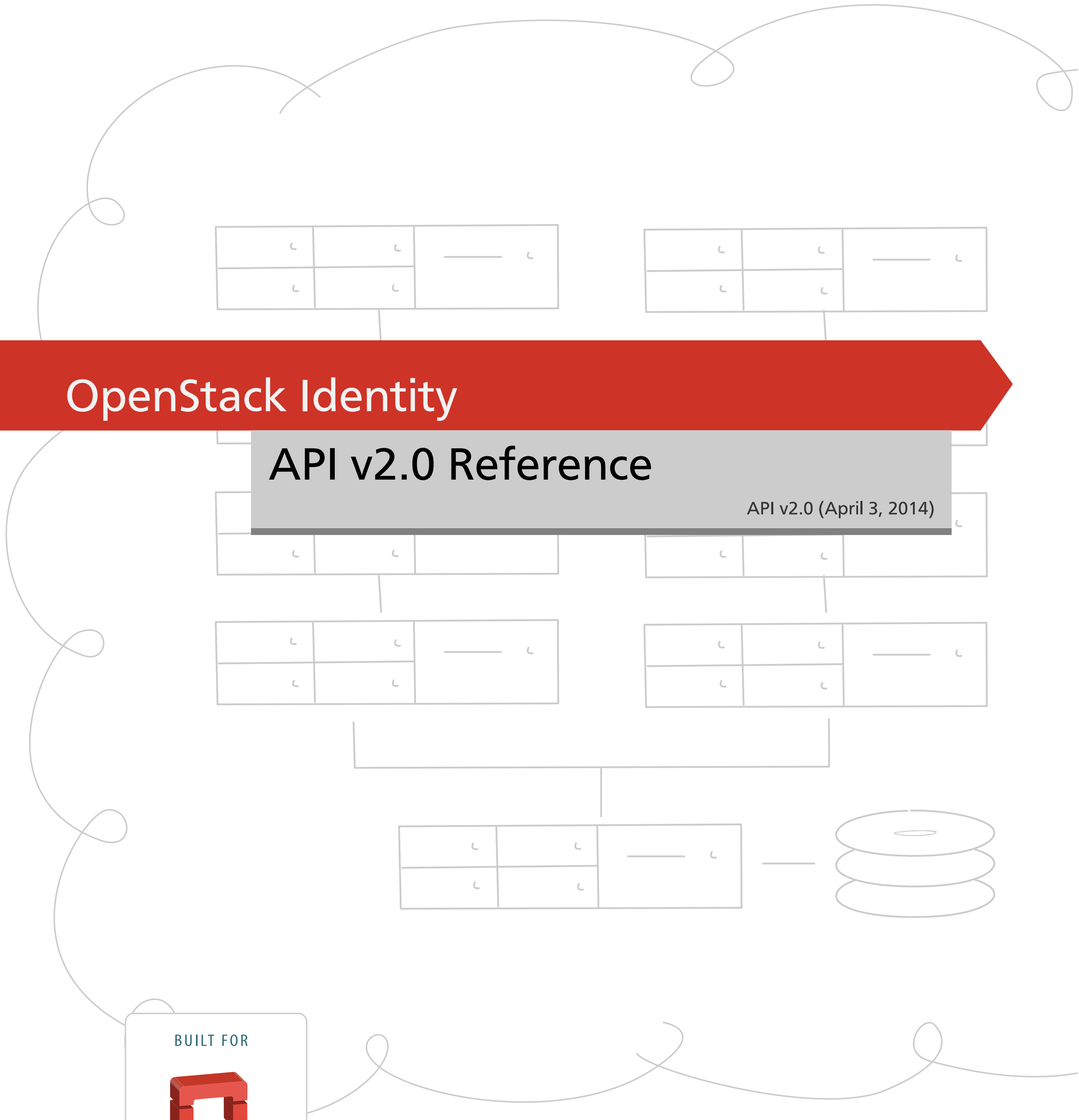

OpenStack Identity

API v2.0 Reference

API v2.0 (April 3, 2014)

docs.openstack.org

OpenStack Identity API v2.0 Reference

API v2.0 (2014-04-03)

This document describes how to develop applications that use the OpenStack Identity API v2.0 for authentication. This document also describes how to integrate services with the OpenStack Identity API v2.0.

Table of Contents

List of Tables

List of Examples

Preface

Table of Contents

OpenStack Identity allows clients to obtain tokens that can be used to access OpenStack cloud services.

Intended Audience

This reference is for software developers who develop applications that use the Identity API for authentication.

This reference assumes that the reader is familiar with RESTful web services, HTTP/1.1, and JSON and/or XML serialization formats.

Document Change History

This version of the reference replaces and obsoletes all previous versions. The following table describes recent changes:

Revision Date	Summary of Changes
July 13, 2013	• Added missing code samples and request parameter descriptions.
May 30, 2013	• Added back missing client operations and extensions.
September 13, 2011	• Initial release.

1. General API Information

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

Identity Concepts

OpenStack Identity has the following key concepts:

Table 1.1. Identity Concepts

Concept	Description
User	A digital representation of a person, system, or service that uses OpenStack cloud services. OpenStack Identity authentication services validate that an incoming request is being made by the user who claims to be making the call. Users have a login and may be assigned tokens to access resources. Users may be directly assigned to a particular tenant and behave as if they are contained in that tenant.
Credentials	Data that belongs to, is owned by, and generally only known by a user that the user can present to prove their identity. Examples include: • A matching username and password • A matching username and API key • A token that was issued to you
Authentication	In the context OpenStack Identity, the act of confirming the identity of a user or the truth of a claim. OpenStack Identity confirms that an incoming request is being made by the user who claims to be making the call by validating a set of claims that the user is making. These claims are initially in the form of a set of credentials (username & password, or username and API key). After initial confirmation, OpenStack Identity issues the user a token, which the user can then provide to demonstrate that their identity has been authenticated when making subsequent requests.
Token	An arbitrary bit of text that is used to access resources. Each token has a scope that describes which resources are accessible with it. A token may be revoked at anytime and is valid for a finite duration.

Concept	Description
	While OpenStack Identity supports token-based authentication in this release, the intention is for it to support additional protocols in the future. The intent is for it to be an integration service foremost, and not aspire to be a full-fledged identity store and management solution.
Tenant	A container used to group or isolate resources and/or identity objects. Depending on the service operator, a tenant can map to a customer, account, organization, or project.
Service	An OpenStack service, such as Compute (Nova), Object Storage (Swift), or Image Service (Glance). A service provides one or more endpoints through which users can access resources and perform operations.
Endpoint	A network-accessible address, usually described by a URL, where a service may be accessed. If using an extension for templates, you can create an endpoint template, which represents the templates of all the consumable services that are available across the regions.
Role	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 that role inherits those rights and privileges. In OpenStack Identity, a token that is issued to a user includes the list of roles that user can assume. Services that are being called by that user determine how they interpret the set of roles a user has and to which operations or resources each role grants access. It is up to individual services such as the Compute service and Image service to assign meaning to these roles. As far as the Identity service is concerned, a role is an arbitrary name assigned by the user.

Request/Response Types

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

Table 1.2. Response Types

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

Example 1.1. JSON Request with Headers

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

```
{
    "auth":{
        "passwordCredentials":{
            "username":"test_user",
            "password":"mypass"
        },
        "tenantName":"customer-x"
    }
}
```

Example 1.2. XML Response with Headers

```
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"?>
<access xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://docs.openstack.org/identity/api/v2.0">
    <token id="ab48a9efdfedb23ty3494" expires="2010-11-01T03:32:15-05:00">
        <tenant id="t1000" name="My Project" />
    </token>
    <user id="u123" name="jqsmith">
        <roles>
            <role id="100" name="compute:admin"/>
            <role id="101" name="object-store:admin" tenantId="t1000"/>
        </roles>
    </user>
    <serviceCatalog>
        <service type="compute" name="Cloud Servers">
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://compute.north.host.com/v1/t1000"
                internalURL="https://compute.north.host.internal/v1/t1000">
                <version
                id="1"
                info="https://compute.north.host.com/v1/"
                list="https://compute.north.host.com/"
                />
            </endpoint>
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://compute.north.host.com/v1.1/t1000"
                internalURL="https://compute.north.host.internal/v1.1/t1000">
                <version
                id="1.1"
                info="https://compute.north.host.com/v1.1/"
                list="https://compute.north.host.com/" />
            </endpoint>
        </service>
        <service type="object-store" name="Cloud Files">
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://storage.north.host.com/v1/t1000"
                internalURL="https://storage.north.host.internal/v1/t1000">
                <version
                id="1"
                info="https://storage.north.host.com/v1/"
                list="https://storage.north.host.com/" />
            </endpoint>
            <endpoint
        tenantId="t1000"
                region="South"
                publicURL="https://storage.south.host.com/v1/t1000"
                internalURL="https://storage.south.host.internal/v1/t1000">
                <version
                id="1"
                info="https://storage.south.host.com/v1/"
                list="https://storage.south.host.com/" />
            </endpoint>
        </service>
        <service type="dnsextension:dns" name="DNS-as-a-Service">
            <endpoint
        tenantId="t1000"
                publicURL="https://dns.host.com/v2.0/t1000">
                <version
                id="2.0"
                info="https://dns.host.com/v2.0/"
                list="https://dns.host.com/" />
            </endpoint>
        </service>
    </serviceCatalog>
</access>
```

Content Compression

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

Table 1.3. Compression Headers

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

Paginated Collections

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

Note

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

For convenience, collections contain atom "next" and "previous" links. The first page in the list will not contain a "previous" link, the last page in the list will not contain a "next" link. The following examples illustrate three pages in a collection of tenants. The first page was retrieved through a **GET** to http://identity.api.openstack.org/v2.0/1234/tenants?limit=1. In these examples, the *limit* parameter sets the page size to a single item. Subsequent "next" and "previous" links will honor the initial page size. Thus, a client may follow links to traverse a paginated collection without having to input the *marker* parameter.

Example 1.3. Tenant Collection, First Page: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<tenants xmlns="http://docs.openstack.org/identity/api/v2.0"
         xmlns:atom="http://www.w3.org/2005/Atom">
    <tenant enabled="true" id="1234" name="ACME Corp">
        <description>A description...</description>
    </tenant>
    <atom:link
        rel="next"
        href="http://identity.api.openstack.org/v2.0/tenants?limit=1&
amp;marker=1234"/>
</tenants>
```

Example 1.4. Tenant Collection, First Page: JSON

```
{
    "tenants":[{
            "id": "1234",
            "name": "ACME corp",
            "description": "A description ...",
            "enabled": true
        }
    ],
    "tenants_links":[{
            "rel": "next",
            "href": "http://identity.api.openstack.org/v2.0/tenants?limit=1&
marker=1234"
        }
    ]
}
```

Example 1.5. Tenant Collection, Second Page: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<tenants xmlns="http://docs.openstack.org/identity/api/v2.0"
         xmlns:atom="http://www.w3.org/2005/Atom">
    <tenant enabled="true" id="3645" name="Iron Works">
        <description>A description...</description>
    </tenant>
    <atom:link
        rel="previous"
        href="http://identity.api.openstack.org/v2.0/tenants?limit=1"/>
    <atom:link
        rel="next"
        href="http://identity.api.openstack.org/v2.0/tenants?limit=1&
amp;marker=3645"/>
</tenants>
```

Example 1.6. Tenant Collection, Second Page: JSON

```
{
    "tenants":[{
            "id": "3645",
            "name": "Iron Works",
            "description": "A description ...",
            "enabled": true
        }
    ],
    "tenants_links":[{
            "rel": "next",
            "href": "http://identity.api.openstack.org/v2.0/tenants?limit=1&
marker=3645"
        },
        {
            "rel": "previous",
            "href": "http://identity.api.openstack.org/v2.0/tenants?limit=1"
        }
    ]
}
```

Example 1.7. Tenant Collection, Last Page: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<tenants xmlns="http://docs.openstack.org/identity/api/v2.0"
         xmlns:atom="http://www.w3.org/2005/Atom">
    <tenant enabled="true" id="9999" name="Bigz">
        <description>A description...</description>
    </tenant>
    <atom:link
        rel="previous"
        href="http://identity.api.openstack.org/v2.0/tenants?limit=1&
amp;marker=1234"/>
</tenants>
```

Example 1.8. Tenant Collection, Last Page: JSON

```
{
    "tenants":[{
            "id": "9999",
            "name": "Bigz",
            "description": "A description ...",
            "enabled": true
        }
    ],
    "tenants_links":[{
            "rel": "previous",
            "href": "http://identity.api.openstack.org/v2.0/tenants?limit=1&
marker=1234"
        }
    ]
}
```

In the JSON representation, paginated collections contain a values property that contains the items in the collections. Links are accessed via the links property. The approach allows for extensibility of both the collection members and of the paginated collection itself. It also allows collections to be embedded in other objects as illustrated below. Here, a subset of groups are presented within a user. Clients must follow the "next" link to continue to retrieve additional groups belonging to a user.

Example 1.9. Paginated Roles in a User: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://docs.openstack.org/identity/api/v2.0"
      xmlns:atom="http://www.w3.org/2005/Atom"
      enabled="true" email="john.smith@example.org"
      username="jqsmith" id="u1000">
    <roles xmlns="http://docs.openstack.org/identity/api/ext/role">
        <role tenantId="1234" id="Admin"/>
        <role tenantId="1234" id="DBUser"/>
        <atom:link
            rel="next"
            href="http://identity.api.openstack.org/v2.0/tenants/1234/users/
u1000/groups?marker=Super"/>
    </roles>
</user>
```

Example 1.10. Paginated Roles in an User: JSON

```
{
    "user":{
        "OS-ROLE:roles":[{
                "tenantId": "1234",
                "id": "Admin"
            },
            {
                "tenantId": "1234",
                "id": "DBUser"
            }
        ],
        "OS-ROLE:roles_links":[{
                "rel": "next",
                "href": "http://identity.api.openstack.org/v2.0/tenants/1234/
users/u1000/roles?marker=Super"
            }
        ],
        "id": "u1000",
        "username": "jqsmith",
        "email": "john.smith@example.org",
        "enabled": true
    }
}
```

Versions

The OpenStack Identity API uses both a URI and a MIME type versioning scheme. In the URI scheme, the first element of the path contains the target version identifier (for example, https://identity.api.openstack.org/ v2.0/...). The MIME type versioning scheme uses HTTP content negotiation where the `Accept` or `Content-Type` headers contains a MIME type that includes the version ID as a parameter (application/vnd.openstack.identity +xml;version=1.1). A version MIME type is always linked to a base MIME type (application/ xml or application/json). If conflicting versions are specified using both an HTTP header and a URI, the URI takes precedence.

Example 1.11. Request with MIME type versioning

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

Example 1.12. Request with URI versioning

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

Note

The MIME type versioning approach allows for the creation of permanent links, because the version scheme is not specified in the URI path: https:// api.identity.openstack.org/tenants/12234.

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

Example 1.13. Multiple Choices Response: XML

```
<?xml version="1.0" encoding="utf-8"?>
<choices
            xmlns="http://docs.openstack.org/common/api/v1.0"
            xmlns:atom="http://www.w3.org/2005/Atom">
    <version id="v1.0" status="DEPRECATED">
        <media-types>
            <media-type
                    base="application/xml"
                    type="application/vnd.openstack.identity+xml;version=1.
0" />
```

```
            <media-type
                    base="application/json"
                    type="application/vnd.openstack.identity+json;version=1.
0" />
        </media-types>
        <atom:link rel="self" href="http://identity.api.openstack.org/v1.0" />
    </version>
    <version id="v1.1" status="CURRENT">
        <media-types>
            <media-type
                    base="application/xml"
                    type="application/vnd.openstack.identity+xml;version=1.
1" />
            <media-type
                    base="application/json"
                    type="application/vnd.openstack.identity+json;version=1.
1" />
        </media-types>
        <atom:link rel="self" href="http://identity.api.openstack.org/v1.1" />
    </version>
    <version id="v2.0" status="BETA">
        <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.openstack.org/v2.0" />
    </version>
</choices>
```

Example 1.14. Multiple Choices Response: JSON

```
{
    "choices":[{
            "id": "v1.0",
            "status": "DEPRECATED",
            "links":[{
                    "rel": "self",
                    "href": "http://identity.api.openstack.org/v1.0"
                }
            ],
            "media-types":{
                "values":[{
                        "base": "application/xml",
                        "type": "application/vnd.openstack.identity
+xml;version=1.0"
                    },
                    {
                        "base": "application/json",
                        "type": "application/vnd.openstack.identity
+json;version=1.0"
                    }
                ]
            }
        },
        {
            "id": "v1.1",
            "status": "CURRENT",
            "links":[{
                    "rel": "self",
                    "href": "http://identity.api.openstack.org/v1.1"
                }
            ],
            "media-types":{
                "values":[{
                        "base": "application/xml",
                        "type": "application/vnd.openstack.identity
+xml;version=1.1"
                    },
                    {
                        "base": "application/json",
                        "type": "application/vnd.openstack.identity
+json;version=1.1"
                    }
                ]
            }
        },
        {
            "id": "v2.0",
            "status": "BETA",
            "links":[{
                    "rel": "self",
                    "href": "http://identity.api.openstack.org/v2.0"
                }
            ],
            "media-types":{
                "values":[{
                        "base": "application/xml",
                        "type": "application/vnd.openstack.identity
+xml;version=2.0"
                    },
                    {
                        "base": "application/json",
                        "type": "application/vnd.openstack.identity
+json;version=2.0"
                    }
                ]
            }
        }
    ],
    "choices_links": ""
}
```

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

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

Example 1.15. Versions List Request

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

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 1.16. 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.openstack.org/v1.0/"/>
  </version>

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

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

</versions>
```

Example 1.17. 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.openstack.org/</id>
    <author><name>OpenStack</name><uri>http://www.openstack.org/</uri></
author>
    <link rel="self" href="http://identity.api.openstack.org/"/>
    <entry>
       <id>http://identity.api.openstack.org/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.openstack.org/v2.0/"/>
       <content type="text">Version v2.1 CURRENT (2011-05-27T20:22:02.25Z)</
content>
    </entry>
    <entry>
       <id>http://identity.api.openstack.org/v1.1/</id>
       <title type="text">Version v1.1</title>
       <updated>2010-12-12T18:30:02.25Z</updated>
       <link rel="self" href="http://identity.api.openstack.org/v1.1/"/>
       <content type="text">Version v1.1 CURRENT (2010-12-12T18:30:02.25Z)</
content>
    </entry>
    <entry>
       <id>http://identity.api.openstack.org/v1.0/</id>
       <title type="text">Version v1.0</title>
       <updated>2009-10-09T11:30:00Z</updated>
       <link rel="self" href="http://identity.api.openstack.org/v1.0/"/>
       <content type="text">Version v1.0 DEPRECATED (2009-10-09T11:30:00Z)</
content>
    </entry>
</feed>
```

Example 1.18. Versions List Response: JSON

```
{
    "versions":[{
            "id": "v1.0",
            "status": "DEPRECATED",
            "updated": "2009-10-09T11:30:00Z",
            "links":[{
                    "rel": "self",
                    "href": "http://identity.api.openstack.org/v1.0/"
                }
            ]
        },
        {
            "id": "v1.1",
            "status": "CURRENT",
            "updated": "2010-12-12T18:30:02.25Z",
            "links":[{
                    "rel": "self",
                    "href": "http://identity.api.openstack.org/v1.1/"
                }
            ]
        },
        {
            "id": "v2.0",
            "status": "BETA",
            "updated": "2011-05-27T20:22:02.25Z",
            "links":[{
                    "rel": "self",
                    "href": "http://identity.api.openstack.org/v2.0/"
                }
            ]
        }
    ],
    "versions_links":[]
}
```

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

Example 1.19. Version Details Request

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

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 1.20. 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.openstack.org/v2.0/"/>

    <atom:link rel="describedby"
               type="application/pdf"
               href="http://docs.openstack.org/identity/api/v2.0/identity-
latest.pdf" />

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

Example 1.21. 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.openstack.org/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.openstack.org/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.openstack.org/v2.0/"/>
      <link rel="describedby" type="application/pdf"
         href="http://docs.openstack.org/api/identity/api/v2.0/identity-
latest.pdf"/>
      <link rel="describedby" type="application/vnd.sun.wadl+xml"
           href="http://docs.openstack.org/identity/api/v2.0/application.
wadl"/>
      <content type="text">Version v2.0 CURRENT (2011-01-21T11:33:21-06:00)</
content>
   </entry>
</feed>
```

Example 1.22. 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.openstack.org/v2.0/"
      }, {
        "rel": "describedby",
        "type": "application/pdf",
        "href": "http://docs.openstack.org/api/identity/api/v2.0/identity-
latest.pdf"
      }, {
        "rel": "describedby",
        "type": "application/vnd.sun.wadl+xml",
        "href": "http://docs.openstack.org/identity/api/v2.0/identity.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.

Extensions

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

Verb	URI	Description
GET	/extensions	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. Additionally an extension contains documentation links in various formats.

Example 1.23. Extensions Response: XML

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

<extensions xmlns="http://docs.openstack.org/common/api/v1.0"
            xmlns:atom="http://www.w3.org/2005/Atom">
    <extension
        name="Reset Password Extension"
        namespace="http://docs.rackspacecloud.com/identity/api/ext/rpe/v1.0"
        alias="RS-RPE"
        updated="2011-01-22T13:25:27-06:00">

        <description>
            Adds the capability to reset a user's password.  The user is
            emailed when the password has been reset.
        </description>

        <atom:link rel="describedby"
                   type="application/pdf"
                   href="http://docs.rackspacecloud.com/identity/api/ext/
identity-rpe-20111111.pdf"/>
        <atom:link rel="describedby"
                   type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/identity/api/ext/
identity-rpe.wadl"/>
    </extension>
    <extension
        name="User Metadata Extension"
        namespace="http://docs.rackspacecloud.com/identity/api/ext/meta/v2.0"
        alias="RS-META"
        updated="2011-01-12T11:22:33-06:00">
        <description>
            Allows associating arbitrary metadata with a user.
        </description>

        <atom:link rel="describedby"
```

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

Example 1.24. Extensions Response: JSON

```
{
    "extensions":[{
            "name": "Reset Password Extension",
            "namespace": "http://docs.rackspacecloud.com/identity/api/ext/rpe/
v2.0",
            "alias": "RS-RPE",
            "updated": "2011-01-22T13:25:27-06:00",
            "description": "Adds the capability to reset a user's password.
 The user is emailed when the password has been reset.",
            "links":[{
                    "rel": "describedby",
                    "type": "application/pdf",
                    "href": "http://docs.rackspacecloud.com/identity/api/ext/
identity-rpe-20111111.pdf"
                },
                {
                    "rel": "describedby",
                    "type": "application/vnd.sun.wadl+xml",
                    "href": "http://docs.rackspacecloud.com/identity/api/ext/
identity-rpe.wadl"
                }
            ]
        },
        {
            "name": "User Metadata Extension",
            "namespace": "http://docs.rackspacecloud.com/identity/api/ext/
meta/v2.0",
            "alias": "RS-META",
            "updated": "2011-01-12T11:22:33-06:00",
            "description": "Allows associating arbritrary metadata with a
 user.",
            "links":[{
                    "rel": "describedby",
                    "type": "application/pdf",
                    "href": "http://docs.rackspacecloud.com/identity/api/ext/
identity-meta-20111201.pdf"
                },
                {
                    "rel": "describedby",
                    "type": "application/vnd.sun.wadl+xml",
                    "href": "http://docs.rackspacecloud.com/identity/api/ext/
identity-meta.wadl"
                }
            ]
        }
    ],
    "extensions_links":[]
}
```

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

Verb	URI	Description
GET	/extensions/*alias*	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 1.25. Extension Response: xml

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

<extension xmlns="http://docs.openstack.org/common/api/v1.0"
           xmlns:atom="http://www.w3.org/2005/Atom"
           name="User Metadata Extension"
           namespace="http://docs.rackspacecloud.com/identity/api/ext/meta/v2.
0"
           alias="RS-META"
           updated="2011-01-12T11:22:33-06:00">

    <description>
        Allows associating arbitrary metadata with a user.
    </description>

    <atom:link rel="describedby"
               type="application/pdf"
               href="http://docs.rackspacecloud.com/identity/api/ext/identity-
meta-20111201.pdf"/>
    <atom:link rel="describedby"
               type="application/vnd.sun.wadl+xml"
               href="http://docs.rackspacecloud.com/identity/api/ext/identity-
meta.wadl"/>

</extension>
```

Example 1.26. Extensions Response: JSON

```json
{
  "extension": {
    "name": "User Metadata Extension",
    "namespace": "http://docs.rackspacecloud.com/identity/api/ext/meta/v2.0",
    "alias": "RS-META",
    "updated": "2011-01-12T11:22:33-06:00",
    "description": "Allows associating arbritrary metadata with a user.",
    "links": [
      {
        "rel": "describedby",
        "type": "application/pdf",
        "href": "http://docs.rackspacecloud.com/identity/api/ext/identity-
meta-20111201.pdf"
      }, {
        "rel": "describedby",
        "type": "application/vnd.sun.wadl+xml",
        "href": "http://docs.rackspacecloud.com/identity/api/ext/identity-cbs.
wadl"
      }
    ]
  }
}
```

Extensions can 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 1.27 [18] and 1.28 [18] is defined in the `RS-META` namespace. Extended headers are always prefixed with `X-` followed by the alias and a dash: (`X-RS-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 also verify that an extension is available before submitting an extended request.

Example 1.27. 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">
    <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 1.28. Extended User Response: JSON

```
{
  "user": {
    "id": "1000",
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true,
    "RS-META:metadata": {
      "values": {
        "MetaKey1": "MetaValue1",
        "MetaKey2": "MetaValue2"
      }
    }
  }
}
```

Faults

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

Example 1.29. XML Fault Response

```
<?xml version="1.0" encoding="UTF-8"?>
<identityFault xmlns="http://docs.openstack.org/identity/api/v2.0"
          code="500">
    <message>Fault</message>
    <details>Error Details...</details>
</identityFault>
```

Example 1.30. JSON Fault Response

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

The error code is returned in the body of the response for convenience. The message section returns a human readable message. The details section is optional and may contain useful information for tracking down an error (such as, a stack trace).

The root element of the fault (for example, identityFault) may change depending on the type of error. The following is an example of an itemNotFound error.

Example 1.31. XML Not Found Fault

```
<?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 1.32. JSON Not Found Fault

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

The following list shows the possible fault types with associated error codes.

Table 1.4. Fault Types

Fault Element	Associated Error Code	Expected in All Requests
identityFault	500, 400	✓
serviceUnavailable	503	✓
badRequest	400	✓
unauthorized	401	✓
overLimit	413	

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

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 (such as JAXB), one should be capable of using identityFault as a “catch-all” if there's no interest in distinguishing between individual fault types.

2. Client API Operations

These operations enable clients to get API version and extension information, get authentication tokens, and list tenants.

Method	URI	Description
GET	`/v2.0/extensions`	Lists available extensions.
POST	`/v2.0/tokens`	Authenticates and generates a token.

List Extensions

Method	URI	Description
GET	`/v2.0/extensions`	Lists available extensions.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This operation does not require a request body.

Response

Example 2.1. List extensions: JSON response

```
{
    "extensions":[{
            "name": "Reset Password Extension",
            "namespace": "http://docs.rackspacecloud.com/identity/api/ext/rpe/
v2.0",
            "alias": "RS-RPE",
            "updated": "2011-01-22T13:25:27-06:00",
            "description": "Adds the capability to reset a user's password.
 The user is emailed when the password has been reset.",
            "links":[{
                    "rel": "describedby",
                    "type": "application/pdf",
                    "href": "http://docs.rackspacecloud.com/identity/api/ext/
identity-rpe-20111111.pdf"
                },
                {
                    "rel": "describedby",
                    "type": "application/vnd.sun.wadl+xml",
                    "href": "http://docs.rackspacecloud.com/identity/api/ext/
identity-rpe.wadl"
                }
            ]
        },
        {
            "name": "User Metadata Extension",
            "namespace": "http://docs.rackspacecloud.com/identity/api/ext/
meta/v2.0",
            "alias": "RS-META",
            "updated": "2011-01-12T11:22:33-06:00",
            "description": "Allows associating arbitrary metadata with a
 user.",
            "links":[{
                    "rel": "describedby",
                    "type": "application/pdf",
                    "href": "http://docs.rackspacecloud.com/identity/api/ext/
identity-meta-20111201.pdf"
                },
                {
                    "rel": "describedby",
                    "type": "application/vnd.sun.wadl+xml",
                    "href": "http://docs.rackspacecloud.com/identity/api/ext/
identity-meta.wadl"
                }
            ]
        }
    ],
    "extensions_links":[]
}
```

Example 2.2. List extensions: XML response

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

<extensions xmlns="http://docs.openstack.org/common/api/v1.0"
            xmlns:atom="http://www.w3.org/2005/Atom">
    <extension
        name="Reset Password Extension"
        namespace="http://docs.rackspacecloud.com/identity/api/ext/rpe/v1.0"
        alias="RS-RPE"
        updated="2011-01-22T13:25:27-06:00">

        <description>
            Adds the capability to reset a user's password.  The user is
            emailed when the password has been reset.
        </description>

        <atom:link rel="describedby"
                   type="application/pdf"
                   href="http://docs.rackspacecloud.com/identity/api/ext/
identity-rpe-20111111.pdf"/>
        <atom:link rel="describedby"
                   type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/identity/api/ext/
identity-rpe.wadl"/>
    </extension>
    <extension
        name="User Metadata Extension"
        namespace="http://docs.rackspacecloud.com/identity/api/ext/meta/v2.0"
        alias="RS-META"
        updated="2011-01-12T11:22:33-06:00">
        <description>
            Allows associating arbitrary metadata with a user.
        </description>

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

Authenticate

Method	URI	Description
POST	`/v2.0/tokens`	Authenticates and generates a token.

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

Each ReST request against the Identity API requires the X-Auth-Token header. Clients obtain this token, along with the URL to other service APIs, by first authenticating against the Identity API with valid credentials.

To authenticate, you must provide either a user ID and password or a token.

If the authentication token has expired, a 401 response code is returned.

If the token specified in the request has expired, this call returns a 404 response code.

The Identity API treats expired tokens as invalid tokens.

The deployment determines how long expired tokens are stored.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), userDisabled (403), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

Example 2.3. Authenticate with user name and password credentials: JSON request

```
{
    "auth":{
        "passwordCredentials":{
            "username":"test_user",
            "password":"mypass"
        },
        "tenantName":"customer-x"
    }
}
```

Example 2.4. Authenticate with user name and password credentials: XML request

```
<?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="customer-x">
  <passwordCredentials username="test_user" password="test"/>
</auth>
```

Response

Example 2.5. Authenticate with user name and password credentials: JSON response

```
{
    "access":{
        "token":{
            "id": "ab48a9efdfedb23ty3494",
            "expires": "2010-11-01T03:32:15-05:00",
            "tenant":{
                "id": "t1000",
                "name": "My Project"
            }
        },
        "user":{
            "id": "u123",
            "name": "jqsmith",
            "roles":[{
                    "id": "100",
                    "name": "compute:admin"
                },
                {
                    "id": "101",
                    "name": "object-store:admin",
                    "tenantId": "t1000"
                }
            ],
            "roles_links":[]
        },
        "serviceCatalog":[{
                "name": "Cloud Servers",
                "type": "compute",
                "endpoints":[{
                        "tenantId": "t1000",
                        "publicURL": "https://compute.north.host.com/v1/
t1000",
                        "internalURL": "https://compute.north.internal/v1/
t1000",
                        "region": "North",
                        "versionId": "1",
                        "versionInfo": "https://compute.north.host.com/v1/",
                        "versionList": "https://compute.north.host.com/"
                    },
                    {
                        "tenantId": "t1000",
                        "publicURL": "https://compute.north.host.com/v1.1/
t1000",
                        "internalURL": "https://compute.north.internal/v1.1/
t1000",
                        "region": "North",
                        "versionId": "1.1",
                        "versionInfo": "https://compute.north.host.com/v1.1/",
                        "versionList": "https://compute.north.host.com/"
                    }
                ],
                "endpoints_links":[]
            },
            {
                "name": "Cloud Files",
                "type": "object-store",
                "endpoints":[{
                        "tenantId": "t1000",
                        "publicURL": "https://storage.north.host.com/v1/
t1000",
                        "internalURL": "https://storage.north.internal/v1/
t1000",
                        "region": "North",
                        "versionId": "1",
                        "versionInfo": "https://storage.north.host.com/v1/",
                        "versionList": "https://storage.north.host.com/"
                    },
                    {
                        "tenantId": "t1000",
                        "publicURL": "https://storage.south.host.com/v1/
t1000",
                        "internalURL": "https://storage.south.internal/v1/
t1000",
                        "region": "South",
                        "versionId": "1",
                        "versionInfo": "https://storage.south.host.com/v1/",
                        "versionList": "https://storage.south.host.com/"
                    }
                ]
            },
            {
                "name": "DNS-as-a-Service",
                "type": "dnsextension:dns",
                "endpoints":[{
                        "tenantId": "t1000",
                        "publicURL": "https://dns.host.com/v2.0/t1000",
                        "versionId": "2.0",
                        "versionInfo": "https://dns.host.com/v2.0/",
                        "versionList": "https://dns.host.com/"
                    }
                ]
            }
        ]
    }
}
```

Example 2.6. Authenticate with user name and password credentials: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<access xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://docs.openstack.org/identity/api/v2.0">
    <token id="ab48a9efdfedb23ty3494" expires="2010-11-01T03:32:15-05:00">
        <tenant id="t1000" name="My Project" />
    </token>
    <user id="u123" name="jqsmith">
        <roles>
            <role id="100" name="compute:admin"/>
            <role id="101" name="object-store:admin" tenantId="t1000"/>
        </roles>
    </user>
    <serviceCatalog>
        <service type="compute" name="Cloud Servers">
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://compute.north.host.com/v1/t1000"
                internalURL="https://compute.north.host.internal/v1/t1000">
                <version
                id="1"
                info="https://compute.north.host.com/v1/"
                list="https://compute.north.host.com/"
                />
            </endpoint>
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://compute.north.host.com/v1.1/t1000"
                internalURL="https://compute.north.host.internal/v1.1/t1000">
                <version
                id="1.1"
                info="https://compute.north.host.com/v1.1/"
                list="https://compute.north.host.com/" />
            </endpoint>
        </service>
        <service type="object-store" name="Cloud Files">
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://storage.north.host.com/v1/t1000"
                internalURL="https://storage.north.host.internal/v1/t1000">
                <version
                id="1"
                info="https://storage.north.host.com/v1/"
                list="https://storage.north.host.com/" />
            </endpoint>
            <endpoint
        tenantId="t1000"
                region="South"
                publicURL="https://storage.south.host.com/v1/t1000"
                internalURL="https://storage.south.host.internal/v1/t1000">
                <version
                id="1"
                info="https://storage.south.host.com/v1/"
                list="https://storage.south.host.com/" />
            </endpoint>
        </service>
        <service type="dnsextension:dns" name="DNS-as-a-Service">
            <endpoint
        tenantId="t1000"
                publicURL="https://dns.host.com/v2.0/t1000">
                <version
                id="2.0"
                info="https://dns.host.com/v2.0/"
                list="https://dns.host.com/" />
            </endpoint>
        </service>
    </serviceCatalog>
</access>
```

3. Administrative API Operations

The OpenStack Identity administrative API operations enable service developers to get and validate access tokens, manage users, tenants, roles, and service endpoints.

Most administrative API calls require authentication. The only calls available without authentication are the calls to discover the service – getting version info, WADL contract, dev guide, help, and so on – and the call to authenticate and get a token.

Authentication is performed by passing in a valid token in the `X-Auth-Token` header on the request from the client. The Identity API will verify the token has (or belongs to a user that has) the `Admin` role.

See the README file or administrator guides for how to bootstrap the Identity API and create your first administrator.

Table 3.1. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

The OpenStack Identity administrative API v2.0 calls are:

Method	URI	Description
	Token Operations	
POST	`/v2.0/tokens`	Authenticates and generates a token.
GET	`/v2.0/tokens/{tokenId}{?belongsTo}`	Validates a token and confirms that it belongs to a specified tenant.
HEAD	`/v2.0/tokens/{tokenId}{?belongsTo}`	Validates a token and confirms that it belongs to a specified tenant, for performance.
GET	`/v2.0/tokens/{tokenId}/endpoints`	Lists the endpoints associated with a specified token.
	User Operations	
GET	`/v2.0/users/{?name}`	Gets detailed information about a specified user by user name.
GET	`/v2.0/users/{user_id}`	Gets detailed information about a specified user by user ID.
GET	`/v2.0/users/{user_id}/roles`	Lists global roles for a specified user. Excludes tenant roles.
	Tenant Operations	
GET	`/v2.0/tenants{?marker,limit}`	Lists tenants to which the specified token has access.
GET	`/v2.0/tenants{?marker,limit,name}`	Gets detailed information about a specified tenant by name.
GET	`/v2.0/tenants/{tenantId}`	Gets detailed information about a specified tenant by ID.
GET	`/v2.0/tenants/{tenantId}/users/{userId}/roles`	Lists roles for a specified user on a specified tenant. Excludes global roles.

Token Operations

Method	URI	Description
POST	`/v2.0/tokens`	Authenticates and generates a token.
GET	`/v2.0/tokens/{tokenId}{?belongsTo}`	Validates a token and confirms that it belongs to a specified tenant.
HEAD	`/v2.0/tokens/{tokenId}{?belongsTo}`	Validates a token and confirms that it belongs to a specified tenant, for performance.
GET	`/v2.0/tokens/{tokenId}/endpoints`	Lists the endpoints associated with a specified token.

Authenticate for Admin API

Method	URI	Description
POST	`/v2.0/tokens`	Authenticates and generates a token.

Client authentication is provided through a ReST interface by using the POST method with v2.0/tokens supplied as the path. Include a payload of credentials in the body.

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

Each ReST request against the Identity API requires the X-Auth-Token header. Clients obtain this token, along with the URL to other service APIs, by first authenticating against the Identity Service with valid credentials.

If the authentication token has expired, a 401 response code is returned.

If the subject token has expired, this call returns a 404 response code.

The Identity API treats expired tokens as invalid tokens.

The deployment determines how long expired tokens are stored.

As the following example responses show, the response to an authentication request returns the token ID in the `X-Subject-Token` header instead of in the token data.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), userDisabled (403), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

Example 3.1. Authenticate with credentials: JSON request

```
{
    "auth":{
        "passwordCredentials":{
            "username":"test_user",
            "password":"mypass"
        },
        "tenantName":"customer-x"
    }
}
```

Example 3.2. Authenticate with credentials: XML request

```
<?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="customer-x">
  <passwordCredentials username="test_user" password="test"/>
```

```
</auth>
```

Response

Example 3.3. Authenticate with credentials: JSON response

```
{
    "access":{
        "token":{
            "id": "ab48a9efdfedb23ty3494",
            "expires": "2010-11-01T03:32:15-05:00",
            "tenant":{
                "id": "t1000",
                "name": "My Project"
            }
        },
        "user":{
            "id": "u123",
            "name": "jqsmith",
            "roles":[{
                    "id": "100",
                    "name": "compute:admin"
                },
                {
                    "id": "101",
                    "name": "object-store:admin",
                    "tenantId": "t1000"
                }
            ],
            "roles_links":[]
        },
        "serviceCatalog":[{
                "name": "Cloud Servers",
                "type": "compute",
                "endpoints":[{
                        "tenantId": "t1000",
                        "publicURL": "https://compute.north.host.com/v1/
t1000",
                        "internalURL": "https://compute.north.internal/v1/
t1000",
                        "region": "North",
                        "versionId": "1",
                        "versionInfo": "https://compute.north.host.com/v1/",
                        "versionList": "https://compute.north.host.com/"
                    },
                    {
                        "tenantId": "t1000",
                        "publicURL": "https://compute.north.host.com/v1.1/
t1000",
                        "internalURL": "https://compute.north.internal/v1.1/
t1000",
                        "region": "North",
                        "versionId": "1.1",
                        "versionInfo": "https://compute.north.host.com/v1.1/",
                        "versionList": "https://compute.north.host.com/"
                    }
                ],
                "endpoints_links":[]
            },
            {
                "name": "Cloud Files",
                "type": "object-store",
                "endpoints":[{
                        "tenantId": "t1000",
                        "publicURL": "https://storage.north.host.com/v1/
t1000",
                        "internalURL": "https://storage.north.internal/v1/
t1000",
                        "region": "North",
                        "versionId": "1",
                        "versionInfo": "https://storage.north.host.com/v1/",
                        "versionList": "https://storage.north.host.com/"
                    },
                    {
                        "tenantId": "t1000",
                        "publicURL": "https://storage.south.host.com/v1/
t1000",
                        "internalURL": "https://storage.south.internal/v1/
t1000",
                        "region": "South",
                        "versionId": "1",
                        "versionInfo": "https://storage.south.host.com/v1/",
                        "versionList": "https://storage.south.host.com/"
                    }
                ]
            },
            {
                "name": "DNS-as-a-Service",
                "type": "dnsextension:dns",
                "endpoints":[{
                        "tenantId": "t1000",
                        "publicURL": "https://dns.host.com/v2.0/t1000",
                        "versionId": "2.0",
                        "versionInfo": "https://dns.host.com/v2.0/",
                        "versionList": "https://dns.host.com/"
                    }
                ]
            }
        ]
    }
}
```

Example 3.4. Authenticate with credentials: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<access xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://docs.openstack.org/identity/api/v2.0">
    <token id="ab48a9efdfedb23ty3494" expires="2010-11-01T03:32:15-05:00">
        <tenant id="t1000" name="My Project" />
    </token>
    <user id="u123" name="jqsmith">
        <roles>
            <role id="100" name="compute:admin"/>
            <role id="101" name="object-store:admin" tenantId="t1000"/>
        </roles>
    </user>
    <serviceCatalog>
        <service type="compute" name="Cloud Servers">
            <endpoint
        tenantId="t1000"
```

```
                region="North"
                publicURL="https://compute.north.host.com/v1/t1000"
                internalURL="https://compute.north.host.internal/v1/t1000">
                <version
                id="1"
                info="https://compute.north.host.com/v1/"
                list="https://compute.north.host.com/"
                />
            </endpoint>
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://compute.north.host.com/v1.1/t1000"
                internalURL="https://compute.north.host.internal/v1.1/t1000">
                <version
                id="1.1"
                info="https://compute.north.host.com/v1.1/"
                list="https://compute.north.host.com/" />
            </endpoint>
        </service>
        <service type="object-store" name="Cloud Files">
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://storage.north.host.com/v1/t1000"
                internalURL="https://storage.north.host.internal/v1/t1000">
                <version
                id="1"
                info="https://storage.north.host.com/v1/"
                list="https://storage.north.host.com/" />
            </endpoint>
            <endpoint
        tenantId="t1000"
                region="South"
                publicURL="https://storage.south.host.com/v1/t1000"
                internalURL="https://storage.south.host.internal/v1/t1000">
                <version
                id="1"
                info="https://storage.south.host.com/v1/"
                list="https://storage.south.host.com/" />
            </endpoint>
        </service>
        <service type="dnsextension:dns" name="DNS-as-a-Service">
            <endpoint
        tenantId="t1000"
                publicURL="https://dns.host.com/v2.0/t1000">
                <version
                id="2.0"
                info="https://dns.host.com/v2.0/"
                list="https://dns.host.com/" />
            </endpoint>
        </service>
    </serviceCatalog>
</access>
```

Validate Token

Method	URI	Description
GET	`/v2.0/tokens/{tokenId}{?belongsTo}`	Validates a token and confirms that it belongs to a specified tenant.

Returns the permissions relevant to a particular client. Valid tokens are in the `/tokens/{tokenId}` path. A user should expect an itemNotFound (`404`) fault for an token that is not valid.

Normal response codes: 200, 203

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

Request

This table shows the URI parameters for the validate token request:

Name	Type	Description
`{tokenId}`	UUID	Required. The token ID.

This operation does not require a request body.

Response

Example 3.5. Validate token: JSON response

```
{
   "access":{
      "token":{
         "id":"ab48a9efdfedb23ty3494",
         "expires":"2010-11-01T03:32:15-05:00",
         "tenant":{
            "id":"345",
            "name":"My Project"
         }
      },
      "user":{
         "id":"123",
         "name":"jqsmith",
         "roles":[
            {
               "id":"234",
               "name":"compute:admin"
            },
            {
               "id":"234",
               "name":"object-store:admin",
               "tenantId":"1"
            }
         ],
         "roles_links":[
         ]
      }
   }
}
```

Example 3.6. Validate token: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<access xmlns="http://docs.openstack.org/identity/api/v2.0">
        <token id="ab48a9efdfedb23ty3494" expires="2010-11-01T03:32:15-05:00">
                <tenant id="456" name="My Project" />
        </token>
        <user id="123" username="jqsmith">
                <roles>
                        <role id="123" name="Admin" tenantId="one" />
                        <role id="234" name="object-store:admin" tenantId=
"1" />
                </roles>
        </user>
</access>
```

Check Token

Method	URI	Description
HEAD	`/v2.0/tokens/{tokenId}{?belongsTo}`	Validates a token and confirms that it belongs to a specified tenant, for performance.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the URI parameters for the check token request:

Name	Type	Description
`{tokenId}`	UUID	Required. The token ID.

This operation does not require a request body.

List Endoints for Token

Method	URI	Description
GET	`/v2.0/tokens/{tokenId}/endpoints`	Lists the endpoints associated with a specified token.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the URI parameters for the list endoints for token request:

Name	Type	Description
`{tokenId}`	UUID	Required. The token ID.

This operation does not require a request body.

Response

Example 3.7. List endpoints for token: JSON response

```
{
  "endpoints": [
    {
      "name": "Nova",
      "adminURL": "http://admin.openstack/nova",
      "region": "north",
      "internalURL": "http://internal.openstack/nova",
      "type": "compute",
      "id": "8c3426bd730c48f5b59527df3a51b901",
      "publicURL": "http://public.openstack/nova"
    }
  ],
  "endpoints_links": []
}
```

Example 3.8. List endpoints for token: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoints xmlns="http://docs.openstack.org/identity/api/v2.0">
  <endpoint name="Nova" adminURL="http://admin.openstack/nova"
    region="north" internalURL="http://internal.openstack/nova"
    type="compute" id="8c3426bd730c48f5b59527df3a51b901"
    publicURL="http://public.openstack/nova"/>
</endpoints>
```

User Operations

Method	URI	Description
GET	`/v2.0/users/{?name}`	Gets detailed information about a specified user by user name.

Method	URI	Description
GET	`/v2.0/users/{user_id}`	Gets detailed information about a specified user by user ID.
GET	`/v2.0/users/{user_id}/roles`	Lists global roles for a specified user. Excludes tenant roles.

Get User Information by Name

Method	URI	Description
GET	`/v2.0/users/{?name}`	Gets detailed information about a specified user by user name.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get user information by name request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This operation does not require a request body.

Response

Example 3.9. Get User Information by Name: XML response

```
<?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" id="u1000"/>
```

Example 3.10. Get User Information by Name: JSON response

```
{
  "user": {
    "id": "u1000",
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true
  }
}
```

Get User Information by ID

Method	URI	Description
GET	`/v2.0/users/{user_id}`	Gets detailed information about a specified user by user ID.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get user information by id request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the get user information by id request:

Name	Type	Description
`{user_id}`	String	The user ID.

This operation does not require a request body.

Response

Example 3.11. Get User Information by ID: XML response

```
<?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" id="u1000"/>
```

Example 3.12. Get User Information by ID: JSON response

```
{
  "user": {
    "id": "u1000",
    "username": "jqsmith",
    "email": "john.smith@example.org",
    "enabled": true
  }
}
```

List User Global Roles

Method	URI	Description
GET	`/v2.0/users/{user_id}/roles`	Lists global roles for a specified user. Excludes tenant roles.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list user global roles request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the list user global roles request:

Name	Type	Description
`{user_id}`	String	The user ID.

This operation does not require a request body.

Response

Example 3.13. List user global roles: JSON response

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

Example 3.14. List user global roles: XML response

```
<?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>
```

Tenant Operations

Method	URI	Description
GET	`/v2.0/tenants{?marker,limit}`	Lists tenants to which the specified token has access.

Method	URI	Description
GET	`/v2.0/tenants{?marker,limit,name}`	Gets detailed information about a specified tenant by name.
GET	`/v2.0/tenants/{tenantId}`	Gets detailed information about a specified tenant by ID.
GET	`/v2.0/tenants/{tenantId}/users/{userId}/roles`	Lists roles for a specified user on a specified tenant. Excludes global roles.

List Tenants

Method	URI	Description
GET	`/v2.0/tenants{?marker,limit}`	Lists tenants to which the specified token has access.

```
GET /v2.0/tenants HTTP/1.1
Host: identity.api.openstack.org
Content-Type: application/json
X-Auth-Token: fa8426a0-8eaf-4d22-8e13-7c1b16a9370c
Accept: application/json
```

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list tenants request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This operation does not require a request body.

Response

Example 3.15. Get tenants: JSON response

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

Example 3.16. Get tenants: XML response

```
<?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>
```

Get Tenant Information by Name

Method	URI	Description
GET	`/v2.0/tenants{?marker,limit,name}`	Gets detailed information about a specified tenant by name.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get tenant information by name request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the query parameters for the get tenant information by name request:

Name	Type	Description
`marker`	String *(Optional)*	The ID of the last item in the previous list.
`limit`	Int *(Optional)*	The page size.
`name`	String *(Required)*	The name of the tenant.

Response

Example 3.17. Get tenant by name: JSON response

```
{
  "tenant": {
    "id": "1234",
    "name": "ACME corp",
    "description": "A description ...",
    "enabled": true
  }
}
```

Example 3.18. Get tenant by name: XML response

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

Get Tenant Information by ID

Method	URI	Description
GET	`/v2.0/tenants/{tenantId}`	Gets detailed information about a specified tenant by ID.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get tenant information by id request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the get tenant information by id request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.

This operation does not require a request body.

Response

Example 3.19. Get tenant by ID: JSON response

```
{
  "tenant": {
    "id": "1234",
    "name": "ACME corp",
    "description": "A description ...",
    "enabled": true
  }
}
```

Example 3.20. Get tenant by ID: XML response

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

List Roles for User

Method	URI	Description
GET	`/v2.0/tenants/{tenantId}/users/{userId}/roles`	Lists roles for a specified user on a specified tenant. Excludes global roles.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list roles for user request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the list roles for user request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.
`{userId}`	String	The user ID.

This operation does not require a request body.

Response

Example 3.21. List roles for user: JSON response

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

Example 3.22. List roles for user: XML response

```
<?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>
```

4. OpenStack Identity Extensions

Method	URI	Description
OS-KSADM Admin Extension		
OS-KSCATALOG Admin Extension		
OS-KSEC2 Admin Extension		
OS-KSS3 Admin Extension		
GET	`/v2.0/users/{userId}/OS-OS-KSS3/credentials{?marker,limit}`	Lists credentials.
POST	`/v2.0/users/{userId}/OS-OS-KSS3/credentials{?marker,limit}`	Adds a credential to a user.
GET	`/v2.0/users/{userId}/OS-OS-KSS3/credentials/OS-KSS3:s3Credentials`	Gets user credentials.
POST	`/v2.0/users/{userId}/OS-OS-KSS3/credentials/OS-KSS3:s3Credentials`	Updates credentials.
DELETE	`/v2.0/users/{userId}/OS-OS-KSS3/credentials/OS-KSS3:s3Credentials`	Deletes user credentials.
OS-KSVALIDATE Admin Extension		
GET	`/v2.0/OS-KSVALIDATE/token/validate{?belongsTo,HP-IDM-serviceId}`	Checks that a token is valid and that it belongs to a specified tenant and service IDs. Returns the permissions for a particular client.
HEAD	`/v2.0/OS-KSVALIDATE/token/validate{?belongsTo,HP-IDM-serviceId}`	Checks that a token is valid and that it belongs to a specified tenant and service IDs, for performance.
GET	`/v2.0/OS-KSVALIDATE/token/endpoints{?HP-IDM-serviceId}`	Lists endpoints associated with a specific token.

OS-KSADM Admin Extension

Extension operations.

Table 4.1. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

The following calls are supported by OS-KSADM-admin Extension:

Method	URI	Description
User Operations		
GET	`/v2.0/users`	Lists users.
POST	`/v2.0/users`	Adds a user.
POST	`/v2.0/users/{userId}`	Updates a user.
DELETE	`/v2.0/users/{userId}`	Deletes a user.
PUT	`/v2.0/users/{userId}/OS-KSADM/enabled`	Enables a specified user.
GET	`/v2.0/users/{userId}/roles{?serviceId}`	Lists global roles for a specified user.
PUT	`/v2.0/users/{userId}/roles/OS-KSADM/{roleId}`	Adds a specific global role to a user.
DELETE	`/v2.0/users/{userId}/roles/OS-KSADM/{roleId}`	Deletes a specific global role from a user.
POST	`/v2.0/users/{userId}/OS-KSADM/credentials`	Adds a credential to a user.
GET	`/v2.0/users/{userId}/OS-KSADM/credentials`	Lists credentials.
POST	`/v2.0/users/{userId}/OS-KSADM/credentials/{credential-type}`	Updates credentials for a specified user.
DELETE	`/v2.0/users/{userId}/OS-KSADM/credentials/{credential-type}`	Deletes credentials for a specified user.
GET	`/v2.0/users/{userId}/OS-KSADM/credentials/{credential-type}`	Gets user credentials.
Tenant Operations		
POST	`/v2.0/tenants`	Creates a tenant.
POST	`/v2.0/tenants/{tenantId}`	Updates a tenant.
DELETE	`/v2.0/tenants/{tenantId}`	Deletes a tenant.
GET	`/v2.0/tenants/{tenantId}/users{?marker,limit}`	Lists all the users for a tenant.
PUT	`/v2.0/tenants/{tenantId}/users/{userId}/roles/OS-KSADM/{roleId}`	Adds a specified role to a user for a tenant.
DELETE	`/v2.0/tenants/{tenantId}/users/{userId}/roles/OS-KSADM/{roleId}`	Deletes a specified role from a user on a tenant.
Role Operations		
GET	`/v2.0/OS-KSADM/roles{?name}`	Gets a role by name.
POST	`/v2.0/OS-KSADM/roles{?name}`	Adds a role.
GET	`/v2.0/OS-KSADM/roles/{roleId}`	Gets a role.
DELETE	`/v2.0/OS-KSADM/roles/{roleId}`	Deletes a role.
Service Operations		
GET	`/v2.0/OS-KSADM/services{?marker, limit}`	Lists services.
POST	`/v2.0/OS-KSADM/services{?marker, limit}`	Adds a service.
GET	`/v2.0/OS-KSADM/services/{serviceId}`	Gets a service.
DELETE	`/v2.0/OS-KSADM/services/{serviceId}`	Deletes a service.

User Operations

Method	URI	Description
GET	`/v2.0/users`	Lists users.
POST	`/v2.0/users`	Adds a user.
POST	`/v2.0/users/{userId}`	Updates a user.
DELETE	`/v2.0/users/{userId}`	Deletes a user.
PUT	`/v2.0/users/{userId}/OS-KSADM/enabled`	Enables a specified user.
GET	`/v2.0/users/{userId}/roles{?serviceId}`	Lists global roles for a specified user.
PUT	`/v2.0/users/{userId}/roles/OS-KSADM/{roleId}`	Adds a specific global role to a user.
DELETE	`/v2.0/users/{userId}/roles/OS-KSADM/{roleId}`	Deletes a specific global role from a user.
POST	`/v2.0/users/{userId}/OS-KSADM/credentials`	Adds a credential to a user.
GET	`/v2.0/users/{userId}/OS-KSADM/credentials`	Lists credentials.
POST	`/v2.0/users/{userId}/OS-KSADM/credentials/{credential-type}`	Updates credentials for a specified user.
DELETE	`/v2.0/users/{userId}/OS-KSADM/credentials/{credential-type}`	Deletes credentials for a specified user.
GET	`/v2.0/users/{userId}/OS-KSADM/credentials/{credential-type}`	Gets user credentials.

List Users

Method	URI	Description
GET	`/v2.0/users`	Lists users.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list users request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This operation does not require a request body.

Response

Example 4.1. List Users: XML response

```
<?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="u1000"/>
    <user xmlns="http://docs.openstack.org/identity/api/v2.0"
          enabled="true" email="john.smith@example.org"
          username="jqsmith" id="u1001"/>
</users>
```

Example 4.2. List Users: JSON response

```
{
    "users":[{
            "id": "u1000",
            "username": "jqsmith",
            "email": "john.smith@example.org",
            "enabled": true
        },
        {
            "id": "u1001",
            "username": "jqsmith",
            "email": "john.smith@example.org",
            "enabled": true
        }
    ],
    "users_links":[]
}
```

Add User

Method	URI	Description
POST	`/v2.0/users`	Adds a user.

Admin users can create a user with the user role. To add a user, specify a user name and initial password.

Admin users access the API through the admin endpoint, which uses port 35357 by default. Non-admin users access the API through the public endpoint, which uses port 5000 by default. These access ports are configurable.

A successful response echoes the requested values, except password, and generates an ID.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404), badMediaType (415)

Request

This table shows the header parameters for the add user request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

Example 4.3. Add User: JSON request

```
{
  "user": {
    "name": "jqsmith",
    "OS-KSADM:password": "secrete",
    "enabled": true,
    "tenantId": "af615b6af60b497e860392ad5e9f8dab",
    "email": "john.smith@example.org"
  }
}
```

Response

Example 4.4. Add User: JSON response

```
{
  "user": {
    "name": "jqsmith",
    "id": "84cdc81d5ba5448f885f857c3e7ae7dd",
    "enabled": true,
    "email": "john.smith@example.org",
    "tenantId": "af615b6af60b497e860392ad5e9f8dab"
```

```
   }
}
```

Update User

Method	URI	Description
POST	`/v2.0/users/{userId}`	Updates a user.

Normal response codes: 200

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the update user request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the update user request:

Name	Type	Description
`{userId}`	String	The user ID.

Example 4.5. Update User: XML request

```
<?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" id="u1000"/>
```

Example 4.6. Update User: JSON request

```
{
  "user": {
    "name": "jqsmith",
    "id": "84cdc81d5ba5448f885f857c3e7ae7dd",
    "enabled": true,
    "email": "john.smith@example.org",
    "tenantId": "af615b6af60b497e860392ad5e9f8dab"
  }
}
```

Response

Example 4.7. Update User: XML response

```
<?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" id="u1000"/>
```

Example 4.8. Update User: JSON response

```
{
  "user": {
    "name": "jqsmith",
    "id": "84cdc81d5ba5448f885f857c3e7ae7dd",
    "enabled": true,
    "email": "john.smith@example.org",
    "tenantId": "af615b6af60b497e860392ad5e9f8dab"
  }
}
```

Delete User

Method	URI	Description
DELETE	`/v2.0/users/{userId}`	Deletes a user.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the delete user request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the delete user request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Enable User

Method	URI	Description
PUT	`/v2.0/users/{userId}/OS-KSADM/enabled`	Enables a specified user.

Table 4.2. Enable User Request Attributes

Attribute	Type	Description
user	string	Required. The user name.

Normal response codes: 200

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the enable user request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the enable user request:

Name	Type	Description
`{userId}`	String	The user ID.

This table shows the body parameters for the enable user request:

Name	Type	Description
`user`	UserWith Only Enabled *(Optional)*	The user name.

Example 4.9. Enable User: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<user xmlns="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="UserWithOnlyEnabled"
 enabled="true"/>
```

Example 4.10. Enable User: JSON request

```
{
  "user": {
    "enabled": true
    }
}
```

Response

Example 4.11. Enable User: XML response

```
<?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" id="u1000"/>
```

Example 4.12. Enable User: JSON response

```
{
  "user": {
    "name": "jqsmith",
    "id": "84cdc81d5ba5448f885f857c3e7ae7dd",
    "enabled": true,
    "email": "john.smith@example.org",
    "tenantId": "af615b6af60b497e860392ad5e9f8dab"
  }
}
```

List Global Roles for User

Method	URI	Description
GET	`/v2.0/users/{userId}/roles{?serviceId}`	Lists global roles for a specified user.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list global roles for user request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the list global roles for user request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Response

Example 4.13. List Global Roles for User: XML response

```
<?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 4.14. List Global Roles for User: JSON response

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

Add Global Role to User

Method	URI	Description
PUT	`/v2.0/users/{userId}/roles/OS-KSADM/{roleId}`	Adds a specific global role to a user.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the add global role to user request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the add global role to user request:

Name	Type	Description
`{userId}`	String	The user ID.
`{roleId}`	String	The role ID.

This operation does not require a request body.

Delete Global Role from User

Method	URI	Description
DELETE	`/v2.0/users/{userId}/roles/OS-KSADM/{roleId}`	Deletes a specific global role from a user.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the delete global role from user request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the delete global role from user request:

Name	Type	Description
`{userId}`	String	The user ID.
`{roleId}`	String	The role ID.

This operation does not require a request body.

Add User Credentials

Method	URI	Description
POST	`/v2.0/users/{userId}/OS-KSADM/credentials`	Adds a credential to a user.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the add user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the add user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

Example 4.15. Add User Credentials: XML request

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

Example 4.16. Add User Credentials: JSON request

```
{
    "passwordCredentials": {
        "username": "test_user",
        "password": "mypass"
    }
}
```

Response

Example 4.17. Add User Credentials: XML response

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

Example 4.18. Add User Credentials: JSON response

```
{
    "passwordCredentials": {
        "username": "test_user",
        "password": "mypass"
    }
}
```

List Credentials

Method	URI	Description
GET	`/v2.0/users/{userId}/OS-KSADM/credentials`	Lists credentials.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the list credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Response

Example 4.19. List Credentials: XML response

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

Example 4.20. List Credentials: JSON response

```
{
    "credentials":[{
            "passwordCredentials":{
                "username": "test_user",
                "password": "mypass"
            }
        }
    ],
    "credentials_links":[]
}
```

Update User Credentials

Method	URI	Description
POST	`/v2.0/users/{userId}/OS-KSADM/credentials/{credential-type}`	Updates credentials for a specified user.

Normal response codes: 200

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the update user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the update user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.
`{credentialType}`	Extensible Credentials Type	The credential type.

Example 4.21. Update User Credentials: XML request

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

Example 4.22. Update User Credentials: JSON request

```
{
    "passwordCredentials": {
        "username": "test_user",
        "password": "mypass"
    }
}
```

Response

Example 4.23. Update User Credentials: XML response

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

Example 4.24. Update User Credentials: JSON response

```
{
    "passwordCredentials": {
        "username": "test_user",
        "password": "mypass"
    }
}
```

Delete User Credentials

Method	URI	Description
DELETE	`/v2.0/users/{userId}/OS-KSADM/credentials/{credential-type}`	Deletes credentials for a specified user.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the delete user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the delete user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.
`{credentialType}`	Extensible Credentials Type	The credential type.

This operation does not require a request body.

Get User Credentials

Method	URI	Description
GET	`/v2.0/users/{userId}/OS-KSADM/credentials/{credential-type}`	Gets user credentials.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the get user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.
`{credentialType}`	Extensible Credentials Type	The credential type.

This operation does not require a request body.

Response

Example 4.25. Get User Credentials: XML response

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

Example 4.26. Get User Credentials: JSON response

```
{
    "passwordCredentials": {
        "username": "test_user",
        "password": "mypass"
    }
}
```

Tenant Operations

Method	URI	Description
POST	`/v2.0/tenants`	Creates a tenant.

Method	URI	Description
POST	`/v2.0/tenants/{tenantId}`	Updates a tenant.
DELETE	`/v2.0/tenants/{tenantId}`	Deletes a tenant.
GET	`/v2.0/tenants/{tenantId}/users{?marker,limit}`	Lists all the users for a tenant.
PUT	`/v2.0/tenants/{tenantId}/users/{userId}/roles/OS-KSADM/{roleId}`	Adds a specified role to a user for a tenant.
DELETE	`/v2.0/tenants/{tenantId}/users/{userId}/roles/OS-KSADM/{roleId}`	Deletes a specified role from a user on a tenant.

Add Tenant

Method	URI	Description
POST	`/v2.0/tenants`	Creates a tenant.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415)

Request

This table shows the header parameters for the add tenant request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

Example 4.27. Add Tenant: XML request

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

Example 4.28. Add Tenant: JSON request

```
{
  "tenant": {
    "name": "ACME corp",
    "description": "A description ...",
    "enabled": true
  }
}
```

Response

Example 4.29. Add Tenant: XML response

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

Example 4.30. Add Tenant: JSON response

```
{
  "tenant": {
    "id": "1234",
    "name": "ACME corp",
    "description": "A description ...",
    "enabled": true
  }
}
```

Update Tenant

Method	URI	Description
POST	`/v2.0/tenants/{tenantId}`	Updates a tenant.

Normal response codes: 200

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404), badMediaType (415)

Request

This table shows the header parameters for the update tenant request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the update tenant request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.

Example 4.31. Update Tenant: XML request

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

Example 4.32. Update Tenant: JSON request

```
{
  "tenant": {
    "id": "1234",
    "name": "ACME corp",
    "description": "A description ...",
    "enabled": true
  }
}
```

Response

Example 4.33. Update Tenant: XML response

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

```
</tenant>
```

Example 4.34. Update Tenant: JSON response

```
{
  "tenant": {
    "id": "1234",
    "name": "ACME corp",
    "description": "A description ...",
    "enabled": true
  }
}
```

Delete Tenant

Method	URI	Description
DELETE	`/v2.0/tenants/{tenantId}`	Deletes a tenant.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the delete tenant request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the delete tenant request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.

This operation does not require a request body.

List Users for a Tenant

Method	URI	Description
GET	`/v2.0/tenants/{tenantId}/users{?marker,limit}`	Lists all the users for a tenant.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list users for a tenant request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the list users for a tenant request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.

This operation does not require a request body.

Response

Example 4.35. List Users for a Tenant: XML response

```
<?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="u1000"/>
    <user xmlns="http://docs.openstack.org/identity/api/v2.0"
          enabled="true" email="john.smith@example.org"
          username="jqsmith" id="u1001"/>
</users>
```

Example 4.36. List Users for a Tenant: JSON response

```
{
    "users":[{
            "id": "u1000",
            "username": "jqsmith",
            "email": "john.smith@example.org",
            "enabled": true
        },
        {
            "id": "u1001",
            "username": "jqsmith",
            "email": "john.smith@example.org",
            "enabled": true
        }
    ],
    "users_links":[]
}
```

Add Roles to User on Tenant

Method	URI	Description
PUT	`/v2.0/tenants/{tenantId}/users/{userId}/roles/OS-KSADM/{roleId}`	Adds a specified role to a user for a tenant.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the add roles to user on tenant request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the add roles to user on tenant request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.
`{userId}`	String	The user ID.
`{roleId}`	String	The role ID.

This operation does not require a request body.

Delete Roles from User on Tenant

Method	URI	Description
DELETE	`/v2.0/tenants/{tenantId}/users/{userId}/roles/OS-KSADM/{roleId}`	Deletes a specified role from a user on a tenant.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the delete roles from user on tenant request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the delete roles from user on tenant request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.
`{userId}`	String	The user ID.
`{roleId}`	String	The role ID.

This operation does not require a request body.

Role Operations

Method	URI	Description
GET	`/v2.0/OS-KSADM/roles{?name}`	Gets a role by name.
POST	`/v2.0/OS-KSADM/roles{?name}`	Adds a role.
GET	`/v2.0/OS-KSADM/roles/{roleId}`	Gets a role.
DELETE	`/v2.0/OS-KSADM/roles/{roleId}`	Deletes a role.

Get Role by Name

Method	URI	Description
GET	`/v2.0/OS-KSADM/roles{?name}`	Gets a role by name.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the get role by name request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This operation does not require a request body.

Response

This table shows the header parameters for the get role by name response:

Name	Type	Description
`Location`	AnyURI *(Optional)*	The location.

Example 4.37. Get Role by Name: XML response

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

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

Example 4.38. Get Role by Name: JSON response

```
{
  "role": {
    "id": "123",
    "name": "Guest",
    "description": "Guest Access"
  }
}
```

Add Role

Method	URI	Description
POST	`/v2.0/OS-KSADM/roles{?name}`	Adds a role.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the add role request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

Example 4.39. Add Role: XML request

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

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

Example 4.40. Add Role: JSON request

```
{
  "role": {
    "id": "123",
    "name": "Guest",
    "description": "Guest Access"
  }
}
```

Response

This table shows the header parameters for the add role response:

Name	Type	Description
`Location`	AnyURI *(Optional)*	The location.

Example 4.41. Add Role: XML response

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

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

Example 4.42. Add Role: JSON response

```
{
  "role": {
    "id": "123",
    "name": "Guest",
    "description": "Guest Access"
  }
}
```

Get Role

Method	URI	Description
GET	`/v2.0/OS-KSADM/roles/{roleId}`	Gets a role.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the get role request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the get role request:

Name	Type	Description
`{roleId}`	String	The role ID.

This operation does not require a request body.

Response

This table shows the header parameters for the get role response:

Name	Type	Description
`Location`	AnyURI *(Optional)*	The location.

Example 4.43. Get Role: XML response

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

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

Example 4.44. Get Role: JSON response

```
{
  "role": {
    "id": "123",
    "name": "Guest",
    "description": "Guest Access"
  }
}
```

Delete Role

Method	URI	Description
DELETE	`/v2.0/OS-KSADM/roles/{roleId}`	Deletes a role.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the delete role request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the delete role request:

Name	Type	Description
`{roleId}`	String	The role ID.

This operation does not require a request body.

Service Operations

Method	URI	Description
GET	`/v2.0/OS-KSADM/services{?marker, limit}`	Lists services.
POST	`/v2.0/OS-KSADM/services{?marker, limit}`	Adds a service.
GET	`/v2.0/OS-KSADM/services/ {serviceId}`	Gets a service.
DELETE	`/v2.0/OS-KSADM/services/ {serviceId}`	Deletes a service.

List Services

Method	URI	Description
GET	`/v2.0/OS-KSADM/services{?marker, limit}`	Lists services.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list services request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This operation does not require a request body.

Response

Example 4.45. List Services: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<services
  xmlns="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0">
  <service id="123" name="nova" type="compute"
    description="OpenStack Compute Service"/>
  <service id="234" name="glance" type="image"
    description="OpenStack Image Service"/>
</services>
```

Example 4.46. List Services: JSON response

```
{
    "OS-KSADM:services":[{
            "id": "123",
            "name": "nova",
            "type": "compute",
            "description": "OpenStack Compute Service"
        },
        {
            "id": "234",
            "name": "glance",
            "type": "image",
            "description": "OpenStack Image Service"
        }
    ],
    "OS-KSADM:services_links":[]
}
```

Add Service

Method	URI	Description
POST	`/v2.0/OS-KSADM/services{?marker, limit}`	Adds a service.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the add service request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

Example 4.47. Add Service: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<service
    xmlns="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
    id="123" name="nova" type="compute"
    description="OpenStack Compute Service"/>
```

Example 4.48. Add Service: JSON request

```
{
    "OS-KSADM:service":{
        "id": "123",
        "name": "nova",
        "type": "compute",
        "description": "OpenStack Compute Service"
    }
}
```

Response

This table shows the header parameters for the add service response:

Name	Type	Description
`Location`	AnyURI *(Optional)*	The location.

Example 4.49. Add Service: XML response

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

```
<service
    xmlns="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
    id="123" name="nova" type="compute"
    description="OpenStack Compute Service"/>
```

Example 4.50. Add Service: JSON response

```
{
    "OS-KSADM:service":{
        "id": "123",
        "name": "nova",
        "type": "compute",
        "description": "OpenStack Compute Service"
    }
}
```

Get Service

Method	URI	Description
GET	`/v2.0/OS-KSADM/services/{serviceId}`	Gets a service.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get service request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the get service request:

Name	Type	Description
`{serviceId}`	String	The service ID.

This operation does not require a request body.

Response

Example 4.51. Get Service: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<service
    xmlns="http://docs.openstack.org/identity/api/ext/OS-KSADM/v1.0"
    id="123" name="nova" type="compute"
    description="OpenStack Compute Service"/>
```

Example 4.52. Get Service: JSON response

```
{
    "OS-KSADM:service":{
        "id": "123",
        "name": "nova",
        "type": "compute",
        "description": "OpenStack Compute Service"
    }
}
```

Delete Service

Method	URI	Description
DELETE	`/v2.0/OS-KSADM/services/{serviceId}`	Deletes a service.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the delete service request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`Content-Type`	String *(Required)*	"application/json" or "application/xml"

This table shows the URI parameters for the delete service request:

Name	Type	Description
`{serviceId}`	String	The service ID.

This operation does not require a request body.

OS-KSCATALOG Admin Extension

Table 4.3. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

The OS-KSCATALOG extension extends the OpenStack Identity Admin API v2.0 with the following calls:

Method	URI	Description
Endpoint Template Operations		
GET	`/v2.0/OS-KSCATALOG/endpointTemplates{?serviceId}`	Lists endpoint templates.
POST	`/v2.0/OS-KSCATALOG/endpointTemplates{?serviceId}`	Adds endpoint template.
GET	`/v2.0/OS-KSCATALOG/endpointTemplates/{endpointTemplateId}`	Gets endpoint templates.
DELETE	`/v2.0/OS-KSCATALOG/endpointTemplates/{endpointTemplateId}`	Deletes an endpoint template.
Endpoint Operations		
GET	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints`	Lists endpoints for a tenant.
POST	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints`	Adds endpoint to a tenant.
GET	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints/{endpointId}`	Gets endpoint for a tenant.
DELETE	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints/{endpointId}`	Deletes an endpoint from a tenant.

Endpoint Template Operations

Method	URI	Description
GET	`/v2.0/OS-KSCATALOG/endpointTemplates{?serviceId}`	Lists endpoint templates.
POST	`/v2.0/OS-KSCATALOG/endpointTemplates{?serviceId}`	Adds endpoint template.
GET	`/v2.0/OS-KSCATALOG/endpointTemplates/{endpointTemplateId}`	Gets endpoint templates.
DELETE	`/v2.0/OS-KSCATALOG/endpointTemplates/{endpointTemplateId}`	Deletes an endpoint template.

List Endpoint Templates

Method	URI	Description
GET	`/v2.0/OS-KSCATALOG/endpointTemplates{?serviceId}`	Lists endpoint templates.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list endpoint templates request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This operation does not require a request body.

Response

Example 4.53. List Endpoint Templates: XML response

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

<endpointTemplates xmlns="http://docs.openstack.org/identity/api/ext/OS-
KSCATALOG/v1.0">
  <endpointTemplate
   id="1"
   region="North"
   global="true"
   type="compute"
   name="Compute"
   publicURL="https://compute.north.public.com/v1"
   internalURL="https://compute.north.internal.com/v1"
   enabled="true">
   <version
     id="1"
     list="https://compute.north.public.com/"
     info="https://compute.north.public.com/v1"/>
  </endpointTemplate>
  <endpointTemplate
   id="2"
   region="south"
   type="compute"
   name="Compute"
   publicURL="https://service2.public.com/v1"
   internalURL="https://service2.internal.public.com/v1"
   enabled="false">
   <version
    id="1"
    list="https://service1.public.com/"
    info="https://service1.public.com/v1"/>
  </endpointTemplate>
  <endpointTemplate
   id="3"
   region="DFW"
   global="true"
   type="ext1:service1"
   name="Compute"
   publicURL="https://service1.public.com/v1"
   enabled="true">
   <version
    id="1"
    list="https://service1.public.com/"
    info="https://service1.public.com/v1"/>
  </endpointTemplate>
  <endpointTemplate
   id="4"
   region="ORD"
   type="compute"
   name="Compute"
   publicURL="https://service2.public.com/v1"
   enabled="true">
   <version
    id="1"
    list="https://service1.public.com/"
    info="https://service1.public.com/v1"/>
  </endpointTemplate>
  <endpointTemplate
   id="5"
   global="true"
   type="compute"
   name="Compute"
   publicURL="https://service3.public.com/v1">
   <version
    id="1"
    list="https://service1.public.com/"
    info="https://service1.public.com/v1"/>
  </endpointTemplate>
</endpointTemplates>
```

Example 4.54. List Endpoint Templates: JSON response

```
{
    "OS-KSCATALOG:endpointsTemplates": [
            {
                "id": 1,
                "region": "North",
                "global": true,
                "type": "compute",
                "publicURL": "https://compute.north.public.com/v1",
                "internalURL": "https://compute.north.internal.com/v1",
                "versionId": "1",
                "versionInfo": "https://compute.north.public.com/v1/",
                "versionList": "https://compute.north.public.com/",
                "enabled": true
            },
            {
                "id": 2,
                "region": "South",
                "type": "compute",
                "publicURL": "https://compute.south.public.com/v1",
                "internalURL": "https://compute.south.internal.com/v1",
                "versionId": "1",
                "versionInfo": "https://compute.south.public.com/v1/",
                "versionList": "https://compute.south.public.com/",
                "enabled": false
            },
            {
                "id": 3,
                "region": "North",
                "global": true,
                "type": "object-store",
                "publicURL": "https://object-store.north.public.com/v1.0",
                "versionId": "1.0",
                "versionInfo": "https://object-store.north.public.com/v1.0/",
                "versionList": "https://object-store.north.public.com/",
                "enabled": true
            },
            {
                "id": 4,
                "region": "South",
                "type": "object-store",
                "publicURL": "https://object-store.south.public.com/v2",
                "versionId": "2",
                "versionInfo": "https://object-store.south.public.com/v2/",
                "versionList": "https://object-store.south.public.com/",
                "enabled": true
            },
            {
                "id": 5,
                "global": true,
                "type": "OS-DNS:DNS",
                "publicURL": "https://dns.public.com/v3.2",
                "versionId": "1.0",
                "versionInfo": "https://dns.public.com/v1.0/",
                "versionList": "https://dns.public.com/",
                "enabled": true
            }
        ],
    "OS-KSCATALOG:endpointsTemplates_links": []
}
```

Add Endpoint Template

Method	URI	Description
POST	`/v2.0/OS-KSCATALOG/endpointTemplates{?serviceId}`	Adds endpoint template.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404), badMediaType (415)

Request

This table shows the header parameters for the add endpoint template request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

Example 4.55. Add Endpoint Template: XML request

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

<endpointTemplate
  xmlns="http://docs.openstack.org/identity/api/ext/OS-KSCATALOG/v1.0"
  id="1"
  region="North"
  global="true"
  type="compute"
  name="Compute"
  publicURL="https://service-public.com/v1"
  internalURL="https://service-internal.com/v1"
  enabled="true">
  <version
    id="1"
    info="https://compute.north.public.com/v1/"
    list="https://compute.north.public.com/"
  />
</endpointTemplate>
```

Example 4.56. Add Endpoint Template: JSON request

```
{
    "OS-KSCATALOG:endpointTemplate":{
        "id": 1,
        "region": "North",
        "global": true,
        "type": "compute",
        "publicURL": "https://compute.north.public.com/v1",
        "internalURL": "https://compute.north.internal.com/v1",
        "versionId": "1",
        "versionInfo": "https://compute.north.public.com/v1/",
        "versionList": "https://compute.north.public.com/",
        "enabled": true
    }
}
```

Response

Example 4.57. Add Endpoint Template: XML response

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

<endpointTemplate
  xmlns="http://docs.openstack.org/identity/api/ext/OS-KSCATALOG/v1.0"
  id="1"
  region="North"
  global="true"
  type="compute"
  name="Compute"
  publicURL="https://service-public.com/v1"
  internalURL="https://service-internal.com/v1"
  enabled="true">
  <version
    id="1"
    info="https://compute.north.public.com/v1/"
    list="https://compute.north.public.com/"
  />
</endpointTemplate>
```

Example 4.58. Add Endpoint Template: JSON response

```
{
    "OS-KSCATALOG:endpointTemplate":{
        "id": 1,
        "region": "North",
        "global": true,
        "type": "compute",
        "publicURL": "https://compute.north.public.com/v1",
        "internalURL": "https://compute.north.internal.com/v1",
        "versionId": "1",
        "versionInfo": "https://compute.north.public.com/v1/",
        "versionList": "https://compute.north.public.com/",
        "enabled": true
    }
}
```

Get Endpoint Template

Method	URI	Description
GET	`/v2.0/OS-KSCATALOG/endpointTemplates/{endpointTemplateId}`	Gets endpoint templates.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get endpoint template request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the get endpoint template request:

Name	Type	Description
`{endpointTemplateId}`	String	The endpoint template ID.

This operation does not require a request body.

Response

Example 4.59. Get Endpoint Template: XML response

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

<endpointTemplate
  xmlns="http://docs.openstack.org/identity/api/ext/OS-KSCATALOG/v1.0"
  id="1"
  region="North"
  global="true"
  type="compute"
  name="Compute"
  publicURL="https://service-public.com/v1"
  internalURL="https://service-internal.com/v1"
  enabled="true">
  <version
    id="1"
    info="https://compute.north.public.com/v1/"
    list="https://compute.north.public.com/"
  />
</endpointTemplate>
```

Example 4.60. Get Endpoint Template: JSON response

```
{
    "OS-KSCATALOG:endpointTemplate":{
        "id": 1,
        "region": "North",
        "global": true,
        "type": "compute",
        "publicURL": "https://compute.north.public.com/v1",
        "internalURL": "https://compute.north.internal.com/v1",
        "versionId": "1",
        "versionInfo": "https://compute.north.public.com/v1/",
        "versionList": "https://compute.north.public.com/",
        "enabled": true
    }
}
```

Delete Endpoint Template.

Method	URI	Description
DELETE	`/v2.0/OS-KSCATALOG/endpointTemplates/{endpointTemplateId}`	Deletes an endpoint template.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the delete endpoint template. request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the delete endpoint template. request:

Name	Type	Description
`{endpointTemplateId}`	String	The endpoint template ID.

This operation does not require a request body.

Endpoint Operations

Method	URI	Description
GET	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints`	Lists endpoints for a tenant.
POST	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints`	Adds endpoint to a tenant.
GET	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints/{endpointId}`	Gets endpoint for a tenant.
DELETE	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints/{endpointId}`	Deletes an endpoint from a tenant.

List Endpoints

Method	URI	Description
GET	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints`	Lists endpoints for a tenant.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list endpoints request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the list endpoints request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.

This operation does not require a request body.

Response

Example 4.61. List Endpoints: XML response

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

<endpoints
    xmlns="http://docs.openstack.org/identity/api/v2.0">
  <endpoint
      id="1"
      tenantId="1"
      type="compute"
      name="Compute"
      region="North"
      publicURL="https://compute.north.public.com/v1"
      internalURL="https://compute.north.internal.com/v1"
      adminURL="https://compute.north.internal.com/v1">
      <version
          id="1"
          info="https://compute.north.public.com/v1/"
          list="https://compute.north.public.com/"
      />
  </endpoint>
  <endpoint
      id="2"
      tenantId="2"
      type="compute"
      name="Compute"
      region="South"
      publicURL="https://compute.north.public.com/v1"
      internalURL="https://compute.north.internal.com/v1"
      adminURL="https://compute.north.internal.com/v1">
      <version
          id="1"
          info="https://compute.north.public.com/v1/"
          list="https://compute.north.public.com/"
      />
  </endpoint>
  <endpoint
      id="3"
      tenantId="1"
      type="compute"
      name="Compute"
      region="East"
      publicURL="https://compute.north.public.com/v1"
      internalURL="https://compute.north.internal.com/v1"
      adminURL="https://compute.north.internal.com/v1"
  />
  <endpoint
      id="4"
      tenantId="1"
      type="compute"
      name="Compute"
      region="West"
      publicURL="https://compute.north.public.com/v1"
      internalURL="https://compute.north.internal.com/v1"
      adminURL="https://compute.north.internal.com/v1">
      <version
          id="1"
          info="https://compute.north.public.com/v1/"
          list="https://compute.north.public.com/"
      />
  </endpoint>
  <endpoint
      id="5"
      tenantId="1"
      type="compute"
      name="Compute"
      region="Global"
      publicURL="https://compute.north.public.com/v1"
      internalURL="https://compute.north.internal.com/v1"
      adminURL="https://compute.north.internal.com/v1">
      <version
          id="1"
          info="https://compute.north.public.com/v1/"
          list="https://compute.north.public.com/"
      />
  </endpoint>
</endpoints>
```

Example 4.62. List Endpoints: JSON response

```
{
    "endpoints":[{
                "id": 1,
                "tenantId": "1",
                "region": "North",
                "type": "compute",
                "publicURL": "https://compute.north.public.com/v1",
                "internalURL": "https://compute.north.internal.com/v1",
                "adminURL": "https://compute.north.internal.com/v1",
                "versionId": "1",
                "versionInfo": "https://compute.north.public.com/v1/",
                "versionList": "https://compute.north.public.com/"
            },
            {
                "id": 2,
                "tenantId": "1",
                "region": "South",
                "type": "compute",
                "publicURL": "https://compute.north.public.com/v1",
                "internalURL": "https://compute.north.internal.com/v1",
                "adminURL": "https://compute.north.internal.com/v1",
                "versionId": "1",
                "versionInfo": "https://compute.north.public.com/v1/",
                "versionList": "https://compute.north.public.com/"
            },
            {
                "id": 3,
                "tenantId": "1",
                "region": "East",
                "type": "compute",
                "publicURL": "https://compute.north.public.com/v1",
                "internalURL": "https://compute.north.internal.com/v1",
                "adminURL": "https://compute.north.internal.com/v1",
                "versionId": "1",
                "versionInfo": "https://compute.north.public.com/v1/",
                "versionList": "https://compute.north.public.com/"
            },
            {
                "id": 4,
                "tenantId": "1",
                "region": "West",
                "type": "compute",
                "publicURL": "https://compute.north.public.com/v1",
                "internalURL": "https://compute.north.internal.com/v1",
                "adminURL": "https://compute.north.internal.com/v1",
                "versionId": "1",
                "versionInfo": "https://compute.north.public.com/v1/",
                "versionList": "https://compute.north.public.com/"
            },
            {
                "id": 5,
                "tenantId": "1",
                "region": "Global",
                "type": "compute",
                "publicURL": "https://compute.north.public.com/v1",
                "internalURL": "https://compute.north.internal.com/v1",
                "adminURL": "https://compute.north.internal.com/v1",
                "versionId": "1",
                "versionInfo": "https://compute.north.public.com/v1/",
                "versionList": "https://compute.north.public.com/"
            }
        ],
    "endpoints_links":[]
}
```

Add Endpoint

Method	URI	Description
POST	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints`	Adds endpoint to a tenant.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404), badMediaType (415)

Request

This table shows the header parameters for the add endpoint request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the add endpoint request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.

This table shows the body parameters for the add endpoint request:

Name	Type	Description
`endpoint`	Endpoint Template WithOnly Id *(Optional)*	

Example 4.63. Add Endpoint: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<endpointTemplate
  xmlns="http://docs.openstack.org/identity/api/ext/OS-KSCATALOG/v1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="EndpointTemplateWithOnlyId"
  id="1"/>
```

Example 4.64. Add Endpoint: JSON request

```
{
    "OS-KSCATALOG:endpointTemplate":{
        "id": 1
    }
}
```

Response

Example 4.65. Add Endpoint: XML response

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

<endpoint
            id="1"
            tenantId="1"
            type="compute"
            name="Compute"
            region="North"
            publicURL="https://compute.north.public.com/v1"
            internalURL="https://compute.north.internal.com/v1"
            adminURL="https://compute.north.internal.com/v1"
            xmlns="http://docs.openstack.org/identity/api/v2.0">
            <version
                        id="1"
                        info="https://compute.north.public.com/v1/"
                        list="https://compute.north.public.com/"
            />
</endpoint>
```

Example 4.66. Add Endpoint: JSON response

```
{
  "endpoint": {
  "id": 1,
  "tenantId": 1,
  "region": "North",
  "type": "compute",
  "publicURL": "https://compute.north.public.com/v1",
  "internalURL": "https://compute.north.internal.com/v1",
  "adminURL": "https://compute.north.internal.com/v1",
  "versionId": "1",
  "versionInfo": "https://compute.north.public.com/v1/",
  "versionList": "https://compute.north.public.com/"
  }
}
```

Get Endpoint

Method	URI	Description
GET	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints/{endpointId}`	Gets endpoint for a tenant.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get endpoint request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the get endpoint request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.
`{endpointId}`	String	The endpoint ID.

This operation does not require a request body.

Response

Example 4.67. Get Endpoint: XML response

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

<endpoint
            id="1"
            tenantId="1"
            type="compute"
            name="Compute"
            region="North"
            publicURL="https://compute.north.public.com/v1"
            internalURL="https://compute.north.internal.com/v1"
            adminURL="https://compute.north.internal.com/v1"
            xmlns="http://docs.openstack.org/identity/api/v2.0">
            <version
                        id="1"
                        info="https://compute.north.public.com/v1/"
                        list="https://compute.north.public.com/"
            />
</endpoint>
```

Example 4.68. Get Endpoint: JSON response

```
{
  "endpoint": {
  "id": 1,
  "tenantId": 1,
  "region": "North",
  "type": "compute",
  "publicURL": "https://compute.north.public.com/v1",
  "internalURL": "https://compute.north.internal.com/v1",
  "adminURL": "https://compute.north.internal.com/v1",
  "versionId": "1",
  "versionInfo": "https://compute.north.public.com/v1/",
  "versionList": "https://compute.north.public.com/"
  }
}
```

Delete Endpoint.

Method	URI	Description
DELETE	`/v2.0/tenants/{tenantId}/OS-KSCATALOG/endpoints/{endpointId}`	Deletes an endpoint from a tenant.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the delete endpoint. request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the delete endpoint. request:

Name	Type	Description
`{tenantId}`	String	The tenant ID.
`{endpointId}`	String	The endpoint ID.

This operation does not require a request body.

OS-KSEC2 Admin Extension

Table 4.4. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

The OpenStack EC2 authentication extension adds the following calls:

Method	URI	Description
	User Operations	
POST	`/v2.0/users/{userId}/OS-KSADM/credentials`	Adds a credential to a user.
GET	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Lists credentials by type.
POST	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Updates credentials for a specified user.
DELETE	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Deletes user credentials.

Method	URI	Description
GET	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Gets user credentials.

User Operations

Method	URI	Description
POST	`/v2.0/users/{userId}/OS-KSADM/credentials`	Adds a credential to a user.
GET	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Lists credentials by type.
POST	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Updates credentials for a specified user.
DELETE	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Deletes user credentials.
GET	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Gets user credentials.

Add User Credentials

Method	URI	Description
POST	`/v2.0/users/{userId}/OS-KSADM/credentials`	Adds a credential to a user.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the add user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token.

This table shows the URI parameters for the add user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

Example 4.69. Add User Credentials: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
  <ec2Credentials
    xmlns="http://docs.openstack.org/identity/api/ext/OS-KSEC2/v1.0"
    username="testuser"
    key="aaaaa"
    signature="bbbbb"/>
```

Example 4.70. Add User Credentials: JSON request

```
{
    "OS-KSEC2-ec2Credentials":{
        "username": "test_user",
        "secret": "aaaaa",
        "signature": "bbb"
    }
}
```

Response

Example 4.71. Add User Credentials: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
  <ec2Credentials
    xmlns="http://docs.openstack.org/identity/api/ext/OS-KSEC2/v1.0"
    username="testuser"
    key="aaaaa"
    signature="bbbbb"/>
```

Example 4.72. Add User Credentials: JSON response

```
{
    "OS-KSEC2-ec2Credentials":{
        "username": "test_user",
        "secret": "aaaaa",
        "signature": "bbb"
    }
}
```

List Credentials by Type

Method	URI	Description
GET	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Lists credentials by type.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list credentials by type request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token.

This table shows the URI parameters for the list credentials by type request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Response

Example 4.73. List Credentials by Type: XML response

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

Example 4.74. List Credentials by Type: JSON response

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

This operation does not return a response body.

Update User Credentials

Method	URI	Description
POST	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Updates credentials for a specified user.

Normal response codes: 200

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the update user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token.

This table shows the URI parameters for the update user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

Example 4.75. Update User Credentials: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
  <ec2Credentials
    xmlns="http://docs.openstack.org/identity/api/ext/OS-KSEC2/v1.0"
    username="testuser"
    key="aaaaa"
    signature="bbbbb"/>
```

Example 4.76. Update User Credentials: JSON request

```
{
    "OS-KSEC2-ec2Credentials":{
        "username": "test_user",
        "secret": "aaaaa",
        "signature": "bbb"
    }
}
```

Response

Example 4.77. Update User Credentials: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
  <ec2Credentials
    xmlns="http://docs.openstack.org/identity/api/ext/OS-KSEC2/v1.0"
    username="testuser"
    key="aaaaa"
    signature="bbbbb"/>
```

Example 4.78. Update User Credentials: JSON response

```
{
    "OS-KSEC2-ec2Credentials":{
        "username": "test_user",
        "secret": "aaaaa",
        "signature": "bbb"
    }
}
```

Delete User Credentials

Method	URI	Description
DELETE	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Deletes user credentials.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the delete user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token.

This table shows the URI parameters for the delete user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Get User Credentials

Method	URI	Description
GET	`/v2.0/users/{userId}/OS-KSADM/credentials//OS-KSEC2:ec2Credentials`	Gets user credentials.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token.

This table shows the URI parameters for the get user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Response

Example 4.79. Get User Credentials: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
  <ec2Credentials
    xmlns="http://docs.openstack.org/identity/api/ext/OS-KSEC2/v1.0"
    username="testuser"
    key="aaaaa"
    signature="bbbbb"/>
```

Example 4.80. Get User Credentials: JSON response

```
{
    "OS-KSEC2-ec2Credentials":{
        "username": "test_user",
        "secret": "aaaaa",
        "signature": "bbb"
    }
}
```

OS-KSS3 Admin Extension

Table 4.5. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

Method	URI	Description
GET	`/v2.0/users/{userId}/OS-OS-KSS3/credentials{?marker,limit}`	Lists credentials.
POST	`/v2.0/users/{userId}/OS-OS-KSS3/credentials{?marker,limit}`	Adds a credential to a user.
GET	`/v2.0/users/{userId}/OS-OS-KSS3/credentials/OS-KSS3:s3Credentials`	Gets user credentials.
POST	`/v2.0/users/{userId}/OS-OS-KSS3/credentials/OS-KSS3:s3Credentials`	Updates credentials.
DELETE	`/v2.0/users/{userId}/OS-OS-KSS3/credentials/OS-KSS3:s3Credentials`	Deletes user credentials.

OS-KSVALIDATE Admin Extension

Table 4.6. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

Method	URI	Description
GET	`/v2.0/OS-KSVALIDATE/token/validate{?belongsTo,HP-IDM-serviceId}`	Checks that a token is valid and that it belongs to a specified tenant and service IDs. Returns the permissions for a particular client.
HEAD	`/v2.0/OS-KSVALIDATE/token/validate{?belongsTo,HP-IDM-serviceId}`	Checks that a token is valid and that it belongs to a specified tenant and service IDs, for performance.
GET	`/v2.0/OS-KSVALIDATE/token/endpoints{?HP-IDM-serviceId}`	Lists endpoints associated with a specific token.

5. HP Identity Extensions

Method	URI	Description
HP-IDM Admin Extension		
GET	`/v2.0/tokens/{tokenId}{?belongsTo, HP-IDM-serviceId}`	Validates a token and that it belongs to a specified tenant and services. Returns the permissions relevant to a particular client.
HEAD	`/v2.0/tokens/{tokenId}{?belongsTo, HP-IDM-serviceId}`	Validates a token and that it belongs to a specified tenant and services, for performance.

HP-IDM Admin Extension

Use the HP-IDM extension to perform the following operations on templates:

Method	URI	Description
GET	`/v2.0/tokens/{tokenId}{?belongsTo, HP-IDM-serviceId}`	Validates a token and that it belongs to a specified tenant and services. Returns the permissions relevant to a particular client.
HEAD	`/v2.0/tokens/{tokenId}{?belongsTo, HP-IDM-serviceId}`	Validates a token and that it belongs to a specified tenant and services, for performance.

Validate Token

Method	URI	Description
GET	`/v2.0/tokens/{tokenId}{?belongsTo,HP-IDM-serviceId}`	Validates a token and that it belongs to a specified tenant and services. Returns the permissions relevant to a particular client.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the validate token request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token.

This table shows the URI parameters for the validate token request:

Name	Type	Description
`{tokenId}`	String	The token ID.

This operation does not require a request body.

Response

Example 5.1. Validate Token: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<access xmlns="http://docs.openstack.org/identity/api/v2.0">
        <token id="ab48a9efdfedb23ty3494" expires="2010-11-01T03:32:15-05:00">
                <tenant id="456" name="My Project" />
        </token>
        <user id="123" username="jqsmith">
                <roles>
                        <role id="123" name="Admin" tenantId="one" />
                        <role id="234" name="object-store:admin" tenantId=
"1" />
                </roles>
        </user>
</access>
```

Example 5.2. Validate Token: JSON response

```
{
   "access":{
      "token":{
         "id":"ab48a9efdfedb23ty3494",
         "expires":"2010-11-01T03:32:15-05:00",
         "tenant":{
            "id":"345",
            "name":"My Project"
         }
      },
      "user":{
         "id":"123",
         "name":"jqsmith",
         "roles":[
            {
               "id":"234",
               "name":"compute:admin"
            },
            {
               "id":"234",
               "name":"object-store:admin",
               "tenantId":"1"
            }
         ],
         "roles_links":[

         ]
      }
   }
}
```

Check Token

Method	URI	Description
HEAD	`/v2.0/tokens/{tokenId}{?belongsTo, HP-IDM-serviceId}`	Validates a token and that it belongs to a specified tenant and services, for performance.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the check token request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token.

This table shows the URI parameters for the check token request:

Name	Type	Description
`{tokenId}`	String	The token ID.

This operation does not require a request body.

6. Rackspace Extensions to OpenStack Identity

RAX-GRPADM Admin Extension

Table 6.1. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

A new resource is created at `/RAX-GRPADM/groups` that enables the management of groups.

Method	URI	Description
Group Operations		
POST	`/v2.0/RAX-GRPADM/groups{?marker, limit,name}`	Adds a group.
GET	`/v2.0/RAX-GRPADM/groups/{groupId}`	Gets group information for a specified group ID.
PUT	`/v2.0/RAX-GRPADM/groups/{groupId}`	Updates a group.
GET	`/v2.0/RAX-GRPADM/groups/{groupId} {?marker,limit,marker,limit}`	Lists users for a group.
DELETE	`/v2.0/RAX-GRPADM/groups/{groupId}`	Deletes a group.
PUT	`/v2.0/RAX-GRPADM/groups/{groupId}/ users/{userId}`	Adds a user to a group.
DELETE	`/v2.0/RAX-GRPADM/groups/{groupId}/ users/{userId}`	Removes a user from a group.

Group Operations

Method	URI	Description
POST	`/v2.0/RAX-GRPADM/groups{?marker, limit,name}`	Adds a group.
GET	`/v2.0/RAX-GRPADM/groups/{groupId}`	Gets group information for a specified group ID.
PUT	`/v2.0/RAX-GRPADM/groups/{groupId}`	Updates a group.
GET	`/v2.0/RAX-GRPADM/groups/{groupId} {?marker,limit,marker,limit}`	Lists users for a group.
DELETE	`/v2.0/RAX-GRPADM/groups/{groupId}`	Deletes a group.
PUT	`/v2.0/RAX-GRPADM/groups/{groupId}/ users/{userId}`	Adds a user to a group.
DELETE	`/v2.0/RAX-GRPADM/groups/{groupId}/ users/{userId}`	Removes a user from a group.

Add a New Group

Method	URI	Description
POST	`/v2.0/RAX-GRPADM/groups{?marker, limit,name}`	Adds a group.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404), badMediaType (415)

Request

This table shows the header parameters for the add a new group request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the body parameters for the add a new group request:

Name	Type	Description
`group`	GroupFor Create *(Required)*	

Example 6.1. Add a New Group: XML request

```
<group name="group1" xmlns="http://docs.rackspace.com/identity/api/ext/RAX-
KSGRP/v1.0">
    <description>A Description of the group</description>
</group>
```

Example 6.2. Add a New Group: JSON request

```
{
  "RAX-KSGRP:group": {
      "name": "group1",
      "description": "A Description of the group"
  }
}
```

Response

This table shows the header parameters for the add a new group response:

Name	Type	Description
`Location`	AnyURI *(Required)*	The full URL to the new group is returned in the `Location` header.

Example 6.3. Add a New Group: XML response

```
<group id="1234" name="group1" xmlns="http://docs.rackspace.com/identity/api/
ext/RAX-KSGRP/v1.0">
    <description>A Description of the group</description>
</group>
```

Example 6.4. Add a New Group: JSON response

```
{
  "RAX-KSGRP:group": {
      "id": "1234",
      "name": "group1",
      "description": "A Description of the group"
  }
}
```

Get Group

Method	URI	Description
GET	`/v2.0/RAX-GRPADM/groups/{groupId}`	Gets group information for a specified group ID.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get group request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the get group request:

Name	Type	Description
`{groupId}`	String	The group ID.

This operation does not require a request body.

Response

Example 6.5. Get Group: XML response

```
<group id="1234" name="group1" xmlns="http://docs.rackspace.com/identity/api/
ext/RAX-KSGRP/v1.0">
    <description>A Description of the group</description>
</group>
```

Example 6.6. Get Group: JSON response

```
{
  "RAX-KSGRP:group": {
      "id": "1234",
      "name": "group1",
      "description": "A Description of the group"
  }
}
```

Update Group

Method	URI	Description
PUT	`/v2.0/RAX-GRPADM/groups/{groupId}`	Updates a group.

Normal response codes: 200

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404), badMediaType (415)

Request

This table shows the header parameters for the update group request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the update group request:

Name	Type	Description
`{groupId}`	String	The group ID.

This table shows the body parameters for the update group request:

Name	Type	Description
`group`	GroupFor Update *(Required)*	

Example 6.7. Update Group: XML request

```
<group name="newName" xmlns="http://docs.rackspace.com/identity/api/ext/RAX-
KSGRP/v1.0">
    <description>A new description</description>
</group>
```

Example 6.8. Update Group: JSON request

```
{
  "RAX-KSGRP:group": {
      "name": "newName",
      "description": "A Description of the group"
  }
}
```

Response

Example 6.9. Update Group: XML response

```
<group id="1234" name="newName" xmlns="http://docs.rackspace.com/identity/api/
ext/RAX-KSGRP/v1.0">
```

```
    <description>A new description</description>
</group>
```

Example 6.10. Update Group: JSON response

```
{
  "RAX-KSGRP:group": {
      "id": "1234",
      "name": "newName",
      "description": "A new description"
  }
}
```

Get Users for Group Get Users for Group

Method	URI	Description
GET	`/v2.0/RAX-GRPADM/groups/{groupId}{?marker,limit,marker,limit}`	Lists users for a group.

A list of users that belong to a specified group.

A list of users that belong to a specified group.

Normal response codes: 200, 203200, 203

Error response codes: identityFault (400, 500, …), identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503)

Request

This table shows the header parameters for the get users for group get users for group request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the get users for group get users for group request:

Name	Type	Description
`{groupId}`	String	The group ID.

This table shows the query parameters for the get users for group get users for group request:

Name	Type	Description
`marker`	String *(Optional)*	
`limit`	Int *(Optional)*	
`marker`	String *(Optional)*	
`limit`	Int *(Optional)*	

Response

This table shows the body parameters for the get users for group get users for group response:

Name	Type	Description
`next`	AnyURI *(Optional)*	
`previous`	AnyURI *(Optional)*	

Example 6.11. Get Users for Group Get Users for Group: XML response

```
<?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="u1000"/>
    <user xmlns="http://docs.openstack.org/identity/api/v2.0"
          enabled="true" email="john.smith@example.org"
          username="jqsmith" id="u1001"/>
</users>
```

Example 6.12. Get Users for Group Get Users for Group: JSON response

```
{
  "users": [
    {
      "id": "u1000",
      "username": "jqsmith",
      "email": "john.smith@example.org",
      "enabled": true
    },
    {
      "id": "u1001",
      "username": "jqsmith",
      "email": "john.smith@example.org",
      "enabled": true
    }
  ],
  "users_links": []
}
```

This table shows the body parameters for the get users for group get users for group response:

Name	Type	Description
`next`	AnyURI *(Optional)*	
`previous`	AnyURI *(Optional)*	

Example 6.13. Get Users for Group Get Users for Group: XML response

```
<?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="u1000"/>
    <user xmlns="http://docs.openstack.org/identity/api/v2.0"
```

```
          enabled="true" email="john.smith@example.org"
          username="jqsmith" id="u1001"/>
</users>
```

Example 6.14. Get Users for Group Get Users for Group: JSON response

```
{
  "users": [
    {
      "id": "u1000",
      "username": "jqsmith",
      "email": "john.smith@example.org",
      "enabled": true
    },
    {
      "id": "u1001",
      "username": "jqsmith",
      "email": "john.smith@example.org",
      "enabled": true
    }
  ],
  "users_links": []
}
```

Delete Group

Method	URI	Description
DELETE	`/v2.0/RAX-GRPADM/groups/{groupId}`	Deletes a group.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the delete group request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the delete group request:

Name	Type	Description
`{groupId}`	String	The group ID.

This operation does not require a request body.

Add User to Group

Method	URI	Description
PUT	`/v2.0/RAX-GRPADM/groups/{groupId}/users/{userId}`	Adds a user to a group.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the add user to group request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the add user to group request:

Name	Type	Description
`{groupId}`	String	The group ID.
`{userId}`	String	The user ID.

This operation does not require a request body.

Remove User from Group

Method	URI	Description
DELETE	`/v2.0/RAX-GRPADM/groups/{groupId}/users/{userId}`	Removes a user from a group.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the remove user from group request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the remove user from group request:

Name	Type	Description
`{groupId}`	String	The group ID.
`{userId}`	String	The user ID.

This operation does not require a request body.

RAX-KSGRP Admin Extension

Table 6.2. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

The Rackspace API Groups Service Extension adds the following calls:

Method	URI	Description
Group Operations		
GET	`/v2.0/users/{userId}/RAX-KSGRP`	List groups for a specified user.

Group Operations

Method	URI	Description
GET	`/v2.0/users/{userId}/RAX-KSGRP`	List groups for a specified user.

List Groups for a User

Method	URI	Description
GET	`/v2.0/users/{userId}/RAX-KSGRP`	List groups for a specified user.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list groups for a user request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the list groups for a user request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Response

Example 6.15. List Groups for a User: XML response

```
<groups xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0">
  <group xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0" id=
"1" name="Default" >
    <description>Default Limits</description>
  </group>
  <group xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0" id=
"1550" name="New Group 1" >
    <description>This is the first new group.</description>
  </group>
  <group xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSGRP/v1.0" id=
"214" name="Faster Defaults" >
    <description>Defaults with faster rate limits</description>
  </group>
</groups>
```

Example 6.16. List Groups for a User: JSON response

```
{
    "RAX-KSGRP:groups": [
        {
            "description": "Default Limits",
            "id": "1",
            "name": "Default"
        },
        {
            "description": "This is the first new group.",
            "id": "1550",
            "name": "New Group 1"
        },
        {
            "description": "Defaults with faster rate limits",
            "id": "214",
            "name": "Faster Defaults"
        }
    ],
    "RAX-KSGRP:groups_links":[]
}
```

RAX-KSKEY Admin Extension

Table 6.3. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

The Rackspace API Key Authentication Extension adds the following calls:

Method	URI	Description
User Operations		
POST	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials`	Adds a credential to a user.
GET	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/{?marker,limit}`	Lists credentials.
POST	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/RAX-KSKEY:apiKeyCredentials`	Updates credentials.
DELETE	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/RAX-KSKEY:apiKeyCredentials`	Deletes user credentials.
GET	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/RAX-KSKEY:apiKeyCredentials`	Gets user credentials.

User Operations

Method	URI	Description
POST	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials`	Adds a credential to a user.
GET	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/{?marker,limit}`	Lists credentials.
POST	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/RAX-KSKEY:apiKeyCredentials`	Updates credentials.
DELETE	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/RAX-KSKEY:apiKeyCredentials`	Deletes user credentials.
GET	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/RAX-KSKEY:apiKeyCredentials`	Gets user credentials.

Add user Credential

Method	URI	Description
POST	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials`	Adds a credential to a user.

Normal response codes: 201

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the add user credential request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the add user credential request:

Name	Type	Description
`{userId}`	String	The user ID.

Example 6.17. Add user Credential: XML request

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

Example 6.18. Add user Credential: JSON request

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

Response

Example 6.19. Add user Credential: XML response

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

Example 6.20. Add user Credential: JSON response

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

List Credentials

Method	URI	Description
GET	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/{?marker,limit}`	Lists credentials.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the list credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the list credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Response

Example 6.21. List Credentials: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<credentials xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://docs.openstack.org/identity/api/v2.0">
    <passwordCredentials username="test_user" password="test"/>
    <apiKeyCredentials
    xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSKEY/v1.0"
    username="testuser"
    apiKey="aaaaa-bbbbb-ccccc-12345678"/>
</credentials>
```

Example 6.22. List Credentials: JSON response

```
{
    "credentials":[{
            "passwordCredentials":{
                "username": "test_user",
                "password": "mypass"
            }
        },
        {
            "RAX-KSKEY:apiKeyCredentials":{
                "username": "test_user",
                "apiKey": "aaaaa-bbbbb-ccccc-12345678"
            }
        }
    ],
    "credentials_links":[]
}
```

Update User Credentials

Method	URI	Description
POST	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/RAX-KSKEY:apiKeyCredentials`	Updates credentials.

Normal response codes: 200

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the update user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the update user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

Example 6.23. Update User Credentials: XML request

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

Example 6.24. Update User Credentials: JSON request

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

Response

Example 6.25. Update User Credentials: XML response

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

Example 6.26. Update User Credentials: JSON response

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

Delete User Credentials

Method	URI	Description
DELETE	`/v2.0/users/{userId}/OS-RAX-KSKEY/credentials/RAX-KSKEY:apiKeyCredentials`	Deletes user credentials.

Normal response codes: 204

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415), itemNotFound (404)

Request

This table shows the header parameters for the delete user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the delete user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Get User Credentials

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

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get user credentials request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the get user credentials request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Response

Example 6.27. Get User Credentials: XML response

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

Example 6.28. Get User Credentials: JSON response

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

RAX-KSKEY apikeyCredentials Extended Attribute

The *`apikeyCredentials`* extended attribute supports Rackspace style authentication.

Verb	URI	Description
POST	/tokens	Authenticates and generates a token.

Normal Response Code(s):200, 203

Error Response Code(s): unauthorized (401), userDisabled (403), badRequest (400), identityFault (500), serviceUnavailable(503)

This call returns a token if successful. Clients obtain this token, along with the URL to other service APIs, by first authenticating against OpenStack Identity and supplying valid credentials. This extension provides support for Rackspace style API Key credentials.

Client authentication is provided through 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. See Chapter 2, “Client API Operations” [21].

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

Example 6.29. Auth with apikeyCredentials: JSON Request

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

Example 6.30. Auth with apikeyCredentials: XML Request

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

Example 6.31. Auth with apikeyCredentials: JSON Response

```
{
    "access":{
        "token":{
            "id": "ab48a9efdfedb23ty3494",
            "expires": "2010-11-01T03:32:15-05:00",
            "tenant":{
                "id": "t1000",
                "name": "My Project"
            }
        },
        "user":{
            "id": "u123",
            "name": "jqsmith",
            "roles":[{
                    "id": "100",
                    "name": "compute:admin"
                },
                {
                    "id": "101",
                    "name": "object-store:admin",
                    "tenantId": "t1000"
                }
            ],
            "roles_links":[]
        },
        "serviceCatalog":[{
                "name": "Cloud Servers",
                "type": "compute",
                "endpoints":[{
                        "tenantId": "t1000",
                        "publicURL": "https://compute.north.host.com/v1/
t1000",
                        "internalURL": "https://compute.north.internal/v1/
t1000",
                        "region": "North",
                        "versionId": "1",
                        "versionInfo": "https://compute.north.host.com/v1/",
                        "versionList": "https://compute.north.host.com/"
                    },
                    {
                        "tenantId": "t1000",
                        "publicURL": "https://compute.north.host.com/v1.1/
t1000",
                        "internalURL": "https://compute.north.internal/v1.1/
t1000",
                        "region": "North",
                        "versionId": "1.1",
                        "versionInfo": "https://compute.north.host.com/v1.1/",
                        "versionList": "https://compute.north.host.com/"
                    }
                ],
                "endpoints_links":[]
            },
            {
                "name": "Cloud Files",
                "type": "object-store",
                "endpoints":[{
                        "tenantId": "t1000",
                        "publicURL": "https://storage.north.host.com/v1/
t1000",
                        "internalURL": "https://storage.north.internal/v1/
t1000",
                        "region": "North",
                        "versionId": "1",
                        "versionInfo": "https://storage.north.host.com/v1/",
                        "versionList": "https://storage.north.host.com/"
                    },
                    {
                        "tenantId": "t1000",
                        "publicURL": "https://storage.south.host.com/v1/
t1000",
                        "internalURL": "https://storage.south.internal/v1/
t1000",
                        "region": "South",
                        "versionId": "1",
                        "versionInfo": "https://storage.south.host.com/v1/",
                        "versionList": "https://storage.south.host.com/"
                    }
                ]
            },
            {
                "name": "DNS-as-a-Service",
                "type": "dnsextension:dns",
                "endpoints":[{
                        "tenantId": "t1000",
                        "publicURL": "https://dns.host.com/v2.0/t1000",
                        "versionId": "2.0",
                        "versionInfo": "https://dns.host.com/v2.0/",
                        "versionList": "https://dns.host.com/"
                    }
                ]
            }
        ]
    }
}
```

Example 6.32. Auth with apikeyCredentials: XML Response

```
<?xml version="1.0" encoding="UTF-8"?>
<access xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://docs.openstack.org/identity/api/v2.0">
    <token id="ab48a9efdfedb23ty3494" expires="2010-11-01T03:32:15-05:00">
        <tenant id="t1000" name="My Project" />
    </token>
    <user id="u123" name="jqsmith">
        <roles>
            <role id="100" name="compute:admin"/>
            <role id="101" name="object-store:admin" tenantId="t1000"/>
        </roles>
    </user>
    <serviceCatalog>
        <service type="compute" name="Cloud Servers">
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://compute.north.host.com/v1/t1000"
                internalURL="https://compute.north.host.internal/v1/t1000">
                <version
                id="1"
                info="https://compute.north.host.com/v1/"
                list="https://compute.north.host.com/"
                />
            </endpoint>
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://compute.north.host.com/v1.1/t1000"
                internalURL="https://compute.north.host.internal/v1.1/t1000">
                <version
                id="1.1"
                info="https://compute.north.host.com/v1.1/"
                list="https://compute.north.host.com/" />
            </endpoint>
        </service>
        <service type="object-store" name="Cloud Files">
            <endpoint
        tenantId="t1000"
                region="North"
                publicURL="https://storage.north.host.com/v1/t1000"
                internalURL="https://storage.north.host.internal/v1/t1000">
                <version
                id="1"
                info="https://storage.north.host.com/v1/"
                list="https://storage.north.host.com/" />
            </endpoint>
            <endpoint
        tenantId="t1000"
                region="South"
                publicURL="https://storage.south.host.com/v1/t1000"
                internalURL="https://storage.south.host.internal/v1/t1000">
                <version
                id="1"
                info="https://storage.south.host.com/v1/"
                list="https://storage.south.host.com/" />
            </endpoint>
        </service>
        <service type="dnsextension:dns" name="DNS-as-a-Service">
            <endpoint
        tenantId="t1000"
                publicURL="https://dns.host.com/v2.0/t1000">
                <version
                id="2.0"
                info="https://dns.host.com/v2.0/"
                list="https://dns.host.com/" />
            </endpoint>
        </service>
    </serviceCatalog>
</access>
```

RAX-KSQA Admin Extension

Table 6.4. Authentication Header

Header Type	Name	Value
HTTP/1.1 Request	X-Auth-Token	txfa8426a08eaf

Following operations are the list of operations supported by Rackspace Secret Question and Answer Extension:

Method	URI	Description
	User Operations	
GET	`/v2.0/users/{userId}/RAX-KSQA/secretqa`	Gets a secret question and answer for a specified user.
PUT	`/v2.0/users/{userId}/RAX-KSQA/secretqa`	Updates a secret question and answer for a specified user.

User Operations

Method	URI	Description
GET	`/v2.0/users/{userId}/RAX-KSQA/secretqa`	Gets a secret question and answer for a specified user.
PUT	`/v2.0/users/{userId}/RAX-KSQA/secretqa`	Updates a secret question and answer for a specified user.

Get User Secret Question and Answer

Method	URI	Description
GET	`/v2.0/users/{userId}/RAX-KSQA/secretqa`	Gets a secret question and answer for a specified user.

Normal response codes: 200, 203

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), itemNotFound (404)

Request

This table shows the header parameters for the get user secret question and answer request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the get user secret question and answer request:

Name	Type	Description
`{userId}`	String	The user ID.

This operation does not require a request body.

Response

Example 6.33. Get User Secret Question and Answer: XML response

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

<secretQA xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSQA/v1.0"
          question="What is the color of my eyes?"
          answer="Leonardo Da Vinci" />
```

Example 6.34. Get User Secret Question and Answer: JSON response

```
{
    "RAX-KSQA:secretQA":{
        "question": "What is the color of my eyes?",
        "answer": "Leonardo Da Vinci"
    }
}
```

Update User Secret Question and Answer

Method	URI	Description
PUT	`/v2.0/users/{userId}/RAX-KSQA/secretqa`	Updates a secret question and answer for a specified user.

Normal response codes: 200

Error response codes: identityFault (400, 500, …), badRequest (400), unauthorized (401), forbidden (403), badMethod (405), overLimit (413), serviceUnavailable (503), badMediaType (415)

Request

This table shows the header parameters for the update user secret question and answer request:

Name	Type	Description
`X-Auth-Token`	String *(Required)*	A valid authentication token for an administrative user.

This table shows the URI parameters for the update user secret question and answer request:

Name	Type	Description
`{userId}`	String	The user ID.

Example 6.35. Update User Secret Question and Answer: XML request

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

<secretQA xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSQA/v1.0"
          question="What is the color of my eyes?"
          answer="Leonardo Da Vinci" />
```

Example 6.36. Update User Secret Question and Answer: JSON request

```
{
    "RAX-KSQA:secretQA":{
        "question": "What is the color of my eyes?",
        "answer": "Leonardo Da Vinci"
    }
}
```

Response

Example 6.37. Update User Secret Question and Answer: XML response

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

<secretQA xmlns="http://docs.rackspace.com/identity/api/ext/RAX-KSQA/v1.0"
          question="What is the color of my eyes?"
          answer="Leonardo Da Vinci" />
```

Example 6.38. Update User Secret Question and Answer: JSON response

```
{
    "RAX-KSQA:secretQA":{
        "question": "What is the color of my eyes?",
        "answer": "Leonardo Da Vinci"
    }
}
```