesCDN">
            <endpoint region="DFW" tenantId="MossoCloudFS_aaaa-bbbbbb-
ccccc-ddddd"
                publicURL="https://cdn1.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbbbb-ccccc-ddddd"
            />
            <endpoint region="ORD" tenantId="MossoCloudFS_aaaa-bbbbbb-
ccccc-ddddd"
                publicURL="https://cdn2.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbbbb-ccccc-ddddd"
            />
        </service>
        <service type="rax:monitor" name="cloudMonitoring">
            <endpoint tenantId="12345"
                publicURL="https://monitoring.api.rackspacecloud.com/
v1.0/12345"/>
        </service>
        <service type="object-store❻" name="cloudFiles❼">
            <endpoint region="DFW❽" tenantId="MossoCloudFS_aaaa-
bbbbbb-ccccc-dddd❾d"
                publicURL="https://storage101.dfw1.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbbbb-ccccc-ddddd"
                internalURL="https://snet-storage101.dfw1.clouddrive.
com/v1/MossoCloudFS_aaaa-bbbbbb-cccc❿c-ddddd"
```

```
            />
            <endpoint region="ORD" tenantId="MossoCloudFS_aaaa-bbbbbb-
ccccc-ddddd"
                publicURL="https://storage101.ord1.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbbbb-ccccc-ddddd"
                internalURL="https://snet-storage101.ord1.clouddrive.
com/v1/MossoCloudFS_aaaa-bbbbbb-ccccc-ddddd"
            />
        </service>
        <service type="compute" name="cloudServers">
            <endpoint tenantId="12345"
                publicURL="https://servers.api.rackspacecloud.com/v1.
0/12345">
                <version id="1.0" info="https://servers.api.
rackspacecloud.com/v1.0"
                    list="https://servers.api.rackspacecloud.com/"/>
            </endpoint>
        </service>
        <service type="compute" name="cloudServersOpenStack">
            <endpoint region="DFW" tenantId="12345"
                publicURL="https://dfw.servers.api.rackspacecloud.com/
v2/12345">
                <version id="2" info="https://dfw.servers.api.
rackspacecloud.com/v2"
                    list="https://dfw.servers.api.rackspacecloud.com/
"/>
            </endpoint>
            <endpoint region="ORD" tenantId="12345"
                publicURL="https://ord.servers.api.rackspacecloud.com/
v2/12345">
                <version id="2" info="https://ord.servers.api.
rackspacecloud.com/v2"
                    list="https://ord.servers.api.rackspacecloud.com/
"/>
            </endpoint>
        </service>
        <service type="rax:dns" name="cloudDNS">
            <endpoint tenantId="12345"
                publicURL="https://dns.api.rackspacecloud.com/v1.0/
12345"/>
        </service>
    </serviceCatalog>
</access>
```

❶ In XML responses only, a list of namespaces identifies API extensions that add functionality to the core API. You can read more about namespaces in Section 3.6, “Namespaces” [32].

❷ This token can be presented to a service as evidence of authentication. Tokens are valid for a finite duration; a token's default lifespan is twenty-four hours.

The token's `expires` attribute denotes the time after which the token will automatically become invalid. A token may be manually revoked before the time identified by the `expires` attribute; `expires` predicts a token's maximum possible lifespan but does not guarantee that it will reach that lifespan. Clients are encouraged to cache a token until it expires.

❸ Users can be assigned a default region so that, when there is a choice between multiple endpoints associated with a service in the user's catalog, the endpoint for the user's default region will be selected if it is available. In this example, the user's default region is `DFW` and several of the services in the user's catalog offer endpoints in that region and the `ORD` region; this user's work will be directed to the `DFW` region whenever possible.

❹ Users can be assigned multiple roles, with each role providing specific privileges. In this example, `joeuser` is the administrative user for the account, holding the fully-privileged `identity:user-admin` role. Other users might hold other roles with different privileges. Roles need not be associated with actual job functions such as Administrator, Operator, Developer, Tester, or Trainer.

❺ The service catalog lists the services this user can access. In this example, the user can access two compute services (Cloud Servers OpenStack and Cloud Servers) and two object storage services (Cloud Files Content Distribution Network (CDN), and Cloud Files), as well as one database service, one DNS service, one loadbalancing service, and one monitoring service. The catalog listing for each service provides at least one endpoint URL for that service. Other information, such as regions and versions and tenants, is provided if it's relevant to this user's access to this service.

❻ The service type attribute identifies services that perform similar functions, whatever those services might be named. In this example, the services named cloudServers and cloudServersOpenstack are both identified as `type="compute"`, identifying them as compute services even though the word "compute" does not appear in their names.

Important

Use service type as the primary value for locating a service. If multiple endpoints of the same service type exist in the same region, use service name as the tiebreaker.

❼ The service name attribute identifies each unique service in the catalog. Once a service is created, its name does not change. However, new services of the same service type may be added to the catalog with new names.

Important

If you are programmatically parsing an authentication response, use service type rather than service name as the basis for determining whether a user has access to a particular kind of service. Service type is stable across all releases; new service types may be developed, but existing service types are not renamed. In this example, `type="compute"` identifies all the available compute services, one of which is named cloudServers and one of which is named cloudServersOpenStack. New compute service names may

be added in future releases; whatever the compute services are named, you can always recognize them by parsing for `type="compute"` in the authentication response's service catalog.

❽ A service may expose endpoints in different regions. Regional endpoints allow clients to provision resources in a manner that provides high availability.

Some services are not region-specific. These services supply a single non-regional endpoint and do not provide access to internal URLs.

❾ Some services recognize specification of a tenant. If a service does recognize tenants, the format of the tenant specification is defined only by the service; for details about whether and how to specify a tenant, check the documentation for the service you are using.

❿ An endpoint can be assigned public and internal URLs. A public URL is accessible from anywhere. Access to a public URL usually incurs traffic charges. Internal URLs are only accessible to services within the same region. Access to an internal URL is free of charge.

Example 3.3. Service Catalog in Authentication Response: JSON

```
{
    "access": {
        "serviceCatalog": [❶
            {
                "endpoints": [
                   {
                        "publicURL": "https://ord.servers.api.rackspacecloud.
com/v2/❷12345",
                        "region": "ORD",❸
                        "tenantId": "12345",❹
                        "versionId": "2",
                        "versionInfo": "https://ord.servers.api.
rackspacecloud.com/v2",
                        "versionList": "https://ord.servers.api.
rackspacecloud.com/"
                    }
                    {
                        "publicURL": "https://dfw.servers.api.rackspacecloud.
com/v2/12345",
                        "region": "DFW",
                        "tenantId": "12345",
                        "versionId": "2",
                        "versionInfo": "https://dfw.servers.api.
rackspacecloud.com/v2",
                        "versionList": "https://dfw.servers.api.
rackspacecloud.com/"
                    }
                ],
                "name": "cloudServersOpenStack",❺
                "type": "compute"❻
            },
            {
                "endpoints": [
```

```
                    {
                        "publicURL": "https://ord.databases.api.
rackspacecloud.com/v1.0/12345",
                        "region": "ORD",
                        "tenantId": "12345"
                    },
                    {
                        "publicURL": "https://dfw.databases.api.
rackspacecloud.com/v1.0/12345",
                        "region": "DFW",
                        "tenantId": "12345"
                    }
                ],
                "name": "cloudDatabases",
                "type": "rax:database"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://ord.loadbalancers.api.
rackspacecloud.com/v1.0/12345",
                        "region": "ORD",
                        "tenantId": "645990"
                    },
                    {
                        "publicURL": "https://dfw.loadbalancers.api.
rackspacecloud.com/v1.0/12345",
                        "region": "DFW",
                        "tenantId": "12345"
                    }
                ],
                "name": "cloudLoadBalancers",
                "type": "rax:load-balancer"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://cdn1.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbb-cccc ",
                        "region": "DFW",
                        "tenantId": "MossoCloudFS_aaaa-bbbb-cccc "
                    }
                    {
                        "publicURL": "https://cdn2.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbb-cccc ",
                        "region": "ORD",
                        "tenantId": "MossoCloudFS_aaaa-bbbb-cccc "
                    }
                ],
                "name": "cloudFilesCDN",
                "type": "rax:object-cdn"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://dns.api.rackspacecloud.com/v1.0/
12345",
                        "tenantId": "12345"
                    }
                ],
```

```
                "name": "cloudDNS",
                "type": "rax:dns"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://servers.api.rackspacecloud.com/
v1.0/12345",
                        "tenantId": "12345",
                        "versionId": "1.0",
                        "versionInfo": "https://servers.api.rackspacecloud.
com/v1.0",
                        "versionList": "https://servers.api.rackspacecloud.
com/"
                    }
                ],
                "name": "cloudServers",
                "type": "compute"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://monitoring.api.rackspacecloud.
com/v1.0/12345",
                        "tenantId": "12345"
                    }
                ],
                "name": "cloudMonitoring",
                "type": "rax:monitor"
            },
            {
                "endpoints": [
                    {
                        "internalURL": "https://snet-storage101.dfw1.
clouddrive.com/v1/MossoCloudFS_aaaa-bbbb-cccc ",
                        "publicURL": "https://storage101.dfw1.clouddrive.com/
v1/MossoCloudFS_aaaa-bbbb-cccc ",
                        "region": "DFW",
                        "tenantId": "MossoCloudFS_aaaa-bbbb-cccc"
                    }
                                        {
                        "internalURL": "https://snet-storage101.ord1.
clouddrive.com/v1/MossoCloudFS_aaaa-bbbb-cccc ",
                        "publicURL": "https://storage101.ord1.clouddrive.com/
v1/MossoCloudFS_aaaa-bbbb-cccc ",
                        "region": "ORD",
                        "tenantId": "MossoCloudFS_aaaa-bbbb-cccc"
                    }
                ],
                "name": "cloudFiles",
                "type": "object-store"
            }
        ],
        "token": {❼
            "expires": "2012-04-13T13:15:00.000-05:00",
            "id": "aaaaa-bbbbb-ccccc-dddd"
        },
        "user": {
        "RAX-AUTH:defaultRegion": "DFW",❽
            "id": "161418",
```

```
            "name": "demoauthor",
            "roles": [❾
                {
                    "description": "User Admin Role.",
                    "id": "3",
                    "name": "identity:user-admin"
                }
            ]
        }
    }
}
```

❶ The service catalog lists the services this user can access. In this example, the user can access two compute services (Cloud Servers OpenStack and Cloud Servers) and two object storage services (Cloud Files Content Distribution Network (CDN), and Cloud Files), as well as one database service, one DNS service, one loadbalancing service, and one monitoring service. The catalog listing for each service provides at least one endpoint URL for that service. Other information, such as regions and versions and tenants, is provided if it's relevant to this user's access to this service.

❷ An endpoint can be assigned public and internal URLs. A public URL is accessible from anywhere. Access to a public URL usually incurs traffic charges. Internal URLs are only accessible to services within the same region. Access to an internal URL is free of charge.

❸ A service may expose endpoints in different regions. Regional endpoints allow clients to provision resources in a manner that provides high availability.

Some services are not region-specific. These services supply a single non-regional endpoint and do not provide access to internal URLs.

❹ Some services recognize specification of a tenant. If a service does recognize tenants, the format of the tenant specification is defined only by the service; for details about whether and how to specify a tenant, check the documentation for the service you are using.

❺ The service name attribute identifies each unique service in the catalog. Once a service is created, its name does not change. However, new services of the same service type may be added to the catalog with new names.

Important

If you are programmatically parsing an authentication response, use service type rather than service name as the basis for determining whether a user has access to a particular kind of service. Service type is stable across all releases; new service types may be developed, but existing service types are not renamed. In this example, `type="compute"` identifies all the available compute services, one of which is named cloudServers and one of which is named cloudServersOpenStack. New compute service names may be added in future releases; whatever the compute services are named, you can always recognize them by parsing for `type="compute"` in the authentication response's service catalog.

❻ The service type attribute identifies services that perform similar functions, whatever those services might be named. In this example, the services named cloudServers and cloudServersOpenstack are both identified as `type="compute"`, identifying them as compute services even though the word "compute" does not appear in their names.

Important

Use service type as the primary value for locating a service. If multiple endpoints of the same service type exist in the same region, use service name as the tiebreaker.

❼ This token can be presented to a service as evidence of authentication. Tokens are valid for a finite duration; a token's default lifespan is twenty-four hours.

The token's `expires` attribute denotes the time after which the token will automatically become invalid. A token may be manually revoked before the time identified by the `expires` attribute; `expires` predicts a token's maximum possible lifespan but does not guarantee that it will reach that lifespan. Clients are encouraged to cache a token until it expires.

❾ Users can be assigned multiple roles, with each role providing specific privileges. In this example, `joeuser` is the administrative user for the account, holding the fully-privileged `identity:user-admin` role. Other users might hold other roles with different privileges. Roles need not be associated with actual job functions such as Administrator, Operator, Developer, Tester, or Trainer.

❸ Users can be assigned a default region so that, when there is a choice between multiple endpoints associated with a service in the user's catalog, the endpoint for the user's default region will be selected if it is available. In this example, the user's default region is `DFW` and several of the services in the user's catalog offer endpoints in that region and the `ORD` region; this user's work will be directed to the `DFW` region whenever possible.

3.2.1. Service Types in the Service Catalog

As shown in Section 3.2, “Sample Authentication Request and Response” [10], every service listed in the service catalog is associated with a service type. You can use the service type to recognize services that perform similar functions, whatever those services are named. The following table shows all the available service types and relates them to their corresponding OpenStack projects and those projects' documentation.

Table 3.2. Service Types

Service Type	Description
`compute`	Services of this type are compatible with the OpenStack Compute (Nova) API 1.1 as documented at http://nova.openstack.org/. *Cloud Servers, a Rackspace service of this type, is documented at* http://docs.rackspace.com/api/.
`identity`	Services of this type are compatible with the OpenStack Keystone API 2.0 as documented at http://keystone.openstack.org/.

Service Type	Description
	Cloud Identity, a Rackspace service of this type, is documented at http://docs.rackspace.com/api/.
`image-service`	Services of this type are compatible with the OpenStack Glance API as documented at http://glance.openstack.org/.
`object-store`	Services of this type are compatible with the OpenStack Swift API as documented at http://swift.openstack.org/. *Cloud Files, a Rackspace service of this type, is documented at* http://docs.rackspace.com/api/.
`rax:database`	*Cloud Databases, a Rackspace service of this type, is documented at* http://docs.rackspace.com/api/.
`rax:dns`	*Cloud DNS, a Rackspace service of this type, is documented at* http://docs.rackspace.com/api/.
`rax:load-balancer`	*Cloud Load Balancers, a Rackspace service of this type, is documented at* http://docs.rackspace.com/api/.
`rax:monitor`	*Cloud Monitoring, a Rackspace service of this type, is documented at* http://docs.rackspace.com/api/.
`rax:object-cdn`	*Cloud Files, a Rackspace service that supports a Content Distribution Network (CDN), is documented at* http://docs.rackspace.com/api/.

3.2.2. Suggested Workflow for Processing a Service Catalog Response

When your client issues a successful authentication request, the Cloud Identity Service responds with a catalog showing the services available to that client, with one endpoint listed for each available service; you can find an annotated example of such a response in Section 3.2, "Sample Authentication Request and Response" [10].

As the client developer, you must decide how your client should use the contents of its service catalog. The first step may be to identify a service to which your client should be connected. Here is one possible workflow for processing the service catalog response to identify available services and their endpoints:

1. If the service catalog lists only one endpoint, use it. This will connect your client to the only service available to that client.

2. If the service catalog lists multiple endpoints, you must establish a process for choosing an endpoint to connect your client to:

   a. If the user has not specified which endpoint to use, generate an error.

   b. If the user has specified which endpoint to use, help the user identify that endpoint from within the catalog:

      i. Support filtering by endpoint name, service name, service type, region name, and version.

      ii. Support manual specification of an endpoint via a URL parameter.

Authentication endpoints follow the process described above.

To connect a client to an endpoint listed as `type="compute"`, do the following:

1. Go that compute service's endpoint and use its `WWW-Authenticate` header to determine what authentication server it uses.
2. Go to that authentication server and authenticate.
3. Return to the compute endpoint and proceed with using the compute service.

3.3. 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. By default, encoding is disabled.

Table 3.3. Compression Headers

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

3.4. Versions

The identity API uses a URI versioning scheme.

Important

Rackspace's implementation of the OpenStack Keystone Identity Service v2.0 does not yet support MIME-type versioning.

The first element of the URI path contains the target version identifier: for example, in https://identity.api.rackspacecloud.com/v2.0/..., the API version is 2.0. Other than requests to query for version, all requests must contain a target version.

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, older versions will be marked as DEPRECATED. Rackspace will work with developers and partners to ensure that there is adequate time to migrate to the new version before deprecated versions are discontinued.

Example 3.4. Request with URI Versioning

```
GET /v2.0/tenants HTTP/1.1
Host: identity.api.rackspacecloud.com
Accept: application/xml
X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
```

New features and functionality that do not break API compatibility will be introduced in the current version of the API as extensions and the URI will remain unchanged. Features or functionality changes that would disrupt API compatibility will require a new version, which will result in the URI 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.

You can programmatically determine available API versions by performing a **GET** on the root URL returned from the authentication system. In the root URL, the version and everything to the right of it is truncated. An Atom representation of the version's resources is supported when issuing a request with the `Accept` header containing `application/atom+xml` or by adding `.atom` to the request URI. This allows standard Atom clients to track version changes.

Example 3.5. Versions List Request

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

Normal Response Code(s): 200, 203

Error Response Code(s): badRequest (400), identityFault (500), serviceUnavailable(503)

This operation does not require a request body.

Example 3.6. 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://identity.api.rackspacecloud.com/v1.0/"/>
  </version>

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

  <version id="v2.0" status="BETA"
          updated="2011-05-27T20:22:02.25Z">
     <atom:link rel="self"
             href="http://identity.api.rackspacecloud.com/v2.0/"/>
  </version>

</versions>
```

Example 3.7. 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://identity.api.rackspacecloud.com/</id>
    <author><name>Rackspace</name><uri>http://www.rackspace.com/</uri></
author>
   <link rel="self" href="http://identity.api.rackspacecloud.com/"/>
    <entry>
       <id>http://identity.api.rackspacecloud.com/v2.0/</id>
       <title type="text">Version v2.0</title>
       <updated>2011-05-27T20:22:02.25Z</updated>
       <link rel="self" href="http://identity.api.rackspacecloud.com/v2.0/"/>
       <content type="text">Version v2.1 CURRENT (2011-05-27T20:22:02.25Z)</
content>
    </entry>
    <entry>
       <id>http://identity.api.rackspacecloud.com/v1.1/</id>
       <title type="text">Version v1.1</title>
       <updated>2010-12-12T18:30:02.25Z</updated>
       <link rel="self" href="http://identity.api.rackspacecloud.com/v1.1/"/>
       <content type="text">Version v1.1 CURRENT (2010-12-12T18:30:02.25Z)</
content>
    </entry>
    <entry>
       <id>http://identity.api.rackspacecloud.com/v1.0/</id>
       <title type="text">Version v1.0</title>
       <updated>2009-10-09T11:30:00Z</updated>
       <link rel="self" href="http://identity.api.rackspacecloud.com/v1.0/"/>
       <content type="text">Version v1.0 DEPRECATED (2009-10-09T11:30:00Z)</
content>
    </entry>
</feed>
```

Example 3.8. Versions List Response: JSON

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

```
        ]
      }, {
        "id": "v2.0",
        "status": "BETA",
        "updated": "2011-05-27T20:22:02.25Z",
        "links": [
          {
            "rel": "self",
            "href": "http://identity.api.rackspacecloud.com/v2.0/"
          }
        ]
      }
    ]
  }
}
```

You can obtain additional detailed information about a specific version by performing a **GET** on the base version URL. For example, `https://identity.api.rackspacecloud.com/v1.0/` is a base version URL, in which `v1.0` is the initial version of the API. 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 trailing slash. For example, `https://identity.api.rackspacecloud.com/v2.0/.xml` includes `.xml` as a format extension. Note that this is a special case that does not hold true for other API requests. In general, requests such as `/tenants.xml` and `/tenants/.xml` are handled equivalently.

Example 3.9. Version Details Request

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

Normal Response Code(s): 200, 203

Error Response Code(s): badRequest (400), identityFault (500), serviceUnavailable(503)

This operation does not require a request body.

Example 3.10. 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="v2.0" status="CURRENT" updated="2011-01-21T11:33:21-06:00">

     <media-types>
         <media-type base="application/xml"
            type="application/vnd.openstack.identity+xml;version=2.0"/>
         <media-type base="application/json"
            type="application/vnd.openstack.identity+json;version=2.0"/>
     </media-types>

     <atom:link rel="self"
         href="http://identity.api.rackspacecloud.com/v2.0/"/>
```

```
    <atom:link rel="describedby"
               type="application/pdf"
               href="http://docs.rackspacecloud.com/auth/api/v2.0/auth-client-
devguide-latest.pdf" />

    <atom:link rel="describedby"
               type="application/vnd.sun.wadl+xml"
               href="http://docs.rackspacecloud.com/auth/api/v2.0/auth.wadl" /
>
</version>
```

Example 3.11. 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://identity.api.rackspacecloud.com/v2.0/</id>
   <author><name>OpenStack</name><uri>http://www.openstack.org/</uri></author>
   <link rel="self" href="http://identity.api.openstack.org/v2.0/"/>
   <entry>
      <id>http://identity.api.rackspacecloud.com/v2.0/</id>
      <title type="text">Version v2.0</title>
      <updated>2011-01-21T11:33:21-06:00</updated>
     <link rel="self" href="http://identity.api.rackspacecloud.com/v2.0/"/>
      <link rel="describedby" type="application/pdf"
           href="http://docs.rackspacecloud.com/auth/api/v2.0/auth-client-
devguide-latest.pdf"/>
      <link rel="describedby" type="application/vnd.sun.wadl+xml"
           href="http://docs.rackspacecloud.com/auth/api/v2.0/auth.wadl"/>
      <content type="text">Version v2.0 CURRENT (2011-01-21T11:33:21-06:00)</
content>
   </entry>
</feed>
```

Example 3.12. Version Details Response: JSON

```
{
  "version": {
    "id": "v2.0",
    "status": "CURRENT",
    "updated": "2011-01-21T11:33:21-06:00",
    "links": [
      {
        "rel": "self",
        "href": "http://identity.api.rackspacecloud.com/v2.0/"
      }, {
        "rel": "describedby",
        "type": "application/pdf",
        "href": "http://docs.rackspacecloud.com/auth/api/v2.0/auth-client-
devguide-latest.pdf"
      }, {
        "rel": "describedby",
        "type": "application/vnd.sun.wadl+xml",
        "href": "http://docs.rackspacecloud.com/auth/api/v2.0/auth.wadl"
```

```
      }
    ],
    "media-types": [
      {
        "base": "application/xml",
        "type": "application/vnd.openstack.identity+xml;version=2.0"
      }, {
        "base": "application/json",
        "type": "application/vnd.openstack.identity+json;version=2.0"
      }
    ]
  }
}
```

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.

3.5. Extensions

The authentication service API is extensible, meaning that the API is structured so that some functions are implemented in the core API and others are implemented via optional extensions to that core. Extensions serve two purposes:

- Extensions allow the introduction of new features in the API without requiring a version change.
- Extensions allow the introduction of vendor-specific niche functionality.

Every service provider can choose its own set of extensions to the core API. 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, so a response is only applicable to a specific version.

Verb	URI	Description
GET	v2.0/extensions	Returns a list of available extensions

Normal Response Code(s): 200, 203

Error Response Code(s): badRequest (400), identityFault (500), serviceUnavailable(503)

This operation does not require a request body.

Each extension is identified by two unique identifiers: a namespace and an alias. You can use these identifiers to associate an extension with its vendor; in the examples in

this section, aliases prefixed RAX- indicate that Rackspace provided these extensions. Additionally, an extension contains links to documentation in multiple formats.

Example 3.13. Extensions Response: XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns5:extensions xmlns:ns1="http://docs.openstack.org/identity/api/ext/OS-
KSADM/v1.0"
    xmlns:ns2="http://docs.openstack.org/identity/api/v2.0"
    xmlns:ns3="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"
    xmlns:ns4="http://docs.rackspace.com/identity/api/ext/RAX-KSQA/v1.0"
    xmlns:ns5="http://docs.openstack.org/common/api/v1.0"
    xmlns:ns6="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0"
    xmlns:ns7="http://docs.openstack.org/identity/api/ext/OS-KSCATALOG/v1.0"
    xmlns:ns8="http://www.w3.org/2005/Atom">

    <ns5:extension name="Rackspace API Key Authentication Admin Extension"
        namespace="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"
        alias="RAX-KSKEY" updated="2011-08-14T13:25:27-06:00">
        <ns5:description>
            Rackspace extension to OpenStack Identity v2.0 API enabling API
 Key authentication.
        </ns5:description>
        <ns8:link rel="describedby" type="application/xhtml+xml"
            href="http://docs.rackspace.com/auth/api/v2.0/auth-client-
devguide/content/QuickStart-000.html"/>
        <ns8:link rel="describedby" type="application/vnd.sun.wadl+xml" href=
"http://docs.rackspacecloud.com/auth/api/v2.0/RAX-KSKEY/RAX-KSKEY-admin.wadl"/
>
    </ns5:extension>

    <ns5:extension name="Rackspace Keystone Group Extensions"
        namespace="http://docs.rackspace.com/identity/api/ext/RAX-KSGROUP/v1.
0"
        alias="RAX-KSGRP" updated="2011-08-14T13:25:27-06:00">
        <ns5:description>
            Rackspace extensions to OpenStack Identity v2.0 API enabling
 groups.
        </ns5:description>
        <ns8:link rel="describedby" type="application/xhtml+xml" href="http:/
/docs.rackspace.com/auth/api/v2.0/auth-client-devguide/content/QuickStart-000.
html"/>
        <ns8:link rel="describedby" type="application/vnd.sun.wadl+xml" href=
"http://docs.rackspacecloud.com/auth/api/v2.0/RAX-KSGRP/RAX-KSGRP-admin.wadl"/
>
    </ns5:extension>

    <ns5:extension name="Rackspace Keystone Secret Question and Answer"
        namespace="http://docs.rackspace.com/identity/api/ext/RAX-KSQA/v1.0"
        alias="RAX-KSQA" updated="2012-01-23T10:40:00-04:00">
        <ns5:description>
            Allows the management of a User's Secret Question and Answer.
        </ns5:description>
        <ns8:link rel="describedby" type="application/xhtml+xml" href="http:/
/docs.rackspace.com/auth/api/v2.0/auth-client-devguide/content/QuickStart-000.
html"/>
        <ns8:link rel="describedby" type="application/xml" href="http://docs.
rackspacecloud.com/auth/api/v2.0/RAX-KSQA/RAX-KSQA-admin.wadl"/>
    </ns5:extension>
```

```
    <ns5:extension name="Group Admin Extension"
        namespace="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0"
        alias="RAX-GRPADM" updated="2012-01-23T10:40:00-04:00">
        <ns5:description>
            Allows the management of groups in the identity server.
        </ns5:description>
        <ns8:link rel="describedby" type="application/xhtml+xml" href="http:/
/docs.rackspace.com/auth/api/v2.0/auth-client-devguide/content/QuickStart-000.
html"/>
        <ns8:link rel="describedby" type="application/xml" href="http://docs.
rackspacecloud.com/auth/api/v2.0/RAX-GRPADM/RAX-GRPADM.wadl"/>
    </ns5:extension>

    <ns5:extension name="OpenStack KSADM Extension"
        namespace="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
        alias="OS-KSADM" updated="2012-01-23T10:40:00-04:00">
        <ns5:description>
            Allows the management of Users, Tenants, Roles and Services in the
 identity server.
        </ns5:description>
        <ns8:link rel="describedby" type="application/xhtml+xml" href="http:/
/docs.rackspace.com/auth/api/v2.0/auth-client-devguide/content/QuickStart-000.
html"/>
        <ns8:link rel="describedby" type="application/xml" href="http://docs.
rackspacecloud.com/auth/api/v2.0/OS-KSADM/OS-KSADM-admin.wadl"/>
    </ns5:extension>

</ns5:extensions>
```

Example 3.14. Extensions Response: JSON

```
{
    "extensions": [
        {
            "alias": "RAX-KSKEY",
            "description": "Rackspace extension to OpenStack Identity v2.0 API
 enabling API Key authentication.",
            "links": [
                {
                    "href": "http://docs.rackspace.com/auth/api/v2.0/auth-
client-devguide/content/QuickStart-000.html",
                    "rel": "describedby",
                    "type": "application/xhtml+xml"
                },
                {
                    "href": "http://docs.rackspacecloud.com/auth/api/v2.0/RAX-
KSKEY/RAX-KSKEY-admin.wadl",
                    "rel": "describedby",
                    "type": "application/vnd.sun.wadl+xml"
                }
            ],
            "name": "Rackspace API Key Authentication Admin Extension",
            "namespace": "http://docs.rackspace.com/identity/api/ext/RAX-
KSKEY/v1.0",
            "updated": "2011-08-14T13:25:27-06:00"
        },
        {
```

```
            "alias": "RAX-KSGRP",
            "description": "Rackspace extensions to OpenStack Identity v2.0
 API enabling groups.",
            "links": [
                {
                    "href": "http://docs.rackspace.com/auth/api/v2.0/auth-
client-devguide/content/QuickStart-000.html",
                    "rel": "describedby",
                    "type": "application/xhtml+xml"
                },
                {
                    "href": "http://docs.rackspacecloud.com/auth/api/v2.0/RAX-
KSGRP/RAX-KSGRP-admin.wadl",
                    "rel": "describedby",
                    "type": "application/vnd.sun.wadl+xml"
                }
            ],
            "name": "Rackspace Keystone Group Extensions",
            "namespace": "http://docs.rackspace.com/identity/api/ext/RAX-
KSGROUP/v1.0",
            "updated": "2011-08-14T13:25:27-06:00"
        },
        {
            "alias": "RAX-KSQA",
            "description": "Allows the management of a User's Secret Question
 and Answer.",
            "links": [
                {
                    "href": "http://docs.rackspace.com/auth/api/v2.0/auth-
client-devguide/content/QuickStart-000.html",
                    "rel": "describedby",
                    "type": "application/xhtml+xml"
                },
                {
                    "href": "http://docs.rackspacecloud.com/auth/api/v2.0/RAX-
KSQA/RAX-KSQA-admin.wadl",
                    "rel": "describedby",
                    "type": "application/xml"
                }
            ],
            "name": "Rackspace Keystone Secret Question and Answer",
            "namespace": "http://docs.rackspace.com/identity/api/ext/RAX-KSQA/
v1.0",
            "updated": "2012-01-23T10:40:00-04:00"
        },
        {
            "alias": "RAX-GRPADM",
            "description": "Allows the management of groups in the identity
 server.",
            "links": [
                {
                    "href": "http://docs.rackspace.com/auth/api/v2.0/auth-
client-devguide/content/QuickStart-000.html",
                    "rel": "describedby",
                    "type": "application/xhtml+xml"
                },
                {
                    "href": "http://docs.rackspacecloud.com/auth/api/v2.0/RAX-
GRPADM/RAX-GRPADM.wadl",
                    "rel": "describedby",
```

```
                    "type": "application/xml"
                }
            ],
            "name": "Group Admin Extension",
            "namespace": "http://docs.rackspace.com/identity/api/ext/RAX-
KSGRP/v1.0",
            "updated": "2012-01-23T10:40:00-04:00"
        },
        {
            "alias": "OS-KSADM",
            "description": "Allows the management of Users, Tenants, Roles and
 Services in the identity server.",
            "links": [
                {
                    "href": "http://docs.rackspace.com/auth/api/v2.0/auth-
client-devguide/content/QuickStart-000.html",
                    "rel": "describedby",
                    "type": "application/xhtml+xml"
                },
                {
                    "href": "http://docs.rackspacecloud.com/auth/api/v2.0/OS-
KSADM/OS-KSADM-admin.wadl",
                    "rel": "describedby",
                    "type": "application/xml"
                }
            ],
            "name": "OpenStack KSADM Extension",
            "namespace": "http://docs.openstack.org/identity/api/ext/OS-KSADM/
v1.0",
            "updated": "2012-01-23T10:40:00-04:00"
        }
    ]
}
```

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

Verb	URI	Description
GET	/extensions/*alias*	Return details of a single extension

Normal Response Code(s): 200, 203

Error Response Code(s): itemNotFound (404), badRequest (400), identityFault (500), serviceUnavailable(503)

This operation does not require a request body.

Example 3.15. Extension Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<ns5:extension xmlns:ns1="http://docs.openstack.org/identity/api/ext/OS-KSADM/
v1.0"
    xmlns:ns2="http://docs.openstack.org/identity/api/v2.0"
    xmlns:ns3="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"
    xmlns:ns4="http://docs.rackspace.com/identity/api/ext/RAX-KSQA/v1.0"
```

```
    xmlns:ns5="http://docs.openstack.org/common/api/v1.0"
    xmlns:ns6="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0"
    xmlns:ns7="http://docs.openstack.org/identity/api/ext/OS-KSCATALOG/v1.0"
    xmlns:ns8="http://www.w3.org/2005/Atom"
    name="Rackspace API Key Authentication Admin Extension"
    namespace="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"
    alias="RAX-KSKEY" updated="2011-08-14T13:25:27-06:00">
    <ns5:description>
        Rackspace extension to OpenStack Identity v2.0 API enabling API Key
 authentication.
    </ns5:description>
    <ns8:link rel="describedby" type="application/xhtml+xml"
        href="http://docs.rackspace.com/auth/api/v2.0/auth-client-devguide/
content/QuickStart-000.html"/>
    <ns8:link rel="describedby" type="application/vnd.sun.wadl+xml"
        href="http://docs.rackspacecloud.com/auth/api/v2.0/RAX-KSKEY/RAX-
KSKEY-admin.wadl"/>
</ns5:extension>
```

Example 3.16. Extension Response: JSON

```
{
    "extension": {
        "updated": "2011-08-14T13:25:27-06:00",
        "alias": "RAX-KSKEY",
        "description": "Rackspace extension to OpenStack Identity v2.0 API
 enabling API Key authentication.",
        "name": "Rackspace API Key Authentication Admin Extension",
        "links": [
            {
                "rel": "describedby",
                "type": "application/xhtml+xml",
                "href": "http://docs.rackspace.com/auth/api/v2.0/auth-client-
devguide/content/QuickStart-000.html"
            },
            {
                "rel": "describedby",
                "type": "application/vnd.sun.wadl+xml",
                "href": "http://docs.rackspacecloud.com/auth/api/v2.0/RAX-
KSKEY/RAX-KSKEY-admin.wadl"
            }
        ],
        "namespace": "http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.
0"
    }
}
```

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 3.17 [32] and 3.18 [32] is defined in the RAX-META namespace. Extended headers are always prefixed with X- followed by the alias and a dash: (X-RAX-META-HEADER1). Parameters must be prefixed with the extension alias followed by a colon.

Important

Applications should be prepared to ignore response data that contains extension elements. Also, applications should verify that an extension is available before submitting an extended request.

Example 3.17. Extended User Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://docs.openstack.org/identity/api/v2.0"
      enabled="true" email="john.smith@example.org"
      id="u1000" username="jqsmith">
    <roles>
        <role tenantId="1234" id="Admin"/>
    </roles>
    <metadata
        xmlns="http://docs.rackspacecloud.com/identity/api/ext/meta/v2.0">
        <meta key="MetaKey1">MetaValue1</meta>
        <meta key="MetaKey2">MetaValue2</meta>
    </metadata>
</user>
```

Example 3.18. Extended User Response: JSON

```
{
  "user": {
    "roles": {
      "values": [
        {
          "tenantId": "1234",
          "id": "Admin"
        }
      ]
    },
    "id": "u1000",
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true,
    "RAX-META:metadata": {
      "values": {
        "MetaKey1": "MetaValue1",
        "MetaKey2": "MetaValue2"
      }
    }
  }
}
```

3.6. Namespaces

The API operations described in Chapter 4, *API Operations for Client Developers* [37] are enabled by a combination of the core Identity API and multiple API extensions. You can read more about extensions, including how to request details about all or specific extensions, at Section 3.5, "Extensions" [26].

Depending upon which operations you perform, you may see evidence of one or several extensions involved in processing your request. The most obvious evidence is the inclusion of namespaces in XML responses; you can identify the extensions involved in processing a request by examining the namespaces shown in the response.

A namespace is a container within which information has a specific meaning; in a different container, the meaning might be different. Specifying a namespace as the source of a definition is widely used as a way of choosing among multiple available meanings; context and scope are closely-related ideas.

In the example below, `xmlns` associates each namespace label (such as `ns1`) with a URL (such as `http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0`). Examining the list of namespaces shows that the core Identity API makes it possible to list `credentials` and the RAX-KSKEY extension adds to that by making it possible to list a specific kind of credentials, `apiKeyCredentials`.

Example 3.19. Multiple Namespaces in an XML Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:credentials❶
    xmlns:ns1="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
    xmlns:ns2="http://docs.openstack.org/identity/api/v2.0"
    xmlns:ns3="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"❷
    xmlns:ns4="http://docs.rackspace.com/identity/api/ext/RAX-KSQA/v1.0"
    xmlns:ns5="http://docs.openstack.org/common/api/v1.0"
    xmlns:ns6="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0"
    xmlns:ns7="http://docs.openstack.org/identity/api/ext/OS-KSCATALOG/v1.0"
    xmlns:ns8="http://www.w3.org/2005/Atom">
    <ns3:apiKeyCredentials❸
        username="bobbuilder"
        apiKey="0f97f489c848438090250d50c7e1eaXZ"/>
</ns2:credentials>
```

❶ The core Identity API, defined in `ns2`, supports `<credentials>`. *This association is valid only within this example; you may see namespaces identified differently in responses that you generate.*

❷ The RAX-KSKEY extension is defined in `ns3`. *This association is valid only within this example; you may see namespaces identified differently in responses that you generate.*

❸ Support for `<apiKeyCredentials>` is provided by the extension defined as `ns3`. *This association is valid only within this example; you may see namespaces identified differently in responses that you generate.*

If you are programmatically parsing responses that describe multiple namespaces, you should not assume that the prefixes that identify namespaces are constant. In this example, `ns2` identifies the namespace for the Identity API and `ns3` identifies the namespace for the RAX-KSKEY extension; under other circumstances, other prefixes could be associated with those namespaces.

3.7. Sub-Users

When an account is created, it is associated with one user. That user holds the `identity:user-admin` role, giving it full administrative privileges to manage the account. Exactly one administrative user exists for each account.

The account's administrative user can create and manage sub-users, also called sub-accounts. A sub-user is a child of the administrative user; a sub-user holds the `identity:default` role. Each sub-user shares its parent's tenant information, group memberships, and endpoints.

Each sub-user can act only upon itself. For example, a sub-user can change its own contact information and password, but it cannot make those changes for any other user. A sub-user can request a list of all users, but the sub-user will be the only user included in that list.

All users, including sub-users, can authenticate with a combination of their username and password. Only administrative users can authenticate with a combination of their username and API key. Only administrative users can authenticate with a combination of their tenant ID and API key.

A sub-user's group memberships matches the group memberships of its parent (the account's administrative user, holding the `identity:user-admin` role.

- When a new sub-user is created, it inherits its parent's groups.
- When a parent is assigned to a new group, its sub-users inherit that group.
- When a parent is removed from a group, its sub-users are removed, too.
- When a service attempts to assign a sub-user to a group, that attempt fails. A 400 error is returned, with the following message: "Cannot add sub-users directly to a group, must assign their parent user." To assign a sub-user to a group, assign its parent to that group; the sub-user will inherit the new group assignment from its parent.

3.8. Statuses

As you work with the Identity API, you may encounter two kinds of information identified as `status`:

- Multiple versions of the API exist. Each API version is marked with a `status`; you can use that status to help you choose which version of the API to execute. You can learn more about API versions in Section 3.4, “Versions” [21].
- Every account is marked with a `status`; based on that status, users associated with the account may be allowed or forbidden to perform certain operations. You can learn more about account statuses in this section.

The status of an account can change based on the account's activity. At any given time, every account has exactly one current status.

The following is a list of possible account statuses.

Table 3.4. Account Statuses

Status Name	Status Value	Notes
NEW	1	
PENDAPPROVAL	2	
ACTIVE	3	
DENIED	4	
DELINQUENT	5	
SUSPENDED	6	
AUPVIOLATION	7	This account is in violation of Rackspace's Acceptable Use Policy (AUP).
CLOSED	8	
UNVERIFIED	9	
PENDINGMIGRATION	10	
TERMINATED	11	

3.9. Limits

Calls to the Identity API are rate-limited at a default of 50 requests per second. Since tokens have an expiration time of 24 hours, you should never hit this limit.

The limit is placed on either the request's IP address, the username, or the token in the `X-Auth-Token` header, depending on the data available in the call's request. If your request is rejected because you have exceeded your rate limit, you will receive a 413 `overLimit` error. While you can retry after a few seconds, we also recommend that you check your client code to verify that it is using the API in an efficient manner.

A future release will provide the ability to view your limits so you can track API utilization before being rate-limited.

The rate limit defaults are likely to change as we gauge realistic levels. If the default limits change, we'll update this document to reflect the current value.

3.10. Faults

When an error occurs, the system returns an HTTP error response code denoting the type of error. The system returns additional information about the fault in the body of the response.

The following table lists possible fault types along with their associated error codes.

Table 3.5. Fault Types

Fault Element	Associated Error Codes	Expected in All Requests?
identityFault	500, 400	YES
serviceUnavailable	503	YES
badRequest	400	YES
unauthorized	401	YES
overLimit	413	YES

Fault Element	Associated Error Codes	Expected in All Requests?
userDisabled	403	
forbidden	403	
itemNotFound	404	
tenantConflict	409	

The root element of the fault may change depending on the type of error. From an XML schema perspective, all API faults are extensions of the base fault type, identityFault. When working with a system that binds XML to actual classes, use identityFault as a catch-all if there is no interest in distinguishing between individual fault types; for example, you may need to do this if you are working with JAXB.

The error code is returned in the body of the response. The message section returns a human-readable message. The details section is optional and may contain useful information for tracking down an error. For example, a stack trace may be provided.

The following are examples of an `itemNotFound` error.

Example 3.20. Item Not Found Fault: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<itemNotFound xmlns="http://docs.openstack.org/identity/api/v2.0"
              code="404">
 <message>Item not found.</message>
 <details>Error Details...</details>
</itemNotFound>
```

Example 3.21. Item Not Found Fault: JSON

```
{
  "itemNotFound": {
    "message": "Item not found.",
    "details": "Error Details...",
    "code": 404
  }
}
```

4. API Operations for Client Developers

Some of these operations require only the core authentication service API v2.0 and some require extensions to the API. If an extension is required, it is included in Rackspace's implementation of the Keystone Identity Service; issuing the calls as described here will use API extensions as needed.

Verb	URI	Description
Users		
GET	`v2.0/users`	List users.
GET	`v2.0/users?name=string`	Return detailed information about a specific user, by user name.
GET	`v2.0/users/{userId}`	Return detailed information about a specific user, by user ID.
POST	`v2.0/users`	Add a user.
POST	`v2.0/users/{userId}`	Update a user.
DELETE	`v2.0/users/{userId}`	Delete a user.
GET	`v2.0/users/{userId}/OS-KSADM/credentials`	List credentials, other than passwords, for all authentication methods.
GET	`v2.0/users/{userId}/OS-KSADM/credentials/RAX-KSKEY:apiKeyCredentials`	Get user credentials.
Roles		
GET	`v2.0/users/{userId}/roles`	Return global roles for a specific user.
Tokens		
POST	`v2.0/tokens`	Authenticate to generate a token.
Tenants		
GET	`v2.0/tenants`	Get a list of tenants.

4.1. Users

The operations described in this section allow clients to manage users.

Note

Some of the functionality described in this section is provided by the OS-KSADM and RAX-KSKEY extensions to the core Identity API. You can learn more about API extensions at https://github.com/openstack/identity-api/tree/master/openstack-identity-api/src/docbkx/.

When these requests are issued by a sub-user, they act only upon that sub-user. You can learn more about sub-users, also called sub-accounts, at Section 3.7, "Sub-Users" [34].

Verb	URI	Description
GET	`v2.0/users`	List users.
GET	`v2.0/users?name=string`	Return detailed information about a specific user, by user name.
GET	`v2.0/users/{userId}`	Return detailed information about a specific user, by user ID.

Verb	URI	Description
POST	`v2.0/users`	Add a user.
POST	`v2.0/users/{userId}`	Update a user.
DELETE	`v2.0/users/{userId}`	Delete a user.
GET	`v2.0/users/{userId}/OS-KSADM/credentials`	List credentials, other than passwords, for all authentication methods.
GET	`v2.0/users/{userId}/OS-KSADM/credentials/RAX-KSKEY:apiKeyCredentials`	Get user credentials.

4.1.1. List Users

Verb	URI	Description
GET	`v2.0/users`	List users.

Normal Response Code(s): 200, 203

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

Note

If you authenticated with the user role, you can see only yourself.

If this request is issued by a user holding the admin role (`identity:user-admin`), it returns a list of all users for the tenant.

If this request is issued by a user holding the user role (`identity:default`), it returns only the user who issued the request.

This request returns a list of users. The list includes identifying information about each user. Identifying information includes the user's email account, username, user ID, and status.

Table 4.1. List Users Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	You need a valid admin token for access. The `X-Auth-Token` header should always be supplied.

This operation does not require a request body.

Example 4.1. List Users Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<users xmlns="http://docs.openstack.org/identity/api/v2.0">
    <user xmlns="http://docs.openstack.org/identity/api/v2.0"
          enabled="true" email="john.smith@example.org"
          username="jqsmith" id="123456"/>
    <user xmlns="http://docs.openstack.org/identity/api/v2.0"
          enabled="true" email="john.smith2@example.org"
          username="jqsmith2" id="1234562"/>
</users>
```

Example 4.2. List Users Response: JSON

```
{
    "users":[{
            "id":"123456",
            "username":"jqsmith",
            "email":"john.smith@example.org",
            "enabled":true
        },
```

```
        {
            "id":"1234562",
            "username":"jqsmith2",
            "email":"john.smith2@example.org",
            "enabled":true
        }
    ],
    "users_links":[]
}
```

4.1.2. Get User by Name

Verb	URI	Description
GET	`v2.0/users?name=string`	Return detailed information about a specific user, by user name.

Normal Response Code(s): 200, 203

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

Note

If you authenticated with the user role, you can see only yourself.

If this request is issued by a user holding the admin role (`identity:user-admin`), the specific user's information is returned only if that user is associated with the same tenant as the requester's `user-admin` token.

If this request is issued by a user holding the user role (`identity:default`), information is returned for only the requester.

This request returns identifying information about the user with the specified user name. Identifying information includes the user's email account, username, user ID, status, and default region.

Default region, indicated by `defaultRegion`, associates this user with a specific regional datacenter: typical values include DFW, ORD and LON. A blank value is also acceptable, indicating that no default region has been assigned for this user. If a default region has been assigned for this user and a service in this user's service catalog is available in multiple regions, the user will obtain the service from the region specified by `defaultRegion`. You can see examples of services available in multiple regions in the "General API Information" chapter's "Sample Authentication Request and Response" section.

When a user creates a new cloud server via the Cloud Control Panel, the region specified by `defaultRegion` is visible on the control panel but the region cannot currently be changed there. To change the association of a user with a default region, use Update Users as you would to change any other descriptive information about the user.

In the examples below, `defaultRegion` is specified as `DFW`: when this user has a choice between consuming a service in the `DFW` region and consuming the same service in any other region, the user will consume the service in `DFW`.

Table 4.2. Get User by Name Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	Arbitrary character string generated by the authentication service in response to valid credentials. The `X-Auth-Token` header should always be supplied.
`name`	Query	String	The `name` parameter should always be supplied.

Example 4.3. Get User by Name Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://docs.openstack.org/identity/api/v2.0"
      xmlns:ns2="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
      xmlns:rax-auth="http://docs.rackspace.com/identity/api/ext/RAX-AUTH/v1.
0"
      id="123456" username="jqsmith"
      enabled="true"
      email="john.smith@example.org"
      rax-auth:defaultRegion="DFW">
</user>
```

Example 4.4. Get User by Name Response: JSON

```
{
  "user": {
    "RAX-AUTH:defaultRegion": "DFW"
    "id": "123456",
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true
  }
}
```

4.1.3. Get User by ID

Verb	URI	Description
GET	`v2.0/users/{userId}`	Return detailed information about a specific user, by user ID.

Normal Response Code(s): 200, 203

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

This request returns identifying information about the user with the specified user name. Identifying information includes the user's email account, username, user ID, status, and default region.

Default region, indicated by `defaultRegion`, associates this user with a specific regional datacenter: typical values include DFW, ORD and LON. A blank value is also acceptable, indicating that no default region has been assigned for this user. If a default region has been assigned for this user and a service in this user's service catalog is available in multiple regions, the user will obtain the service from the region specified by `defaultRegion`. You can see examples of services available in multiple regions in the "General API Information" chapter's "Sample Authentication Request and Response" section.

When a user creates a new cloud server via the Cloud Control Panel, the region specified by `defaultRegion` is visible on the control panel but the region cannot currently be changed there. To change the association of a user with a default region, use Update Users as you would to change any other descriptive information about the user.

In the examples below, `defaultRegion` is specified as `DFW`: when this user has a choice between consuming a service in the `DFW` region and consuming the same service in any other region, the user will consume the service in `DFW`.

Table 4.3. Get User by ID Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	Arbitrary character string generated by the authentication service in response to valid credentials. The `X-Auth-Token` header should always be supplied.
`userId`	Template	String	

This operation does not require a request body.

Example 4.5. Get User by ID Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://docs.openstack.org/identity/api/v2.0"
      xmlns:ns2="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
      xmlns:rax-auth="http://docs.rackspace.com/identity/api/ext/RAX-AUTH/v1.
0"
      id="123456" username="jqsmith"
      enabled="true"
```

```
      email="john.smith@example.org"
      rax-auth:defaultRegion="DFW">
</user>
```

Example 4.6. Get User by ID Response: JSON

```
{
  "user": {
    "RAX-AUTH:defaultRegion": "DFW"
    "id": "123456",
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true
  }
}
```

4.1.4. Add User

Verb	URI	Description
POST	`v2.0/users`	Add a user.

Normal Response Code(s): 201

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

Note

This call is available only to users who hold the admin role (`identity:user-admin`). Using the admin role, you can create a user who holds the user role (`identity:default`).

Warning

If you use this POST `/users` call to create a user, that user will have access to APIs but will not have access to the Cloud Control Panel. Cloud Control Panel access is provided to the account's admin user; the admin user is the user who initiated the account by completing the Sign-Up process.

To add a user, specify the user's `email` and `username` values, as well as whether the user's initial status is `enabled`. A successful response echoes the requested values and adds a value for `id`.

To add a user and specify an initial password for that user, include the password in a request resembling the "Add User with Password Request" examples shown below.

If you do not specify a password for the new user, one will be generated.

If an initial password is generated, it is included in a successful response to this request. After the user is created, the password cannot be retrieved by any means. Make note of the password in the response and supply that password to the user.

Within an account, a maximum of 100 sub-users (sub-accounts) can be added.

Table 4.4. Add User Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	You need a valid admin token for access. The `X-Auth-Token` header should always be supplied.

Example 4.7. Add User Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://docs.openstack.org/identity/api/v2.0"
      enabled="true" email="john.smith@example.org"
      username="jqsmith"/>
```

Example 4.8. Add User Request: JSON

```
{
  "user": {
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true
 }
}
```

Example 4.9. Add User with Password Request: XML

```
<user username="cmarin1subX2"
    email="cmarin1-sub@example.com"
    enabled="true"
    ns1:password="Password48"
    xmlns:ns1="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
    xmlns:ns2="http://docs.openstack.org/identity/api/v2.0" />
```

Example 4.10. Add User with Password Request: JSON

```
{
    "user": {
            "username": "cmarin1subX2",
            "email": "cmarin1-sub@example.com",
            "enabled": true,
            "OS-KSADM:password":"Password48"
        }
}
```

This operation does not require a request body.

Example 4.11. Add User Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://docs.openstack.org/identity/api/v2.0"
      xmlns:ns2="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
      xmlns:rax-auth="http://docs.rackspace.com/identity/api/ext/RAX-AUTH/v1.
0"
      id="123456" username="jqsmith"
      enabled="true"
      email="john.smith@example.org"
      rax-auth:defaultRegion="DFW">
</user>
```

Example 4.12. Add User Response: JSON

```
{
  "user": {
    "RAX-AUTH:defaultRegion": "DFW"
    "id": "123456",
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true
  }
}
```

4.1.5. Update User

Verb	URI	Description
POST	`v2.0/users/{userId}`	Update a user.

Normal Response Code(s): 200

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

Note

Users who hold the admin role can update users who hold the user role (`identity:default`) or the admin role (`identity:user-admin`) for the same tenant.

Warning

Before making any change to a user, you should confirm that you are changing the user you intend to change. You can do this by treating every change as a two-step operation: first, check the target of the change; second, make the change.

"Update User" acts upon the user specified by `userId`. Before attempting to update a user, use Get User by ID [43] to obtain the username, email account, and status associated with `userId`. That's Step 1. Check the returned information carefully; if you are certain that it describes the user you intend to change, proceed with Step 2, updating the user you just examined.

To update a user, specify the user's identifying information (`email`, `username`, `id`, `defaultRegion`, and status) values, replacing the current value with a new value where necessary. For example, to change the user's status from disabled to enabled, specify `enabled=true`. A successful response echoes the requested values.

To update a user's password, include the new password in a request resembling the "Update User Password Request" examples shown below.

Table 4.5. Update User Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	You need a valid admin token for access. The `X-Auth-Token` header should always be supplied.
`userId`	Template	String	

Example 4.13. Update User Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://docs.openstack.org/identity/api/v2.0"
      xmlns:ns2="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
      xmlns:rax-auth="http://docs.rackspace.com/identity/api/ext/RAX-AUTH/v1.
0"
```

```
      id="123456" username="jqsmith"
      enabled="true"
      email="john.smith@example.org"
      rax-auth:defaultRegion="DFW">
</user>
```

Example 4.14. Update User Request: JSON

```
{
  "user": {
    "RAX-AUTH:defaultRegion": "DFW"
    "id": "123456",
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true
  }
}
```

Example 4.15. Update User Password Request: XML

```
<user username="abc123"
    ns1:password="ungu355ab13"
    xmlns:ns1="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
    xmlns:ns2="http://docs.openstack.org/identity/api/v2.0" />
```

Example 4.16. Update User Password Request: JSON

```
{
    "user": {
            "username": "abc123",
            "OS-KSADM:password":"ungu355ab13"
        }
}
```

This operation does not require a request body.

Example 4.17. Update User Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://docs.openstack.org/identity/api/v2.0"
      xmlns:ns2="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
      xmlns:rax-auth="http://docs.rackspace.com/identity/api/ext/RAX-AUTH/v1.
0"
      id="123456" username="jqsmith"
      enabled="true"
      email="john.smith@example.org"
      rax-auth:defaultRegion="DFW">
</user>
```

Example 4.18. Update User Response: JSON

```
{
  "user": {
    "RAX-AUTH:defaultRegion": "DFW"
    "id": "123456",
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true
  }
```

```
}
```

4.1.6. Delete User

Verb	URI	Description
DELETE	`v2.0/users/{userId}`	Delete a user.

Normal Response Code(s): 204

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

Note

This call is available only to users who hold the admin role (`identity:user-admin`). Using the admin role, you can delete a user who holds the user role (`identity:default`) for the same tenant.

Warning

Before making any change to a user, you should confirm that you are changing the user you intend to change. You can do this by treating every change as a two-step operation: first, check the target of the change; second, make the change.

"Delete User" acts upon the user specified by `userId`. Before attempting to update a user, use Get User by ID [43] to obtain the username, email account, and status associated with `userId`. That's Step 1. Check the returned information carefully; if you are certain that it describes the user you intend to change, proceed with Step 2, deleting the user you just examined.

To delete a user, specify the user's ID in the request. If you know the user's name but not the user's ID, use Get User by Name [41] to obtain complete identifying information about the user.

Table 4.6. Delete User Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	You need a valid admin token for access. The `X-Auth-Token` header should always be supplied.
`userId`	Template	String	

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

4.1.7. List Credentials

Verb	URI	Description
GET	`v2.0/users/{userId}/OS-KSADM/credentials`	List credentials, other than passwords, for all authentication methods.

Normal Response Code(s): 200, 203

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

Note

You can see only yourself.

This call is available to users who hold either the admin role (`identity:user-admin`) or the user role (`identity:default`). In either case, the credentials returned are only those of the user who issued the request.

A user's password cannot be obtained by this or any other means. To list a user's non-password credentials for all authentication methods available to that user, specify the user's ID in the request. If you know the user's name but not the user's ID, use Get User by Name [41] to obtain complete identifying information about the user.

API credentials are currently not available for sub-users. Sub-users, also called sub-accounts, are users who were created by the administrative user. Sub-users can access API features with their password credentials. The administrative user holds the `identity:user-admin` role, giving it full administrative privileges to manage the account; sub-users hold the `identity:default` role and their privileges are limited.

Table 4.7. List Credentials Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	You need a valid admin token for access. The `X-Auth-Token` header should always be supplied.
`userId`	Template	String	

This operation does not require a request body.

Example 4.19. List Credentials Response: XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:credentials
    xmlns:ns1="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
    xmlns:ns2="http://docs.openstack.org/identity/api/v2.0"
    xmlns:ns3="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"
    xmlns:ns4="http://docs.rackspace.com/identity/api/ext/RAX-KSQA/v1.0"
    xmlns:ns5="http://docs.openstack.org/common/api/v1.0"
    xmlns:ns6="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0"
    xmlns:ns7="http://docs.openstack.org/identity/api/ext/OS-KSCATALOG/v1.0"
    xmlns:ns8="http://www.w3.org/2005/Atom">
    <ns3:apiKeyCredentials
```

```
        username="bobbuilder"
        apiKey="0f97f489c848438090250d50c7e1eaXZ"/>
</ns2:credentials>
```

Example 4.20. List Credentials Response: JSON

```
{
    "credentials": [
        {
            "RAX-KSKEY:apiKeyCredentials": {
                "apiKey": "0f97f489c848438090250d50c7e1eaXZ",
                "username": "bobbuilder"
            }
        }
    ]
}
```

4.1.8. Get User Credentials

Verb	URI	Description
GET	`v2.0/users/{userId}/OS-KSADM/credentials/RAX-KSKEY:apiKeyCredentials`	Get user credentials.

Normal Response Code(s): 200, 203

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

Note

You can see only yourself.

This call is available to users who hold either the admin role (`identity:user-admin`) or the user role (`identity:default`). In either case, the credentials returned are only those of the user who issued the request.

To list a user's API key credentials, specify the user's ID in the request. If you know the user's name but not the user's ID, use "Get User by Name" to obtain complete identifying information about the user.

Table 4.8. Get User Credentials Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	You need a valid admin token for access. The `X-Auth-Token` header should always be supplied.
`userId`	Template	String	

This operation does not require a request body.

Example 4.21. Get User Credentials Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<apiKeyCredentials
 xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"
 username="demoauthor"
 apiKey="aaaaa-bbbbb-ccccc-12345678"/>
```

Example 4.22. Get User Credentials Response: JSON

```
{
    "RAX-KSKEY:apiKeyCredentials":{
        "username":"demoauthor",
        "apiKey":"aaaaa-bbbbb-ccccc-12345678"
    }
}
```

4.2. Roles

The operations described in this section allow clients to manage roles.

Verb	URI	Description
GET	`v2.0/users/{userId}/roles`	Return global roles for a specific user.

4.2.1. List User Global Roles

Verb	URI	Description
GET	`v2.0/users/{userId}/roles`	Return global roles for a specific user.

Normal Response Code(s): 200, 203

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

Note

This call is available to users who hold either the admin role (`identity:user-admin`) or the user role (`identity:default`).

For users holding the `user` role, the call is valid only if `userID` and the `user-default` token have the same value for `tenant`.

This call returns a list of global roles associated with the user with the specified user ID. If you have only the user's name but not the user's ID, use "Get User by Name" to obtain complete identifying information about the user. For each role listed, the response includes identifying information such as the role's ID (such as `123`), name (such as `Admin`, and description (such as `All Access`). The list of global roles excludes any tenant roles associated with this user.

Table 4.9. List User Global Roles Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	Arbitrary character string generated by the authentication service in response to valid credentials. The `X-Auth-Token` header should always be supplied.
`userId`	Template	String	

This operation does not require a request body.

Example 4.23. List User Global Roles Response: XML

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

<roles xmlns="http://docs.openstack.org/identity/api/v2.0">
  <role id="123" name="Admin" description="All Access" />
  <role id="234" name="Guest" description="Guest Access" />
</roles>
```

Example 4.24. List User Global Roles Response: JSON

```
{
    "roles":[{
            "id":"123",
            "name":"compute:admin",
            "description":"Nova Administrator",
```

```
        }
    ],
    "roles_links":[]
}
```

4.3. Tokens

The operations described in this section allow clients to get and validate access tokens.

Verb	URI	Description
POST	`v2.0/tokens`	Authenticate to generate a token.

4.3.1. Authenticate

Verb	URI	Description
POST	`v2.0/tokens`	Authenticate to generate a token.

Normal Response Code(s): 200, 203

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

Note

This call is available to users who have authenticated as a holder of either the admin role or the user role.

- Users holding the `admin` role must authenticate as `identity:user-admin`.
- Users holding the `user` role must authenticate as `identity:default`.

This call returns a token if successful. Each ReST request requires the inclusion of a specific authorization token HTTP x-header, defined as `X-Auth-Token`. The token must be supplied for calls against other services and for other calls to the authentication service, such as the `GET /tenants` call. Clients obtain `X-Auth-Token` and the URL endpoints for other service APIs by supplying their valid credentials to the authentication service.

Client authentication is provided via a ReST interface using the POST method, with `v2.0/tokens` supplied as the path. A payload of credentials must be included in the body.

The authentication service is a ReSTful web service. It is the entry point to all service APIs. To access the authentication service, you must know its URL.

You can authenticate by providing any of several kinds of credentials:

- username and API key
- username, API key, and tenant ID
- username, API key, and tenant name
- username and password
- username, password and tenant ID
- username, password and tenant name
- token

You can use whichever valid credentials you prefer. Unless you specify a tenant, the authentication service's response is not affected by which kind of credentials you use.

Important

If you specify a tenant, the `Service Catalog` returned to you will include only endpoints to which that tenant has access.

To see authentication requests and responses with annotations explaining key ideas, see the examples in the "General API Information" chapter's "Sample Authentication Request and Response" section.

The examples below demonstrate authentication with several kinds of credentials.

Example 4.25. Authenticate (with Username and API Key Credentials) Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<auth>
  <apiKeyCredentials
    xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"
    username="demoauthor"
    apiKey="aaaaa-bbbbb-ccccc-12345678"/>
  </auth>
```

Example 4.26. Authenticate (with Username, API Key, and Tenant ID Credentials) Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<auth>
  <apiKeyCredentials
    xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"
    username="demoauthor"
    apiKey="aaaaa-bbbbb-ccccc-12345678"
    tenantId="1100111" />
</auth>
```

Example 4.27. Authenticate (with Username and Password Credentials) Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<auth xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://docs.openstack.org/identity/api/v2.0">
  <passwordCredentials username="demoauthor" password="theUsersPassword"/>
</auth>
```

Example 4.28. Authenticate (with Username, Password, and Tenant ID Credentials) Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<auth xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://docs.openstack.org/identity/api/v2.0">
  <passwordCredentials username="demoauthor" password="theUsersPassword"
 tenantId="1100111"/>
</auth>
```

Example 4.29. Authenticate (with Tenant ID and Token Credentials) Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<auth xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://docs.openstack.org/identity/api/v2.0"
 tenantId="1100111">
  <token id="vvvvvvvv-wwww-xxxx-yyyy-zzzzzzzzzzzz" />
</auth>
```

Example 4.30. Authenticate (with Tenant Name and Token Credentials) Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<auth xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://docs.openstack.org/identity/api/v2.0"
 tenantName="tenantabc">
  <token id="vvvvvvvv-wwww-xxxx-yyyy-zzzzzzzzzzzz" />
</auth>
```

Example 4.31. Authenticate (with Username and API Key Credentials) Request: JSON

```
{
    "auth": {
        "RAX-KSKEY:apiKeyCredentials": {
            "username": "demoauthor",
            "apiKey": "aaaaa-bbbbb-ccccc-12345678"
        }
    }
}
```

Example 4.32. Authenticate (with Username, API Key, and Tenant ID Credentials) Request: JSON

```
{
    "auth": {
        "RAX-KSKEY:apiKeyCredentials": {
            "username": "demoauthor",
            "apiKey": "aaaaa-bbbbb-ccccc-12345678"
        },
        "tenantId": "1100111"
    }
}
```

Example 4.33. Authenticate (with Username and Password Credentials) Request: JSON

```
{
    "auth":{
        "passwordCredentials":{
            "username":"demoauthor",
            "password":"theUsersPassword"
        }
    }
}
```

Example 4.34. Authenticate (with Username, Password, and Tenant ID Credentials) Request: JSON

```
{
    "auth":{
        "passwordCredentials":{
            "username":"demoauthor",
            "password":"theUsersPassword"
        },
        "tenantId": "12345678"
    }
```

```
}
```

Example 4.35. Authenticate (with Tenant ID and Token Credentials) Request: JSON

```
{
    "auth": {
        "tenantId": "1100111",
        "token": {
            "id": "vvvvvvvv-wwww-xxxx-yyyy-zzzzzzzzzzzz"
        }
    }
}
```

Example 4.36. Authenticate (with Tenant Name and Token Credentials) Request: JSON

```
{
    "auth": {
        "tenantName": "tenantabc",
        "token": {
            "id": "vvvvvvvv-wwww-xxxx-yyyy-zzzzzzzzzzzz"
        }
    }
}
```

Example 4.37. Authentication Response: XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<access
    xmlns:os-ksadm="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
    xmlns="http://docs.openstack.org/identity/api/v2.0"
    xmlns:rax-kskey="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.
0"
    xmlns:rax-ksqa="http://docs.rackspace.com/identity/api/ext/RAX-KSQA/v1.0"
    xmlns:common="http://docs.openstack.org/common/api/v1.0"
    xmlns:ksgrp="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0"
    xmlns:rax-kscatalog="http://docs.openstack.org/identity/api/ext/OS-
KSCATALOG/v1.0"
    xmlns:atom="http://www.w3.org/2005/Atom">
    <token
        id="aaaaa-bbbbb-ccccc-dddd"
        expires="2012-04-13T13:15:00.000-05:00"/>
    <user
        xmlns:rax-auth="http://docs.rackspace.com/identity/api/ext/RAX-AUTH/
v1.0"
        id="161418" name="demoauthor" rax-auth:defaultRegion="DFW">
        <roles>
            <role id="3" name="identity:user-admin"
                description="User Admin Role."/>
        </roles>
    </user>
    <serviceCatalog>
        <service type="rax:database" name="cloudDatabases">
            <endpoint region="DFW" tenantId="12345"
                publicURL="https://dfw.databases.api.rackspacecloud.com/v1.0/
12345"/>
            <endpoint region="ORD" tenantId="12345"
                publicURL="https://ord.databases.api.rackspacecloud.com/v1.0/
12345"/>
```

```
        </service>
        <service type="rax:load-balancer" name="cloudLoadBalancers">
            <endpoint region="ORD" tenantId="12345"
                publicURL="https://ord.loadbalancers.api.rackspacecloud.com/
v1.0/12345"/>
            <endpoint region="DFW" tenantId="12345"
                publicURL="https://dfw.loadbalancers.api.rackspacecloud.com/
v1.0/12345"/>
        </service>
        <service type="rax:object-cdn" name="cloudFilesCDN">
            <endpoint region="DFW" tenantId="MossoCloudFS_aaaa-bbbbbb-ccccc-
ddddd"
                publicURL="https://cdn1.clouddrive.com/v1/MossoCloudFS_aaaa-
bbbbbb-ccccc-ddddd"
            />
            <endpoint region="ORD" tenantId="MossoCloudFS_aaaa-bbbbbb-ccccc-
ddddd"
                publicURL="https://cdn2.clouddrive.com/v1/MossoCloudFS_aaaa-
bbbbbb-ccccc-ddddd"
            />
        </service>
        <service type="rax:monitor" name="cloudMonitoring">
            <endpoint tenantId="12345"
                publicURL="https://monitoring.api.rackspacecloud.com/v1.0/
12345"/>
        </service>
        <service type="object-store" name="cloudFiles">
            <endpoint region="DFW" tenantId="MossoCloudFS_aaaa-bbbbbb-ccccc-
ddddd"
                publicURL="https://storage101.dfw1.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbbbb-ccccc-ddddd"
                internalURL="https://snet-storage101.dfw1.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbbbb-ccccc-ddddd"
            />
            <endpoint region="ORD" tenantId="MossoCloudFS_aaaa-bbbbbb-ccccc-
ddddd"
                publicURL="https://storage101.ord1.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbbbb-ccccc-ddddd"
                internalURL="https://snet-storage101.ord1.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbbbb-ccccc-ddddd"
            />
        </service>
        <service type="compute" name="cloudServers">
            <endpoint tenantId="12345"
                publicURL="https://servers.api.rackspacecloud.com/v1.0/12345">
                <version id="1.0" info="https://servers.api.rackspacecloud.
com/v1.0"
                    list="https://servers.api.rackspacecloud.com/"/>
            </endpoint>
        </service>
        <service type="compute" name="cloudServersOpenStack">
            <endpoint region="DFW" tenantId="12345"
                publicURL="https://dfw.servers.api.rackspacecloud.com/v2/
12345">
                <version id="2" info="https://dfw.servers.api.rackspacecloud.
com/v2"
                    list="https://dfw.servers.api.rackspacecloud.com/"/>
            </endpoint>
            <endpoint region="ORD" tenantId="12345"
```

```
                publicURL="https://ord.servers.api.rackspacecloud.com/v2/
12345">
                <version id="2" info="https://ord.servers.api.rackspacecloud.
com/v2"
                    list="https://ord.servers.api.rackspacecloud.com/"/>
            </endpoint>
        </service>
        <service type="rax:dns" name="cloudDNS">
            <endpoint tenantId="12345"
                publicURL="https://dns.api.rackspacecloud.com/v1.0/12345"/>
        </service>
    </serviceCatalog>
</access>
```

Example 4.38. Authentication Response: JSON

```
{
    "access": {
        "serviceCatalog": [
            {
                "endpoints": [
                   {
                        "publicURL": "https://ord.servers.api.rackspacecloud.
com/v2/12345",
                        "region": "ORD",
                        "tenantId": "12345",
                        "versionId": "2",
                        "versionInfo": "https://ord.servers.api.
rackspacecloud.com/v2",
                        "versionList": "https://ord.servers.api.
rackspacecloud.com/"
                    }
                    {
                        "publicURL": "https://dfw.servers.api.rackspacecloud.
com/v2/12345",
                        "region": "DFW",
                        "tenantId": "12345",
                        "versionId": "2",
                        "versionInfo": "https://dfw.servers.api.
rackspacecloud.com/v2",
                        "versionList": "https://dfw.servers.api.
rackspacecloud.com/"
                    }
                ],
                "name": "cloudServersOpenStack",
                "type": "compute"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://ord.databases.api.
rackspacecloud.com/v1.0/12345",
                        "region": "ORD",
                        "tenantId": "12345"
                    },
                    {
                        "publicURL": "https://dfw.databases.api.
rackspacecloud.com/v1.0/12345",
                        "region": "DFW",
                        "tenantId": "12345"
```

```
                    }
                ],
                "name": "cloudDatabases",
                "type": "rax:database"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://ord.loadbalancers.api.
rackspacecloud.com/v1.0/12345",
                        "region": "ORD",
                        "tenantId": "645990"
                    },
                    {
                        "publicURL": "https://dfw.loadbalancers.api.
rackspacecloud.com/v1.0/12345",
                        "region": "DFW",
                        "tenantId": "12345"
                    }
                ],
                "name": "cloudLoadBalancers",
                "type": "rax:load-balancer"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://cdn1.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbb-cccc ",
                        "region": "DFW",
                        "tenantId": "MossoCloudFS_aaaa-bbbb-cccc "
                    }
                    {
                        "publicURL": "https://cdn2.clouddrive.com/v1/
MossoCloudFS_aaaa-bbbb-cccc ",
                        "region": "ORD",
                        "tenantId": "MossoCloudFS_aaaa-bbbb-cccc "
                    }
                ],
                "name": "cloudFilesCDN",
                "type": "rax:object-cdn"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://dns.api.rackspacecloud.com/v1.0/
12345",
                        "tenantId": "12345"
                    }
                ],
                "name": "cloudDNS",
                "type": "rax:dns"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://servers.api.rackspacecloud.com/
v1.0/12345",
                        "tenantId": "12345",
                        "versionId": "1.0",
```

```
                        "versionInfo": "https://servers.api.rackspacecloud.
com/v1.0",
                        "versionList": "https://servers.api.rackspacecloud.
com/"
                    }
                ],
                "name": "cloudServers",
                "type": "compute"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://monitoring.api.rackspacecloud.
com/v1.0/12345",
                        "tenantId": "12345"
                    }
                ],
                "name": "cloudMonitoring",
                "type": "rax:monitor"
            },
            {
                "endpoints": [
                    {
                        "internalURL": "https://snet-storage101.dfw1.
clouddrive.com/v1/MossoCloudFS_aaaa-bbbb-cccc ",
                        "publicURL": "https://storage101.dfw1.clouddrive.com/
v1/MossoCloudFS_aaaa-bbbb-cccc ",
                        "region": "DFW",
                        "tenantId": "MossoCloudFS_aaaa-bbbb-cccc"
                    }
                                        {
                        "internalURL": "https://snet-storage101.ord1.
clouddrive.com/v1/MossoCloudFS_aaaa-bbbb-cccc ",
                        "publicURL": "https://storage101.ord1.clouddrive.com/
v1/MossoCloudFS_aaaa-bbbb-cccc ",
                        "region": "ORD",
                        "tenantId": "MossoCloudFS_aaaa-bbbb-cccc"
                    }
                ],
                "name": "cloudFiles",
                "type": "object-store"
            }
        ],
        "token": {
            "expires": "2012-04-13T13:15:00.000-05:00",
            "id": "aaaaa-bbbbb-ccccc-dddd"
        },
        "user": {
        "RAX-AUTH:defaultRegion": "DFW",
            "id": "161418",
            "name": "demoauthor",
            "roles": [
                {
                    "description": "User Admin Role.",
                    "id": "3",
                    "name": "identity:user-admin"
                }
            ]
        }
    }
```

```
}
```

To see authentication requests and responses with annotations explaining key ideas, see the examples in the "General API Information" chapter's "Sample Authentication Request and Response" section.

4.4. Tenants

The operations described in this section allow clients to manage tenants.

Verb	URI	Description
GET	`v2.0/tenants`	Get a list of tenants.

4.4.1. Get Tenants

Verb	URI	Description
GET	`v2.0/tenants`	Get a list of tenants.

Normal Response Code(s): 200, 203

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

This call must be authenticated, so a valid `X-Auth-Token` must be passed in as a header. The operation returns a list of tenants for which the supplied token provides access.

Table 4.10. Get Tenants Request Parameters

Name	Style	Type	Description
`X-Auth-Token`	Header	String	Arbitrary character string generated by the authentication service in response to valid credentials. The `X-Auth-Token` header should always be supplied.

This operation does not require a request body.

Example 4.39. Get Tenants Response: XML

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=UTF-8
Content-Length: 200
Date: Sun, 1 Jan 2011 9:00:00 GMT

<?xml version="1.0" encoding="UTF-8"?>
<tenants xmlns="http://docs.openstack.org/identity/api/v2.0">
    <tenant enabled="true" id="1234" name="ACME Corp">
        <description>A description...</description>
    </tenant>
    <tenant enabled="true" id="3645" name="Iron Works">
        <description>A description...</description>
    </tenant>
</tenants>
```

Example 4.40. Get Tenants Response: JSON

```
{
    "tenants":[{
            "id":"1234",
            "name":"ACME Corp",
            "description":"A description ...",
            "enabled":true
        },
        {
            "id":"3456",
            "name":"Iron Works",
            "description":"A description ...",
            "enabled":true
```

```
        }
    ],
    "tenants_links":[]
}
```