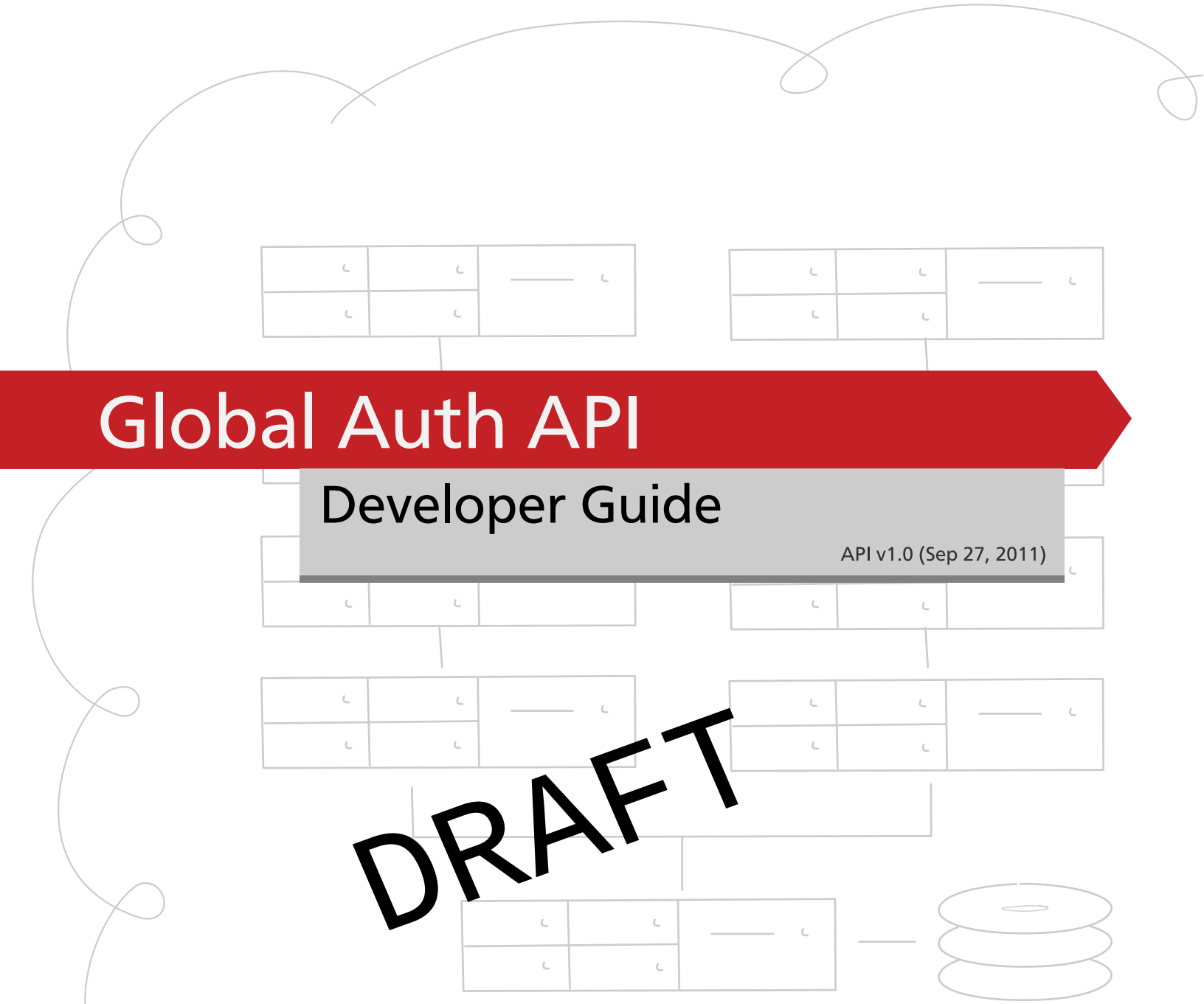
Global Auth API

Developer Guide

API v1.0 (Sep 27, 2011)



rackspace® HOSTING

docs.rackspace.com/api

Global Auth API Developer Guide

API v1.0 (2011-09-27)

This document is intended for software developers interested in developing applications which utilizes Global Auth API System as the authentication engine. It includes details on how to integrate with Global Auth API.

Table of Contents

List of Tables

List of Examples

1. Concepts

The Global Customer IDM system has several key concepts that are important to understand

Token	This is an opaque string that grants a client the right to access one or more Applications. Tokens may be revoked at any time and are valid for a finite duration..
User	This is a type of client associated with a person that has access to Rackspace Applications.
Application	An application is a user interface or service API that a client can interact with to consume specific business functionality. e.g Control Panel, Customer Service, Billing Service, etc.
Customer Identity Profile	This describes identity-related information about its associated customer. Examples include whether or not the customer is locked, its passsword rotation policy, etc.
Tenant	This is a container that contains a collection of a customers products and services that Rackspace provides.
Role	Clients interact with Applications to invoke specific functions. Applications use the Roles assigned to a client to determine what functions that client can perform in the context of those interactions. A Role in this context is basically analagous with a traditional LDAP role.

2. General API Information

This document is intended for software developers interested in developing applications which utilizes Global Auth API as the authentication engine. It includes details on how to integrate with Global Auth API.

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

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

2.1. Request/Response Types

The Global Auth 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, XML is the default. If conflicting formats are specified using both an `Accept` header and a query extension, the query extension takes precedence.

Table 2.1. Response Types

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

Example 2.1. JSON Request with Headers

```
POST /v1.0/token HTTP/1.1
Host: {host name TBD}
Content-Type: application/json
Accept: application/xml
```

```
{
  "grant_type": "PASSWORD",
  "client_id": "8972348923400fdshasdf",
  "client_secret": "0923899flewriudsb",
  "username": "testuser",
  "password": "P@ssword1"
}
```

Example 2.2. XML Response with Headers

```
HTTP/1.1 200 OKAY
Date: Mon, 12 Nov 2010 15:55:01 GMT
```

```
Server: Apache
Content-Length:
Content-Type: application/xml; charset=UTF-8
```


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

<auth xmlns="http://idm.api.rackspace.com/v1.0">
 <access_token
    expires_in="3600"
    id="ab48a9efdfedb23ty3494" />
 <refresh_token
    id="8792gdfskjbadf98y234r" />
 <user
     customerId="RCN-000-000-000"
     username="jqsmith" />
</auth>
```


2.2. Contracts

The Global Auth API uses a URI versioning scheme. The first element of the path contains the target version identifier (e.g. https://idm.api.rackspace.com/v1.0/…) All requests (except to query for contract version - see below) must contain a target version. Any features or functionality changes that would necessitate a break in API-compatibility will require a new version, which will result in the URI version being updated accordingly. When new API versions are released, older versions will be marked as `Deprecated`. Rackspace will work with developers and partners to ensure there is adequate time to migrate to the new version before deprecated versions are discontinued.

Example 2.3. Request with URI versioning

```
GET /v1.0/users HTTP/1.1
Host: idm.api.rackspace.com
Accept: application/xml
Authorization: Oauth ab48a9efdfedb23ty3494
```

Your application can programmatically determine available API contract versions by performing a **GET** on the root URL (i.e. with the version and everything to the right of it truncated) returned from the authentication system.

Example 2.4. Service Profile Request

```
GET HTTP/1.1
Host: idm.api.rackspace.com
Accept: application/xml
```

This operation does not require a request body.

Normal Response Code(s):200

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

Your application can programmatically determine available API contract versions by performing a **GET** on the root URL (https://idm.api.rackspace.com/).

Example 2.5. Service Profile Response

```
          <service-profile
  xmlns="http://service-registry.api.rackspace.com/service-profile"
  xmlns:atom="http://www.w3.org/2005/Atom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://service-registry.api.rackspace.com/service-
profile ./../resources/xsd/service-profile-v2.0.xsd"

  id="000"
  name="Customer Idm"
  canonical-name="idm"
  dns-zone="idm.api.rackspace.com"
  service-model="Utility">

    <atom:link rel="self" href="https://idm.api.rackspace.com" />

    <short-description>Allows users access Rackspace resources and systems.</
short-description>

    <detailed-description>
      The customer idm api allows Rackspace clients to obtain tokens that can
 be used to access resources in Rackspace.
      It also allows clients manage identities and delegate access to
 resources.
    </detailed-description>

    <contract version="v1.0" status="BETA">
      <media-types>
        <media-type base="application/xml" type="application/vnd.rackspace.
idm-v1.0+xml">
          <atom:link rel="describedby" type="application/xml" href=
"${baseUrl}v1.0/docs/xsd/idmapi.xsd" />
        </media-type>
      </media-types>

      <atom:link rel="self" href="${baseUrl}v1.0"/>
      <atom:link rel="describedby" type="application/vnd.sun.wadl+xml" href=
"${baseUrl}v1.0/application.wadl" title="Wadl" />
      <atom:link rel="documentation" href="${baseUrl}docs/v1.0/developerguide.
pdf" title="Developer Guide"/>
   </contract>
</service-profile>
```

You can also obtain additional information about a contract version by performing a **GET** on the base version URL (e.g. https://idm.api.rackspace.com/v1.0).

Example 2.6. Service Contract Request

```
          GET HTTP/1.1
          Host: idm.api.rackspace.com/v1.0
          Accept: application/xml
```

This operation does not require a request body.

Normal Response Code(s):200

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

Example 2.7. Service Contract Response

```
          <contract xmlns="http://service-registry.api.rackspace.com/service-
profile"
  xmlns:atom="http://www.w3.org/2005/Atom" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
  xsi:schemaLocation="https://service-registry.api.rackspace.com/service-
profile ./../resources/xsd/service-profile-v2.0.xsd"

  version="v1.0" status="BETA">

   <media-types>
     <media-type base="application/xml" type="application/vnd.rackspace.idm-
v1.0+xml">
       <atom:link rel="describedby" type="application/xml"
          href="https://idm.api.rackspace.com/v1.0/docs/xsd/idmapi.xsd" />
     </media-type>
  </media-types>

  <atom:link rel="self" href="https://idm.api.rackspace.com/v1.0" />
  <atom:link rel="describedby" type="application/vnd.sun.wadl+xml" href=
"https://idm.api.rackspace.com/v1.0/application.wadl" />
  <atom:link rel="documentation" href="https://idm.api.rackspace.com/docs/v1.
0/developerguide.pdf" />
</contract>
```

2.3. Faults

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

Example 2.8. XML Fault Response

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

<serviceFault xmlns="http://common.api.rackspace.com/v1.0"
    code="500">
    <message>Fault</message>
    <details>Error Details...</details>
</serviceFault>
```

Example 2.9. JSON Fault Response

```
{
  "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 (e.g a stack trace).

The root element of the fault (serviceFault) may change depending on how important it is to explicitly identify a specific fault type for a service. E.g it might be important to a client to distinguish between usernameConflict and applicationConflict even though they both have an error code of 409.The following is an example of a usernameConflict error.

Example 2.10. XML Username Conflict Fault

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

<usernameConflict xmlns="http://idm.api.rackspace.com/v1.0"
    code="409">
    <message>Conflict.</message>
    <details>Error Details...</details>
</usernameConflict>
```

Example 2.11. JSON Username Conflict Fault

```
{
  "message": "Conflict",
  "details": "Error Details...",
  "code": 409
}
```

The following is a list of possible fault types along with their associated error codes.

Table 2.2. Fault Types

Fault Element	Associated Error Code	Expected in Most Requests
serviceFault	500	✓
badRequest	400	✓
serviceUnavailable	503	✓
unauthorized	401	✓
forbidden	403	
itemNotFound	404	
passwordValidation	400	
userDisabled	403	
applicationNameConflict	409	
usernameConflict	409	

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

2.4. Getting Started

The first step to using the Global Auth API is registering your application with Global Auth. The Global Auth Team will need the following information in order to register your application.

Table 2.3. Application Information needed to register with Global Auth API

Item	Description	Example
Name	The name of the application	Cloud Servers API
Simple Name	A continuous lower case name	cloud_servers_api
Description	A human readable description of what the application does, which should be presentable to end users.	Cloud Servers API is ...

Once the Global Auth Team has this information we will add your application to the Global Auth System and send you the clientId and clientSecret that you will need in order to authenticate with the Global Auth API (see api details below for Authentication).

2.5. Authorization Header

Most calls made against the Global Auth API require the addition of an authorization header in the request.

The format of the authoriztion header is the access token id that belongs to the entity making the call. Here is an example request below

Example 2.12. Authorization Header Request Format

```
       POST /v1.0/users HTTP/1.1
Host: {host name TBD}
Accept: application/xml
X-Auth-Token: ab48a9efdfedb23ty3494
```

3. Admin API (Service Developer Operations)

Verb	URI	Description
	Token Operations	
POST	/tokens	Authenticate a client and generate an access token.
GET	/tokens/{tokenId}	Check that a token is valid and return the token details.
DELETE	/tokens/{tokenId}	Revoke Token.
GET	/tokens/{tokenId}/applications/{applicationId}	Check if token has access to an application.
GET	/tokens/{tokenId}/applications/{applicationId}/roles/{roleId}	Check if token has role on an application.
	User Operations	
GET	/users?username=*string*&offset=*int*&limit=*int*	Gets a list of users.
POST	/users	Adds a new user.
GET	/users/{userId}	Get a user.
PUT	/users/{userId}	Update a user.
DELETE	/users/{userId}	Delete a user.
GET	/users/{userId}/secret	Get a user's secret question and answer.
PUT	/users/{userId}/secret	Sets a user's secret question and answer.
GET	/users/{userId}/passwordcredentials	Get a user's password credentials.
PUT	/users/{userId}/passwordcredentials	Sets a user's password credentials.
POST	/users/{userId}/passwordcredentials	Reset a user's password credentials.
GET	/users/{userId}/passwordcredentials/recoverytoken	Get a token that can be used to reset a user's password credentials.
GET	/users/{userId}/delegatedrefreshtokens	Get a list of a user's delegated refresh tokens.
GET	/users/{userId}/delegatedrefreshtokens/{tokenId}	Get a user's delegated refresh token.
DELETE	/users/{userId}/delegatedrefreshtokens/{tokenId}	Deletes a user's delegated refresh token.
GET	/users/{userId}/applications	Get the applications that have been provisioned for a user.
PUT	/users/{userId}/applications/{applicationId}	Provision an application for a user.
DELETE	/users/{userId}/applications/{applicationId}	Removed a provisioned application from a user.
GET	/users/{userId}/tenants/roles?applicationId=*string*&tenantId=*string*	Get a list of user's roles on all tenants.
PUT	/users/{userId}/tenants/{tenantId}/roles/{roleId}	Grant user a role on the tenant
DELETE	/users/{userId}/tenants/{tenantId}/roles/{roleId}	Revoke a user's role on a tenant.
GET	/users/{userId}/roles?applicationId=*string*	Get a list of user's global roles.

Verb	URI	Description
PUT	/users/{userId}/roles/{roleId}	Grant user a global role.
DELETE	/users/{userId}/roles/{roleId}	Remove global role from user.
Application Operations		
GET	/applications?name=*string*&offset=*int*&limit=*int*	Gets a list of applications.
POST	/applications	Add an application.
GET	/applications/{applicationId}	Get an application.
PUT	/applications/{applicationId}	Update an application.
DELETE	/applications/{applicationId}	Delete as application.
POST	/applications/{applicationId}/secretcredentials	Reset application secret.
GET	/applications/{applicationId}/applications	Get all applications that have been provisioned for this application..
PUT	/applications/{applicationId}/applications/{provisionedApplicationId}	Provision an application for this application.
DELETE	/applications/{applicationId}/applications/{provisionedApplicationId}	Removes a provisioned application from this application.
GET	/applications/{applicationId}/tenants/roles?applicationId=*string*&tenantId=*string*	Get a list of applications's roles on all tenants.
PUT	/applications/{applicationId}/tenants/{tenantId}/roles/{roleId}	Grant application a role on the tenant
DELETE	/applications/{applicationId}/tenants/{tenantId}/roles/{roleId}	Revoke an application's role on a tenant.
GET	/applications/{applicationId}/roles?applicationId=*string*	Get a list of aplications's global roles.
PUT	/applications/{applicationId}/roles/{roleId}	Grant application a global role.
DELETE	/applications/{applicationId}/roles/{roleId}	Remove global role from application.
Customer Identity Profile Operations		
POST	/customeridentityprofiles/{customerId}	Adds a customer's identity profile.
GET	/customeridentityprofiles/{customerId}	Gets a customer's identity profile.
PUT	/customeridentityprofiles/{customerId}	Updates a customer's identity profile.
DELETE	/customeridentityprofiles/{customerId}	Delete a customer's identity profile.
GET	/customeridentityprofiles/{customerId}/passwordrotationpolicy	Gets the password rotation policy for a customer.
PUT	/customeridentityprofiles/{customerId}/passwordrotationpolicy	Sets the password rotation policy for a customer.
GET	/customeridentityprofiles/{customerId}/users?offset=*int*&limit=*int*	Gets a list of users that this customer owns.

Verb	URI	Description
GET	`/customeridentityprofiles/{customerId}/applications?offset=int&limit=int`	Gets a list of applications that this customer owns.
Password Operations		
GET	`/passwordrules`	Get password rules.
POST	`/passwordrules/validation`	Validate password to ensure it conforms to password rules.
Role Operations		
GET	`/roles?name=string&applicationId=string`	Get all roles.
POST	`/roles`	Add a new role.
GET	`/roles/{roleId}`	Get a Role
POST	`/roles/{roleId}`	Modify an existing role.
POST	`/roles/{roleId}`	Delete an existing role.

3.1. Token Operations

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

Verb	URI	Description
POST	`/tokens`	Authenticate a client and generate an access token.
GET	`/tokens/{tokenId}`	Check that a token is valid and return the token details.
DELETE	`/tokens/{tokenId}`	Revoke Token.
GET	`/tokens/{tokenId}/applications/{applicationId}`	Check if token has access to an application.
GET	`/tokens/{tokenId}/applications/{applicationId}/roles/{roleId}`	Check if token has role on an application.

3.1.1. Authenticate

Verb	URI	Description
POST	/tokens	Authenticate a client and generate an access token.

Normal Response Code(s): 200, 203

Error Response Code(s): userDisabled (403), serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

There are various ways to authenticate in the global auth system. You can authenticate as a Racker, an application on behalf of a customer, or an application directly. Examples below show the different authentication mechanisms.

Example 3.1. Auth as Application Request: XML

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

<authCredentials
    xmlns="http://idm.api.rackspace.com/v1.0"
    client_secret="0923899flewriudsb"
    client_id="8972348923400fdshasdf"
    grant_type="CLIENT_CREDENTIALS" />
```

Example 3.2. Auth as Application Request: JSON

```
{
  "grant_type": "CLIENT_CREDENTIALS",
  "client_id": "8972348923400fdshasdf",
  "client_secret": "0923899flewriudsb"
}
```

Example 3.3. Auth as Application Response: XML

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

<auth xmlns="http://idm.api.rackspace.com/v1.0">
 <access_token
    expires_in="3600"
    id="ab48a9efdfedb23ty3494" />
 <refresh_token
    id="8792gdfskjbadf98y234r" />
 <application
     customerId="RCN-000-000-000"
     clientId="781hhh47kj34kj3434" />
</auth>
```

Example 3.4. Auth as Application Response: JSON

```
{
  "accessToken": {
    "id": "ab48a9efdfedb23ty3494",
    "expiresIn": 3600
  },
  "refreshToken": {
    "id": "8792gdfskjbadf98y234r"
```

```
  },
  "user" : {
    "clientId": "781hhh47kj34kj3434",
    "customerId": "RCN-000-000-000",
  }
}
```

Example 3.5. Auth as Customer Request: XML

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

<authCredentials
    xmlns="http://idm.api.rackspace.com/v1.0"
    password="P@ssword1"
    username="testuser"
    client_secret="0923899flewriudsb"
    client_id="8972348923400fdshasdf"
    grant_type="PASSWORD" />
```

Example 3.6. Auth as Customer Request: JSON

```
{
  "grant_type": "PASSWORD",
  "client_id": "8972348923400fdshasdf",
  "client_secret": "0923899flewriudsb",
  "username": "testuser",
  "password": "P@ssword1"
}
```

Example 3.7. Auth as Customer Response: XML

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

<auth xmlns="http://idm.api.rackspace.com/v1.0">
 <access_token
    expires_in="3600"
    id="ab48a9efdfedb23ty3494" />
 <refresh_token
    id="8792gdfskjbadf98y234r" />
 <user
     customerId="RCN-000-000-000"
     username="jqsmith" />
</auth>
```

Example 3.8. Auth as Customer Response: JSON

```
{
  "accessToken": {
    "id": "ab48a9efdfedb23ty3494",
    "expiresIn": 3600
  },
  "refreshToken": {
    "id": "8792gdfskjbadf98y234r"
  },
  "user" : {
    "username": "jqsmith",
    "customerId": "RCN-000-000-000",
  }
```

```
}
```

Example 3.9. Auth as Racker Request: XML

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

<rackerCredentials
    xmlns="http://idm.api.rackspace.com/v1.0"
    password="password"
    username="racker.sso"
    client_secret="0923899flewriudsb"
    client_id="8972348923400fdshasdf"
    grant_type="PASSWORD" />
```

Example 3.10. Auth as Racker Request: JSON

```
{
  "grant_type": "PASSWORD",
  "client_id": "8972348923400fdshasdf",
  "client_secret": "0923899flewriudsb",
  "username": "racker.sso",
  "password": "password"
}
```

Example 3.11. Auth as Racker Response: XML

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

<auth xmlns="http://idm.api.rackspace.com/v1.0">
 <access_token
    expires_in="3600"
    id="ab48a9efdfedb23ty3494" />
 <refresh_token
    id="8792gdfskjbadf98y234r" />
 <racker username="john.so/">

</auth>
```

Example 3.12. Auth as Racker Response: JSON

```
{
  "accessToken": {
    "id": "ab48a9efdfedb23ty3494",
    "expiresIn": 3600
  },
  "refreshToken": {
    "id": "8792gdfskjbadf98y234r"
  },
  "racker" : {
    "username": "john.so"
  }
}
```

3.1.2. Validate Token

Verb	URI	Description
GET	/tokens/{tokenId}	Check that a token is valid and return the token details.

Normal Response Code(s): 200, 203

Error Response Code(s): userDisabled (403), serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.13. Validate Token Response: XML

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

<auth xmlns="http://idm.api.rackspace.com/v1.0">
 <access_token
    expires_in="3600"
    id="ab48a9efdfedb23ty3494" />
 <user
     customerId="RCN-000-000-000"
     username="jqsmith">
  <roles xmlns="http://idm.api.rackspace.com/v1.0">
       <role name="network_admin" applicationId="compute"/>
       <role name="technical_reviewer" tenantId="57884" applicationId="swift"/
>
       <role name="admin" tenantId="cf_89983" applicationId="billing"/>
  </roles>
 </user>
</auth>
```

Example 3.14. Validate Token Response: JSON

```
{
    "access_token": {
        "id": "ab48a9efdfedb23ty3494",
        "expires_in": "3600"
    },
    "user" : {
          "username": "jqsmith",
          "customerId": "RCN-000-000-000",
     },
    "roles": [{ "name": "network_admin",
                "applicationId": "compute"},
              { "name": "technical_reviewer",
                "tenantId": "57884",
                "applicationId": "swift"},
              { "name": "admin",
                "tenantId": "cf_89983",
                "applicationId": "billing"}]

}
```

Table 3.1. Validate Token Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.

Name	Style	Type	Description
tokenId	template	String	

This operation does not require a request body.

3.1.3. Revoke Token

Verb	URI	Description
DELETE	`/tokens/{tokenId}`	Revoke Token.

Normal Response Code(s): 204

Error Response Code(s): userDisabled (403), serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.2. Revoke Token Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`tokenId`	template	String	

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

3.1.4. Check if Token has access to an Application

Verb	URI	Description
GET	`/tokens/{tokenId}/applications/{applicationId}`	Check if token has access to an application.

Normal Response Code(s): 204, 404

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.3. Check if Token has access to an Application Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`tokenId`	template	String	
`applicationId`	template	String	

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

3.1.5. Check if Token has Role on an Application

Verb	URI	Description
GET	`/tokens/{tokenId}/applications/{applicationId}/roles/{roleId}`	Check if token has role on an application.

Normal Response Code(s): 204, 404

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.4. Check if Token has Role on an Application Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`tokenId`	template	String	
`applicationId`	template	String	
`roleId`	template	String	

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

3.2. User Operations

The operations described in this section allows service developers manage users.

Verb	URI	Description
GET	`/users?username=`*`string`*`&offset=`*`int`*`&limit=`*`int`*	Gets a list of users.
POST	`/users`	Adds a new user.
GET	`/users/{userId}`	Get a user.
PUT	`/users/{userId}`	Update a user.
DELETE	`/users/{userId}`	Delete a user.
GET	`/users/{userId}/secret`	Get a user's secret question and answer.
PUT	`/users/{userId}/secret`	Sets a user's secret question and answer.
GET	`/users/{userId}/passwordcredentials`	Get a user's password credentials.
PUT	`/users/{userId}/passwordcredentials`	Sets a user's password credentials.
POST	`/users/{userId}/passwordcredentials`	Reset a user's password credentials.
GET	`/users/{userId}/passwordcredentials/recoverytoken`	Get a token that can be used to reset a user's password credentials.
GET	`/users/{userId}/delegatedrefreshtokens`	Get a list of a user's delegated refresh tokens.
GET	`/users/{userId}/delegatedrefreshtokens/{tokenId}`	Get a user's delegated refresh token.
DELETE	`/users/{userId}/delegatedrefreshtokens/{tokenId}`	Deletes a user's delegated refresh token.
GET	`/users/{userId}/applications`	Get the applications that have been provisioned for a user.

Verb	URI	Description
PUT	`/users/{userId}/applications/{applicationId}`	Provision an application for a user.
DELETE	`/users/{userId}/applications/{applicationId}`	Removed a provisioned application from a user.
GET	`/users/{userId}/tenants/roles?applicationId=`*`string`*`&tenantId=`*`string`*	Get a list of user's roles on all tenants.
PUT	`/users/{userId}/tenants/{tenantId}/roles/{roleId}`	Grant user a role on the tenant
DELETE	`/users/{userId}/tenants/{tenantId}/roles/{roleId}`	Revoke a user's role on a tenant.
GET	`/users/{userId}/roles?applicationId=`*`string`*	Get a list of user's global roles.
PUT	`/users/{userId}/roles/{roleId}`	Grant user a global role.
DELETE	`/users/{userId}/roles/{roleId}`	Remove global role from user.

3.2.1. Get Users

Verb	URI	Description
GET	`/users?username=string&offset=int&limit=int`	Gets a list of users.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.15. Get Users Response: XML

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

<users xmlns="http://idm.api.rackspace.com/v1.0"
    limit="10"
    offset="0"
    totalRecords="2">
    <user
        softDeleted="false" locked="false"
        status="ACTIVE" region="America/Chicago"
        iname="@Example.Smith*John"
        inum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!1111"
        prefLanguage="US_en" displayName="John Smith"
        lastName="Smith" middleName="Quincy"
        firstName="John" personId="RPN-111-111-111"
        email="john.smith@example.org"
        customerInum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE"
        customerId="RCN-000-000-000"
        username="jqsmith" />
    <user softDeleted="false" locked="false"
        status="ACTIVE" region="America/Chicago"
        iname="@Example.Anderson*Bob"
        inum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!2222"
        prefLanguage="US_en" displayName="Bob Anderson"
        lastName="Anderson" middleName="Mark"
        firstName="Bob" personId="RPN-111-111-222"
        email="bob.anderson@example.org"
        customerInum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE"
        customerId="RCN-000-000-000"
        username="bmanderson" />
</users>
```

Example 3.16. Get Users Response: JSON

```
{"user": [
    {
      "username": "jqsmith",
      "customerId": "RCN-000-000-000",
      "customerInum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE",
      "email": "john.smith@example.org",
      "personId": "RPN-111-111-111",
      "firstName": "John",
      "middleName": "Quincy",
      "lastName": "Smith",
      "displayName": "John Smith",
      "prefLanguage": "US_en",
```

```
      "inum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!1111",
      "iname": "@Example.Smith*John",
      "region": "America/Chicago",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    },
    {
      "username": "bmanderson",
      "customerId": "RCN-000-000-000",
      "customerInum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE",
      "email": "bob.anderson@example.org",
      "personId": "RPN-111-111-222",
      "firstName": "Bob",
      "middleName": "Mark",
      "lastName": "Anderson",
      "displayName": "Bob Anderson",
      "prefLanguage": "US_en",
      "inum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!2222",
      "iname": "@Example.Anderson*Bob",
      "timeZone": "America/Chicago",
      "region": "SAT",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    }
  ],
  "totalRecords": 2,
  "offset": 0,
  "limit": 10
}
```

Table 3.5. Get Users Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
username	query	String	Allows you search for a user by username Optional.
offset	query	Int	0 based index to start list Optional.
limit	query	Int	Number of records to be returned Optional.

3.2.2. Add a User

Verb	URI	Description
POST	`/users`	Adds a new user.

Normal Response Code(s): 200

Error Response Code(s): passwordValidationFault (400), usernameConflict (409), serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503)

The Username attribute is required in this request.If a blank password is passed in the Password element, the API will generate a random password for the user. The prefLanguage attribute defaults to "US_en" and the timeZone attribute defaults to "America/Chicago".

Example 3.17. Add User Request: XML

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

<user xmlns="http://idm.api.rackspace.com/v1.0"
    region="SAT" prefLanguage="US_en" timeZone="America/Chicago"
    displayName="John Smith" lastName="Smith"
    middleName="Quincy" firstName="John"
    personId="RPN-111-111-111"
    email="john.smith@example.org"
    customerId="RCN-000-000-000" username="jqsmith">
    <secret secretAnswer="Francis"
        secretQuestion="What is your dogs name?" />
    <passwordCredentials xmlns="http://idm.api.rackspace.com/v1.0" >
       <currentPassword password="C@n+f001me!" />
    </passwordCredentials>
</user>
```

Example 3.18. Add User Request: JSON

```
{
  "secret": {
    "secretAnswer": "Francis",
    "secretQuestion": "What is your dogs name?"
  },
  "passwordCredentials": {
   "currentPassword": {
          "password": "P@ssword1"
    }
  },
  "username": "jqsmith",
  "customerId": "RCN-000-000-000",
  "email": "john.smith@example.org",
  "personId": "RPN-111-111-111",
  "firstName": "John",
  "middleName": "Quincy",
  "lastName": "Smith",
  "displayName": "John Smith",
  "prefLanguage": "US_en",
  "region": "America/Chicago"
}
```

Example 3.19. Add User Response: XML

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

<user xmlns="http://idm.api.rackspace.com/v1.0"
    region="SAT" prefLanguage="US_en" timeZone="America/Chicago"
    displayName="John Smith" lastName="Smith"
    middleName="Quincy" firstName="John"
    personId="RPN-111-111-111"
    email="john.smith@example.org"
    customerId="RCN-000-000-000" username="jqsmith"
    timeZone="America/Chicago"
    prefLanguage="US_en"
    region="SAT">
    <secret secretAnswer="Francis"
        secretQuestion="What is your dogs name?" />
    <passwordCredentials xmlns="http://idm.api.rackspace.com/v1.0" >
       <currentPassword password="C@n+f001me!" />
    </passwordCredentials>
</user>
```

Example 3.20. Add User Response: JSON

```
{
  "secret": {
    "secretQuestion": "What is your dogs name?",
    "secretAnswer": "Francis"
  },
  "passwordCredentials": {
   "currentPassword": {
         "password": "C@n+f001me!"
         }
  },
  "username": "jqsmith",
  "customerId": "RCN-000-000-000",
  "email": "john.smith@example.org",
  "personId": "RPN-111-111-111",
  "firstName": "John",
  "middleName": "Quincy",
  "lastName": "Smith",
  "displayName": "John Smith",
  "prefLanguage": "US_en",
  "timeZone": "America/Chicago",
  "region": "SAT",
  "enabled": "true",
  "softDeleted": false
}
```

Table 3.6. Add a User Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.

Table 3.7. Add a User Response Parameters

Name	Style	Type	Description
Location	header	String	Location to the uri of the newly created resource

Name	Style	Type	Description
			Required.

3.2.3. Get a User

Verb	URI	Description
GET	`/users/{userId}`	Get a user.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.21. Get User Response: XML


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

<user
    xmlns="http://idm.api.rackspace.com/v1.0"
    softDeleted="false" locked="false"
    status="ACTIVE" region="America/Chicago"
    inum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!1111"
    prefLanguage="US_en" displayName="John Smith"
    lastName="Smith" middleName="Quincy"
    firstName="John" personId="RPN-111-111-111"
    email="john.smith@example.org"
    customerInum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE"
    customerId="RCN-000-000-000"
    username="jqsmith">
</user>
```


Example 3.22. Get User Response: JSON


```
{
  "username": "jqsmith",
  "customerId": "RCN-000-000-000",
  "customerInum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE",
  "email": "john.smith@example.org",
  "personId": "RPN-111-111-111",
  "firstName": "John",
  "middleName": "Quincy",
  "lastName": "Smith",
  "displayName": "John Smith",
  "prefLanguage": "US_en",
  "inum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!1111",
  "region": "America/Chicago",
  "status": "ACTIVE",
  "locked": false,
  "softDeleted": false
}
```


Table 3.8. Get a User Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	

This operation does not require a request body.

3.2.4. Update a User

Verb	URI	Description
PUT	/users/{userId}	Update a user.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.23. Update User Request: XML


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

<user xmlns="http://idm.api.rackspace.com/v1.0"
    email="john.smith@somenewemail.org"
    username="jqsmith" />
```


Example 3.24. Update User Request: JSON


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


Example 3.25. Update User Response: XML


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

<user xmlns="http://idm.api.rackspace.com/v1.0"
    timeZone="America/Chicago" prefLanguage="US_en"
    displayName="John Smith" lastName="Smith"
    middleName="Quincy" firstName="John"
    region="SAT" personId="RPN-111-111-111"
    email="john.smith@somenewemail.org"
    customerId="RCN-000-000-000" username="jqsmith" />
```


Example 3.26. Update User Response: JSON


```
{
  "username": "jqsmith",
  "customerId": "RCN-000-000-000",
  "customerInum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE",
  "email": "john.smith@somenewemail.org",
  "personId": "RPN-111-111-111",
  "firstName": "John",
  "middleName": "Quincy",
  "lastName": "Smith",
  "displayName": "John Smith",
  "prefLanguage": "US_en",
  "inum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!1111",
  "iname": "@Example.Smith*John",
  "region": "America/Chicago",
  "status": "ACTIVE",
  "locked": false,
```

```
  "softDeleted": false
}
```

Table 3.9. Update a User Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
userId	template	String	

3.2.5. Delete a User

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

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.10. Delete a User Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	

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

3.2.6. Get User Secret

Verb	URI	Description
GET	`/users/{userId}/secret`	Get a user's secret question and answer.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.27. Get User Secret Response: XML

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

<secret xmlns="http://idm.api.rackspace.com/v1.0"
    secretAnswer="Not a very good one."
    secretQuestion="Is this a secret question?" />
```

Example 3.28. Get User Secret Response: JSON

```
{
  "secretQuestion": "Is this a secret question?",
  "secretAnswer": "Not a very good one."
}
```

Table 3.11. Get User Secret Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	

This operation does not require a request body.

3.2.7. Set User Secret

Verb	URI	Description
PUT	`/users/{userId}/secret`	Sets a user's secret question and answer.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.29. Set User Secret Request: XML

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

<secret xmlns="http://idm.api.rackspace.com/v1.0"
    secretAnswer="Not a very good one."
    secretQuestion="Is this a secret question?" />
```

Example 3.30. Set User Secret Request: JSON

```
{
  "secretQuestion": "Is this a secret question?",
  "secretAnswer": "Not a very good one."
}
```

Table 3.12. Set User Secret Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	

This operation does not return a response body.

3.2.8. Get User's Password

Verb	URI	Description
GET	/users/{userId}/passwordcredentials	Get a user's password credentials.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.31. Get Password Credentials Response: XML

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

<passwordCredentials xmlns="http://idm.api.rackspace.com/v1.0" >
    <currentPassword password="mypassword" />
</passwordCredentials>
```

Example 3.32. Get Password Credentials Response: JSON

```
{
  "currentPassword": {
    "password": "mypassword"
  }
}
```

Table 3.13. Get User's Password Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
userId	template	String	

This operation does not require a request body.

3.2.9. Set User's Password

Verb	URI	Description
PUT	/users/{userId}/passwordcredentials	Sets a user's password credentials.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.33. Set User Credential Request: XML

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

<passwordCredentials xmlns="http://idm.api.rackspace.com/v1.0"
    verifyCurrentPassword="true">
    <newPassword password="newpassword" />
    <currentPassword password="oldpassword" />
</passwordCredentials>
```

Example 3.34. Set User Credential Request: JSON

```
{
  "newPassword": {
    "password": "newpassword"
  },
  "currentPassword": {
    "password": "oldpassword"
  },
  "verifyCurrentPassword" : "true"
}
```

Table 3.14. Set User's Password Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
userId	template	String	

This operation does not return a response body.

3.2.10. Reset User's Password

Verb	URI	Description
POST	`/users/{userId}/passwordcredentials`	Reset a user's password credentials.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.35. Reset User Credential Response: XML

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

<passwordCredentials xmlns="http://idm.api.rackspace.com/v1.0" >
    <currentPassword password="mypassword" />
</passwordCredentials>
```

Example 3.36. Reset User Credential Response: JSON

```
{
  "currentPassword": {
    "password": "mypassword"
  }
}
```

Table 3.15. Reset User's Password Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	

This operation does not require a request body.

3.2.11. Get Password Recovery Token

Verb	URI	Description
GET	/users/{userId}/passwordcredentials/recoverytoken	Get a token that can be used to reset a user's password credentials.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.37. Get Credentials Recovery Token Response: XML

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

<token xmlns="http://idm.api.rackspace.com/v1.0"
    expires_in="3600"
    id="309487987f0892397a9439875900b" />
```

Example 3.38. Get Credentials Recovery Token Response: JSON

```
{
  "id": "309487987f0892397a9439875900b",
  "expiresIn": 3600
}
```

Table 3.16. Get Password Recovery Token Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
userId	template	String	

This operation does not require a request body.

3.2.12. Get a List of delegated refresh Tokens for a User

Verb	URI	Description
GET	`/users/{userId}/delegatedrefreshtokens`	Get a list of a user's delegated refresh tokens.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.39. Delegated Refresh Tokens Response: XML

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

<delegatedTokens  xmlns="http://idm.api.rackspace.com/v1.0">
    <delegatedToken
        expires="2010-11-02T03:32:15-05:00"
        id="309487987f0892397a9439875900b"
        clientId="ab4820dhcb39347" />
    <delegatedToken
        expires="2010-11-02T03:32:15-05:00"
        id="27aaf6789a9dcb789879972729abf"
        clientId="af62785da123567" />
</delegatedTokens>
```

Example 3.40. Delegated Refresh Tokens Response: JSON

```
{ "delegatedToken": [
   {
     "id": "309487987f0892397a9439875900b",
     "expires": "2010-11-02T03:32:15-05:00",
     "clientId": "ab4820dhcb39347"
   },
   {
     "id": "27aaf6789a9dcb789879972729abf",
     "expires": "2010-11-02T03:32:15-05:00",
     "clientId": "af62785da123567"
   }]
}
```

Table 3.17. Get a List of delegated refresh Tokens for a User Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	

This operation does not require a request body.

3.2.13. Get a Delegated Refresh Token for a User

Verb	URI	Description
GET	`/users/{userId}/delegatedrefreshtokens/{tokenId}`	Get a user's delegated refresh token.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.41. Delegated Refresh Token Response: XML

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

<delegatedToken xmlns="http://idm.api.rackspace.com/v1.0"
    expires="2010-11-02T03:32:15-05:00"
    id="309487987f0892397a9439875900b"
    clientId="ab4820dhcb39347" />
```

Example 3.42. Delegated Refresh Token Response: JSON

```
{
  "id": "309487987f0892397a9439875900b",
  "expires": "2010-11-02T03:32:15-05:00",
  "clientId": "ab4820dhcb39347"
}
```

Table 3.18. Get a Delegated Refresh Token for a User Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	
`tokenId`	template	String	

This operation does not require a request body.

3.2.14. Delete a Delegated Refresh Token for a User

Verb	URI	Description
DELETE	`/users/{userId}/delegatedrefreshtokens/{tokenId}`	Deletes a user's delegated refresh token.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.19. Delete a Delegated Refresh Token for a User Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	
`tokenId`	template	String	

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

3.2.15. Get the Applications that a User has Provisioned

Verb	URI	Description
GET	`/users/{userId}/applications`	Get the applications that have been provisioned for a user.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.43. Get User Applications Response: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0">
 <application
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347"/>
 <application
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="632h389cv902bde"/>
</application>
```

Example 3.44. Get User Applications Response: JSON

```
{
  "application": [
    {
      "clientId": "ab4820dhcb39347",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
    },
    {
      "clientId": "632h389cv902bde",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
    }
  ]
}
```

Table 3.20. Get the Applications that a User has Provisioned Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	

This operation does not require a request body.

3.2.16. Provision an Application for a User

Verb	URI	Description
PUT	`/users/{userId}/applications/{applicationId}`	Provision an application for a user.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.21. Provision an Application for a User Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	
`applicationId`	template	String	

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

3.2.17. Remove Provisioned Application from a User

Verb	URI	Description
DELETE	`/users/{userId}/applications/{applicationId}`	Removed a provisioned application from a user.

Normal Response Code(s): 204

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

Table 3.22. Remove Provisioned Application from a User Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	
`applicationId`	template	String	

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

3.2.18. Get User Tenants Roles

Verb	URI	Description
GET	/users/{userId}/tenants/roles?applicationId=*string*&tenantId=*string*	Get a list of user's roles on all tenants.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.45. Get User Tenant Roles Response: XML

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

<roles xmlns="http://idm.api.rackspace.com/v1.0">
    <role name="network_admin" applicationId="compute" tenantId="332"/>
    <role name="technical_reviewer" tenantId="57884" applicationId="swift"/>
    <role name="admin" tenantId="cf_89983" applicationId="billing"/>
</roles>
```

Example 3.46. Get User Tenant Roles Response: JSON

```json
{
  "role": [
    {
      "name": "network_admin",
      "applicationId": "compute",
      "tenantId": "332"
    },
    {
      "name": "technical_reviewer",
      "tenantId": "57884",
      "applicationid": "swift"
    }
    {
      "name": "admin",
      "tenantId": "cf_89983",
      "applicationId": "billing"
    }
  ]
}
```

Table 3.23. Get User Tenants Roles Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
applicationId	query	String	Filter list of user tenant roles by applicationId Optional.
tenantId	query	String	Filter list of user tenant roles by tenantId Optional.
userId	template	String	

3.2.19. Grant User Role on Tenant

Verb	URI	Description
PUT	`/users/{userId}/tenants/{tenantId}/roles/{roleId}`	Grant user a role on the tenant

Normal Response Code(s): 204

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

Table 3.24. Grant User Role on Tenant Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	
`tenantId`	template	String	
`roleId`	template	String	

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

3.2.20. Revoke a User Role on Tenant

Verb	URI	Description
DELETE	`/users/{userId}/tenants/{tenantId}/roles/{roleId}`	Revoke a user's role on a tenant.

Normal Response Code(s): 204

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

Table 3.25. Revoke a User Role on Tenant Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	
`tenantId`	template	String	
`roleId`	template	String	

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

3.2.21. Get User Global Roles

Verb	URI	Description
GET	/users/{userId}/roles?applicationId=*string*	Get a list of user's global roles.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.47. Get User Roles Response: XML


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

<roles xmlns="http://idm.api.rackspace.com/v1.0">
    <role id="23" name="network_admin" applicationId="compute"/>
    <role id="24" name="technical_reviewer" applicationId="swift"/>
    <role id="56" name="admin" />
</roles>
```


Example 3.48. Get User Roles Response: JSON


```
{
  "role": [
    {
      "name": "network_admin",
      "applicationId": "compute"
      "id": "23"
    },
    {
      "name": "technical_reviewer",
      "applicationid": "swift"
      "id": "24"
    }
    {
      "name": "admin",
      "id": "56"
    }
  ]
}
```


Table 3.26. Get User Global Roles Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
applicationId	query	String	Filter list of user roles by applicationId Optional.
userId	template	String	

3.2.22. Grant User a Global Role

Verb	URI	Description
PUT	`/users/{userId}/roles/{roleId}`	Grant user a global role.

Normal Response Code(s): 204

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

A global role is one that is not restricted just to a tenant. A global role could be application specific. A user could have the global admin role for the nova application, or a global admin role for all applications.

Table 3.27. Grant User a Global Role Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	
`roleId`	template	String	

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

3.2.23. Remove User's Global Role

Verb	URI	Description
DELETE	`/users/{userId}/roles/{roleId}`	Remove global role from user.

Normal Response Code(s): 204

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

Table 3.28. Remove User's Global Role Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`userId`	template	String	
`roleId`	template	String	

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

3.3. Application Operations

The operations described in this section allows service developers manage applications.

Verb	URI	Description
GET	`/applications?name=`*`string`*`&offset=`*`int`*`&limit=`*`int`*	Gets a list of applications.
POST	`/applications`	Add an application.
GET	`/applications/{applicationId}`	Get an application.
PUT	`/applications/{applicationId}`	Update an application.
DELETE	`/applications/{applicationId}`	Delete as application.
POST	`/applications/{applicationId}/secretcredentials`	Reset application secret.
GET	`/applications/{applicationId}/applications`	Get all applications that have been provisioned for this application..
PUT	`/applications/{applicationId}/applications/{provisionedApplicationId}`	Provision an application for this application.
DELETE	`/applications/{applicationId}/applications/{provisionedApplicationId}`	Removes a provisioned application from this application.
GET	`/applications/{applicationId}/tenants/roles?applicationId=`*`string`*`&tenantId=`*`string`*	Get a list of applications's roles on all tenants.
PUT	`/applications/{applicationId}/tenants/{tenantId}/roles/{roleId}`	Grant application a role on the tenant
DELETE	`/applications/{applicationId}/tenants/{tenantId}/roles/{roleId}`	Revoke an application's role on a tenant.
GET	`/applications/{applicationId}/roles?applicationId=`*`string`*	Get a list of aplications's global roles.

Verb	URI	Description
PUT	`/applications/{applicationId}/roles/{roleId}`	Grant application a global role.
DELETE	`/applications/{applicationId}/roles/{roleId}`	Remove global role from application.

3.3.1. Get Applications

Verb	URI	Description
GET	/applications?name=*string*&offset=*int*&limit=*int*	Gets a list of applications.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.49. Get Applications Response: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0">
 <application
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347"/>
 <application
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="632h389cv902bde"/>
</application>
```

Example 3.50. Get Applications Response: JSON

```
{
  "application": [
    {
      "clientId": "ab4820dhcb39347",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
    },
    {
      "clientId": "632h389cv902bde",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
    }
  ]
}
```

Table 3.29. Get Applications Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
name	query	String	Allows you search for an application by name Optional.
offset	query	Int	0 based index to start list Optional.
limit	query	Int	Number of records to be returned Optional.

3.3.2. Add an Application

Verb	URI	Description
POST	`/applications`	Add an application.

Normal Response Code(s): 200

Error Response Code(s): applicationNameConflict (409), serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503)

Example 3.51. Add Application Request: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0"
    name="Test Application2"
    customerId="RCN-000-000-000"
    description="Used for adding all applications in Rackspace" />
```

Example 3.52. Add Application Request: JSON

```
{
  "customerId": "RCN-000-000-000",
  "name": "Test Application2",
  "description": "Used for adding all applications in Rackspace"
}
```

Example 3.53. Add Application Request: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0"
    softDeleted="false"
    enabled="true"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347">
    <secretCredentials clientSecret="3af738fbeiwu23" />
</application>
```

Example 3.54. Add Application Request: JSON

```
{
  "secretCredentials": {
    "clientSecret": "3af738fbeiwu23"
  },
  "clientId": "ab4820dhcb39347",
  "customerId": "RCN-000-000-000",
  "name": "Test Application2",
  "enabled": true,
  "softDeleted": false
}
```

Table 3.30. Add an Application Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access.

Name	Style	Type	Description
			Required.

Table 3.31. Add an Application Response Parameters

Name	Style	Type	Description
`Location`	header	String	Location to the uri of the newly created resource Required.

3.3.3. Get an Application

Verb	URI	Description
GET	/applications/{applicationId}	Get an application.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.55. Get Application Response: XML


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

<application xmlns="http://idm.api.rackspace.com/v1.0"
    softDeleted="false"
    enabled="true"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347" />
```


Example 3.56. Get Application Response: JSON


```json
{
  "clientId": "ab4820dhcb39347",
  "customerId": "RCN-000-000-000",
  "name": "Test Application2",
  "enabled": true,
  "softDeleted": false
}
```


Table 3.32. Get an Application Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
applicationId	template	String	

This operation does not require a request body.

3.3.4. Update an Application

Verb	URI	Description
PUT	`/applications/{applicationId}`	Update an application.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.57. Update Application Request: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0"
    scope="cloud_servers"
    clientId="ab4820dhcb39347" />
```

Example 3.58. Update Application Request: JSON

```
{
  "clientId": "ab4820dhcb39347",
  "scope": "cloud_servers"
}
```

Example 3.59. Update Application Response: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0"
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    iname="@Rackspace*Rackspace*ControlPanel"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347"
    scope="cloud_servers" />
```

Example 3.60. Update Application Response: JSON

```
{
  "clientId": "ab4820dhcb39347",
  "customerId": "RCN-000-000-000",
  "name": "Test Application2",
  "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001",
  "scope": "cloud_servers",
  "status": "ACTIVE",
  "locked": false,
  "softDeleted": false
}
```

Table 3.33. Update an Application Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access.

Name	Style	Type	Description
			Required.
`applicationId`	template	String	

3.3.5. Delete an Application

Verb	URI	Description
DELETE	`/applications/{applicationId}`	Delete as application.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.34. Delete an Application Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`applicationId`	template	String	

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

3.3.6. Reset Application's Secret

Verb	URI	Description
POST	`/applications/{applicationId}/secretcredentials`	Reset application secret.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.61. Reset Application Secret Response: XML

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

<secretCredentials xmlns="http://idm.api.rackspace.com/v1.0"
 clientSecret="cncv9823823bfb" />
```

Example 3.62. Reset Application Secret Response: JSON

```
{
  "clientSecret": "cncv9823823bfb"
}
```

Table 3.35. Reset Application's Secret Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`applicationId`	template	String	

This operation does not require a request body.

3.3.7. Get the Applications that an Application has Provisioned

Verb	URI	Description
GET	/applications/{applicationId}/ applications	Get all applications that have been provisioned for this application..

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.63. Get Provisioned Applications Response: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0">
 <application
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347"/>
 <application
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="632h389cv902bde"/>
</application>
```

Example 3.64. Get Provisioned Applications Response: JSON

```
{
  "application": [
    {
      "clientId": "ab4820dhcb39347",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
    },
    {
      "clientId": "632h389cv902bde",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
    }
  ]
}
```

Table 3.36. Get the Applications that an Application has Provisioned Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
applicationId	template	String	

This operation does not require a request body.

3.3.8. Provision an Application for an Application

Verb	URI	Description
PUT	`/applications/{applicationId}/applications/{provisionedApplicationId}`	Provision an application for this application.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.37. Provision an Application for an Application Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`applicationId`	template	String	
`provisionedApplicationId`	template	String	

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

3.3.9. Remove Provisioned Application from an Application

Verb	URI	Description
DELETE	`/applications/{applicationId}/applications/{provisionedApplicationId}`	Removes a provisioned application from this application.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.38. Remove Provisioned Application from an Application Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`applicationId`	template	String	
`provisionedApplicationId`	template	String	

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

3.3.10. Get Application Tenants Roles

Verb	URI	Description
GET	/applications/{applicationId}/tenants/roles?applicationId=*string*&tenantId=*string*	Get a list of applications's roles on all tenants.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.65. Get Application Tenant Roles Response: XML


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

<roles xmlns="http://idm.api.rackspace.com/v1.0">
    <role name="network_admin" applicationId="compute" tenantId="332"/>
    <role name="technical_reviewer" tenantId="57884" applicationId="swift"/>
    <role name="admin" tenantId="cf_89983" applicationId="billing"/>
</roles>
```


Example 3.66. Get Application Tenant Roles Response: JSON


```
{
  "role": [
    {
      "name": "network_admin",
      "applicationId": "compute",
      "tenantId": "332"
    },
    {
      "name": "technical_reviewer",
      "tenantId": "57884",
      "applicationid": "swift"
    }
    {
      "name": "admin",
      "tenantId": "cf_89983",
      "applicationId": "billing"
    }
  ]
}
```


Table 3.39. Get Application Tenants Roles Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`applicationId`	query	String	Filter list of application tenant roles by applicationId Optional.
`tenantId`	query	String	Filter list of application tenant roles by tenantId Optional.

Name	Style	Type	Description
`applicationId`	template	String	

3.3.11. Grant Application Role on Tenant

Verb	URI	Description
PUT	`/applications/{applicationId}/tenants/{tenantId}/roles/{roleId}`	Grant application a role on the tenant

Normal Response Code(s): 204

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

Table 3.40. Grant Application Role on Tenant Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`applicationId`	template	String	
`tenantId`	template	String	
`roleId`	template	String	

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

3.3.12. Revoke an Application Role on Tenant

Verb	URI	Description
DELETE	`/applications/{applicationId}/tenants/{tenantId}/roles/{roleId}`	Revoke an application's role on a tenant.

Normal Response Code(s): 204

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

Table 3.41. Revoke an Application Role on Tenant Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`applicationId`	template	String	
`tenantId`	template	String	
`roleId`	template	String	

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

3.3.13. Get Application Global Roles

Verb	URI	Description
GET	`/applications/{applicationId}/roles?applicationId=string`	Get a list of aplications's global roles.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.67. Get Application Roles Response: XML

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

<roles xmlns="http://idm.api.rackspace.com/v1.0">
    <role id="23" name="network_admin" applicationId="compute"/>
    <role id="24" name="technical_reviewer" applicationId="swift"/>
    <role id="56" name="admin" />
</roles>
```

Example 3.68. Get Application Roles Response: JSON

```
{
  "role": [
    {
      "name": "network_admin",
      "applicationId": "compute"
      "id": "23"
    },
    {
      "name": "technical_reviewer",
      "applicationid": "swift"
      "id": "24"
    }
    {
      "name": "admin",
      "id": "56"
    }
  ]
}
```

Table 3.42. Get Application Global Roles Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`applicationId`	query	String	Filter list of aplication roles by applicationId Optional.
`applicationId`	template	String	

3.3.14. Grant Application a Global Role

Verb	URI	Description
PUT	`/applications/{applicationId}/roles/{roleId}`	Grant application a global role.

Normal Response Code(s): 204

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

A global role is one that is not restricted just to a tenant. A global role could be application specific. An application could have the global admin role for the nova application, or a global admin role for all applications.

Table 3.43. Grant Application a Global Role Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`applicationId`	template	String	
`roleId`	template	String	

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

3.3.15. Remove Application's Global Role

Verb	URI	Description
DELETE	/applications/{applicationId}/roles/{roleId}	Remove global role from application.

Normal Response Code(s): 204

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

Table 3.44. Remove Application's Global Role Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
applicationId	template	String	
roleId	template	String	

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

3.4. Customer Identity Profile Operations

The operations described in this section allows service developers manage customer identity profiles.

Verb	URI	Description
POST	/customeridentityprofiles/{customerId}	Adds a customer's identity profile.
GET	/customeridentityprofiles/{customerId}	Gets a customer's identity profile.
PUT	/customeridentityprofiles/{customerId}	Updates a customer's identity profile.
DELETE	/customeridentityprofiles/{customerId}	Delete a customer's identity profile.
GET	/customeridentityprofiles/{customerId}/passwordrotationpolicy	Gets the password rotation policy for a customer.
PUT	/customeridentityprofiles/{customerId}/passwordrotationpolicy	Sets the password rotation policy for a customer.
GET	/customeridentityprofiles/{customerId}/users?offset=*int*&limit=*int*	Gets a list of users that this customer owns.
GET	/customeridentityprofiles/{customerId}/applications?offset=*int*&limit=*int*	Gets a list of applications that this customer owns.

3.4.1. Adds a Customer's Identity Profile

Verb	URI	Description
POST	`/customeridentityprofiles/{customerId}`	Adds a customer's identity profile.

Normal Response Code(s): 201

Error Response Code(s): customerIdConflict (409), serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503)

Example 3.69. Customer Identity Profile Add Request: XML

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

<customerIdentityProfile xmlns="http://idm.api.rackspace.com/v1.0"
    enabled="true"
    rcn="RCN-123-549-034" />
```

Example 3.70. Customer Identity Profile Add Request: JSON

```
{
  "rcn": "RCN-123-549-034",
  "enabled": true,
}
```

Table 3.45. Adds a Customer's Identity Profile Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`customerId`	template	String	This is the customer's RCN.

Table 3.46. Adds a Customer's Identity Profile Response Parameters

Name	Style	Type	Description
`Location`	header	String	Location to the uri of the newly created customer identity profile Required.

This operation does not return a response body.

3.4.2. Get a Customer's Identity Profile

Verb	URI	Description
GET	`/customeridentityprofiles/{customerId}`	Gets a customer's identity profile.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.71. Get Customer Identity Profile Response: XML

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

<customerIdentityProfile xmlns="http://idm.api.rackspace.com/v1.0"
    softDeleted="false"
    enabled="false"
    rcn="RCN-123-549-034" />
```

Example 3.72. Get Customer Identity Profile Response: JSON

```
{
  "rcn": "RCN-123-549-034",
  "enabled": false,
  "softDeleted": false
}
```

Table 3.47. Get a Customer's Identity Profile Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`customerId`	template	String	This is the customer's RCN.

This operation does not require a request body.

3.4.3. Delete a Customer's Identity Profile

Verb	URI	Description
DELETE	`/customeridentityprofiles/{customerId}`	Delete a customer's identity profile.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Table 3.48. Delete a Customer's Identity Profile Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`customerId`	template	String	This is the customer's RCN.

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

3.4.4. Updates a Customer's Identity Profile

Verb	URI	Description
PUT	`/customeridentityprofiles/{customerId}`	Updates a customer's identity profile.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.73. Customer Identity Profile Update Request: XML

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

<customerIdentityProfile xmlns="http://idm.api.rackspace.com/v1.0"
    softDeleted="false"
    enabled="false"
    rcn="RCN-123-549-034" />
```

Example 3.74. Customer Identity Profile Update Request: JSON

```
{
  "rcn": "RCN-123-549-034",
  "enabled": false,
  "softDeleted": false
}
```

Table 3.49. Updates a Customer's Identity Profile Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`customerId`	template	String	This is the customer's RCN.

This operation does not return a response body.

3.4.5. Get Password Rotation Policy

Verb	URI	Description
GET	`/customeridentityprofiles/{customerId}/passwordrotationpolicy`	Gets the password rotation policy for a customer.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.75. Password Rotation Policy Response: XML

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

<passwordRotationPolicy xmlns="http://idm.api.rackspace.com/v1.0"
    enabled="true"
    duration="90" />
```

Example 3.76. Password Rotation Policy Response: JSON

```
{
  "enabled": true,
  "duration": "60"
}
```

Table 3.50. Get Password Rotation Policy Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`customerId`	template	String	This is the customer's RCN.

This operation does not require a request body.

3.4.6. Set Password Rotation Policy

Verb	URI	Description
PUT	`/customeridentityprofiles/{customerId}/passwordrotationpolicy`	Sets the password rotation policy for a customer.

Normal Response Code(s): 204

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.77. Password Rotation Policy Request: XML

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

<passwordRotationPolicy xmlns="http://idm.api.rackspace.com/v1.0"
    enabled="true"
    duration="90" />
```

Example 3.78. Password Rotation Policy Request: JSON

```
{
  "enabled": true,
  "duration": "60"
}
```

Table 3.51. Set Password Rotation Policy Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`customerId`	template	String	This is the customer's RCN.

This operation does not return a response body.

3.4.7. Get Users

Verb	URI	Description
GET	`/customeridentityprofiles/{customerId}/users?offset=`*int*`&limit=`*int*	Gets a list of users that this customer owns.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.79. Get Users Response: XML

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

<users xmlns="http://idm.api.rackspace.com/v1.0"
    limit="10"
    offset="0"
    totalRecords="2">
    <user
        softDeleted="false" locked="false"
        status="ACTIVE" region="America/Chicago"
        iname="@Example.Smith*John"
        inum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!1111"
        prefLanguage="US_en" displayName="John Smith"
        lastName="Smith" middleName="Quincy"
        firstName="John" personId="RPN-111-111-111"
        email="john.smith@example.org"
        customerInum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE"
        customerId="RCN-000-000-000"
        username="jqsmith" />
    <user softDeleted="false" locked="false"
        status="ACTIVE" region="America/Chicago"
        iname="@Example.Anderson*Bob"
        inum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!2222"
        prefLanguage="US_en" displayName="Bob Anderson"
        lastName="Anderson" middleName="Mark"
        firstName="Bob" personId="RPN-111-111-222"
        email="bob.anderson@example.org"
        customerInum="@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE"
        customerId="RCN-000-000-000"
        username="bmanderson" />
</users>
```

Example 3.80. Get Users Response: JSON

```
{"user": [
    {
      "username": "jqsmith",
      "customerId": "RCN-000-000-000",
      "customerInum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE",
      "email": "john.smith@example.org",
      "personId": "RPN-111-111-111",
      "firstName": "John",
      "middleName": "Quincy",
      "lastName": "Smith",
      "displayName": "John Smith",
```

```
      "prefLanguage": "US_en",
      "inum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!1111",
      "iname": "@Example.Smith*John",
      "region": "America/Chicago",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    },
    {
      "username": "bmanderson",
      "customerId": "RCN-000-000-000",
      "customerInum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE",
      "email": "bob.anderson@example.org",
      "personId": "RPN-111-111-222",
      "firstName": "Bob",
      "middleName": "Mark",
      "lastName": "Anderson",
      "displayName": "Bob Anderson",
      "prefLanguage": "US_en",
      "inum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!2222",
      "iname": "@Example.Anderson*Bob",
      "timeZone": "America/Chicago",
      "region": "SAT",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    }
  ],
  "totalRecords": 2,
  "offset": 0,
  "limit": 10
}
```


Table 3.52. Get Users Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
offset	query	Int	0 based index to start list Optional.
limit	query	Int	Number of records to be returned Optional.
customerId	template	String	This is the customer's RCN.

3.4.8. Get Applications

Verb	URI	Description
GET	/customeridentityprofiles/{customerId}/applications?offset=*int*&limit=*int*	Gets a list of applications that this customer owns.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.81. Get Applications Response: XML


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

<application xmlns="http://idm.api.rackspace.com/v1.0">
 <application
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347"/>
 <application
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="632h389cv902bde"/>
</application>
```


Example 3.82. Get Applications Response: JSON


```json
{
  "application": [
    {
      "clientId": "ab4820dhcb39347",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
    },
    {
      "clientId": "632h389cv902bde",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
    }
  ]
}
```


Table 3.53. Get Applications Request Parameters

Name	Style	Type	Description
Authorization	header	String	You need a valid token for access. Required.
offset	query	Int	0 based index to start list Optional.
limit	query	Int	Number of records to be returned Optional.
customerId	template	String	This is the customer's RCN.

3.5. Password Operations

The operations described in this section allows service developers manage password policies.

Verb	URI	Description
GET	`/passwordrules`	Get password rules.
POST	`/passwordrules/validation`	Validate password to ensure it conforms to password rules.

3.5.1. Get Password Rules

Verb	URI	Description
GET	`/passwordrules`	Get password rules.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.83. Get Password Rules Response: XML

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

<passwordRuleResults
    xmlns="http://idm.api.rackspace.com/v1.0">
    <passwordRuleResults
        ruleMessage="Password must be at least 7 characters."
        ruleName="Minimum Length"
        ruleId="1"
        passed="true" />
    <passwordRuleResults
        ruleMessage="Password must contain a lowercase."
        ruleName="Lowercase Character"
        ruleId="2"
        passed="true" />
</passwordRuleResults>
```

Example 3.84. Get Password Rules Response: JSON

```
{
  "passwordRuleResult": [
    {
      "passed": true,
      "ruleId": 1,
      "ruleName": "Minimum Length",
      "ruleMessage": "Password must be at least 7 characters long."
    },
    {
      "passed": true,
      "ruleId": 2,
      "ruleName": "Lowercase Character",
      "ruleMessage": "Password must contain a lowercase character."
    }
  ]
}
```

Table 3.54. Get Password Rules Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.

This operation does not require a request body.

3.5.2. Password Validation Check

Verb	URI	Description
POST	/passwordrules/validation	Validate password to ensure it conforms to password rules.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.85. Password Validation Request: XML

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

<userPassword xmlns="http://idm.api.rackspace.com/v1.0"
    password="newpassword" />
```

Example 3.86. Password Validation Request: JSON

```
{
  "password": "newpassword"
}
```

Example 3.87. Password Validation Response: XML

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

<passwordValidation xmlns="http://idm.api.rackspace.com/v1.0"
    validPassword="true">
    <passwordRuleResults>
        <passwordRuleResults
          ruleMessage="Password must be at least 7 characters long."
          ruleName="Minimum Length"
          ruleId="1"
          passed="true" />
        <passwordRuleResults
          ruleMessage="Password must contain a lowercase character."
          ruleName="Lowercase Character"
          ruleId="2"
          passed="true" />
    </passwordRuleResults>
</passwordValidation>
```

Example 3.88. Password Validation Response: JSON

```
{"passwordRuleResults":{
    "passwordRuleResults":[
        {
            "passed":true,
            "ruleId":1,"ruleName":
            "Mininumn Length",
            "ruleMessage":"The password must be at least 7 characters long"
        },
        {
            "passed":false,
```

```
            "ruleId":2,"ruleName":
            "Uppercase Rule","ruleMessage":
            "The password must contain an uppercase charater"
        }
     ]},
     "validPassword":false
}
```

Table 3.55. Password Validation Check Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.

3.6. Role Operations

The operations described in this section allows service developers manage roles.

Verb	URI	Description
GET	/roles?name=*string*& applicationId=*string*	Get all roles.
POST	/roles	Add a new role.
GET	/roles/{roleId}	Get a Role
POST	/roles/{roleId}	Modify an existing role.
POST	/roles/{roleId}	Delete an existing role.

3.6.1. Get All Roles

Verb	URI	Description
GET	`/roles?name=`*string*`&` `applicationId=`*string*	Get all roles.

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.89. Get Roles Response: XML

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

<roles xmlns="http://idm.api.rackspace.com/v1.0">
    <role id="23" name="network_admin" applicationId="compute"/>
    <role id="24" name="technical_reviewer" applicationId="swift"/>
    <role id="56" name="admin" />
</roles>
```

Example 3.90. Get Roles Response: JSON

```
{
  "role": [
    {
      "name": "network_admin",
      "applicationId": "compute"
      "id": "23"
    },
    {
      "name": "technical_reviewer",
      "applicationid": "swift"
      "id": "24"
    }
    {
      "name": "admin",
      "id": "56"
    }
  ]
}
```

Table 3.56. Get All Roles Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`name`	query	String	Name of the role Optional.
`applicationId`	query	String	Application that role is tied to. Optional.

3.6.2. Add a Role

Verb	URI	Description
POST	`/roles`	Add a new role.

Normal Response Code(s): 201

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

Can also add application specific roles by specifying the application id in the request payload.

Example 3.91. Add Role Request: XML

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

<role xmlns="http://idm.api.rackspace.com/v1.0" name="admin"
 description="Can admin the heck out of anything" applicationId=
"18e7a7032733486cd32f472d7bd58f709ac0d221" />
```

Example 3.92. Add Role Request: JSON

```
{
      "name": "admin",
      "description": "Can admin the heck out of anything",
      "applicationid": "18e7a7032733486cd32f472d7bd58f709ac0d221"
}
```

Table 3.57. Add a Role Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.

Table 3.58. Add a Role Response Parameters

Name	Style	Type	Description
`Location`	header	String	Location to the uri of the newly created role Required.

This operation does not return a response body.

3.6.3. Get Role

Verb	URI	Description
GET	`/roles/{roleId}`	Get a Role

Normal Response Code(s): 200

Error Response Code(s): serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

Example 3.93. Get Role Response: XML

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

<role id="452" name="admin" />
```

Example 3.94. Get Role Response: JSON

```
{
     "id": "452",
     "name": "admin"
}
```

Table 3.59. Get Role Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`roleId`	template	String	

This operation does not require a request body.

3.6.4. Modify a Role

Verb	URI	Description
POST	`/roles/{roleId}`	Modify an existing role.

Normal Response Code(s): 204

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

Example 3.95. Update Role Request: XML

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

<role xmlns="http://idm.api.rackspace.com/v1.0" name="admin"
 description="Can admin the heck out of anything" applicationId=
"18e7a7032733486cd32f472d7bd58f709ac0d221" />
```

Example 3.96. Update Role Request: JSON

```
{
      "name": "admin",
      "description": "Can admin the heck out of anything",
      "applicationid": "18e7a7032733486cd32f472d7bd58f709ac0d221"
}
```

Table 3.60. Modify a Role Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`roleId`	template	String	

This operation does not return a response body.

3.6.5. Delete a Role

Verb	URI	Description
POST	`/roles/{roleId}`	Delete an existing role.

Normal Response Code(s): 204

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

Table 3.61. Delete a Role Request Parameters

Name	Style	Type	Description
`Authorization`	header	String	You need a valid token for access. Required.
`roleId`	template	String	

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