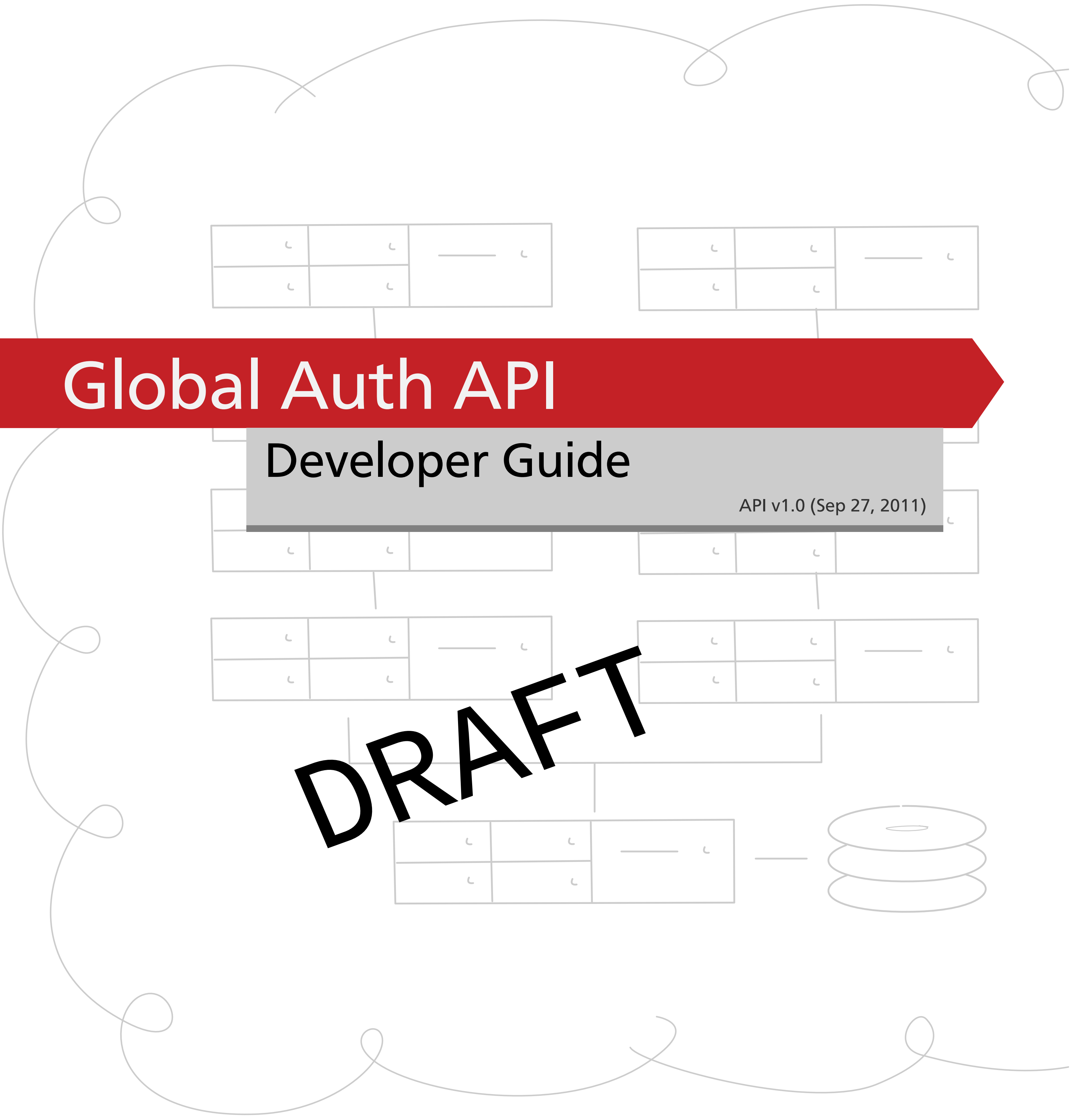

Global Auth API

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 word "OAuth" followed by a space followed by the access token id that belongs to the entity making the call. For example, here is an sample request for getting a user's details.

Example 2.12. Authorization Header Request Format

```
        GET /v1.0/users/joeuser HTTP/1.1
Host: {host name TBD}
Accept: application/xml
Authorization: OAuth ab48a9efdfedb23ty3494
```

3. Service API (Client Operations)

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

3.1. Token Operations

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

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

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)

Example 3.1. Auth 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 3.2. Auth Request: JSON

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

Example 3.3. Auth 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 3.4. Auth Response: JSON

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

4. 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`*	Gets a list of users.
POST	`/users`	Adds a new user. If the customer specified does not exist, adds this first user as the admin user.
GET	`/users/{userId}`	Get a user.
PUT	`/users/{userId}`	Update a user.
DELETE	`/users/{userId}`	Delete a user.
PUT	`/users/{userId}/softdeleted`	Soft deletes a user.
PUT	`/users/{userId}/lock`	Lock or unlock a user.
PUT	`/users/{userId}/status`	Set a user's status to ACTIVE or INACTIVE.
PUT	`/users/{userId}/secret`	Sets a user's secret question and answer.
GET	`/users/{userId}/password`	Get a user's password.
PUT	`/users/{userId}/password`	Sets a user's password.
POST	`/users/{userId}/password`	Reset a user's password.
GET	`/users/{userId}/password/recoverytoken`	Get a token that can be used to reset a user's password.
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.
PUT	`/users/{userId}/applications/{applicationId}/roles/{roleId}`	Grant user a role on the application.
DELETE	`/users/{userId}/applications/{applicationId}/roles/{roleId}`	Revoke a user's role on the application.
GET	`/users/{userId}/roles?applicationId=`*`string`*	Get the applications that have been provisioned for a user.
Application Operations		
GET	`/applications?name=`*`string`*	Gets a list of applications.
POST	`/applications`	Add an application.

Verb	URI	Description
GET	`/applications/{applicationId}`	Get an application.
PUT	`/applications/{applicationId}`	Update an application.
DELETE	`/applications/{applicationId}`	Delete as application.
PUT	`/applications/{applicationId}/softdeleted`	Soft deletes an application.
GET	`/applications/{applicationId}/definedroles`	Get roles defined by an application.
PUT	`/applications/{applicationId}/definedroles/{roleId}`	Add a role defined by an application.
DELETE	`/applications/{applicationId}/definedroles/{roleId}`	Delete a role defined by an application.
POST	`/applications/{applicationId}/secret`	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.
PUT	`/applications/{applicationId}/applications/{provisionedApplicationId}/roles/{roleId}`	Grant application a role on another application.
DELETE	`/applications/{applicationId}/applications/{provisionedApplicationId}/roles/{roleId}`	Revoke an application's role on another application..
GET	`/applications/{applicationId}/roles?applicationId=`*`string`*	Get the applications that have been provisioned for this application.
Customer Identity Profile Operations		
GET	`/customeridentityprofiles/{customerId}`	Gets a customer's identity profile.
DELETE	`/customeridentityprofiles/{customerId}`	Delete a customer's identity profile.
PUT	`/customeridentityprofiles/{customerId}/locked`	Lock or unlock a customer's identity profile.
PUT	`/customeridentityprofiles/{customerId}/passwordrotationpolicy`	Sets the password rotation policy for a customer.
GET	`/customeridentityprofiles/{customerId}/users`	Gets a list of users that this customer owns.
GET	`/customeridentityprofiles/{customerId}/applications`	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.

4.1. 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.

4.1.1. 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)

Example 4.1. Auth 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 4.2. Auth Request: JSON

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

Example 4.3. Auth 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 4.4. Auth Response: JSON

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

4.1.2. 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 4.5. 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 4.6. 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 4.1. Validate Token Request Parameters

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

Name	Style	Description
`tokenId`	template	String.

This operation does not require a request body.

4.1.3. 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 4.2. Revoke Token Request Parameters

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

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

4.1.4. 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 4.3. Check if Token has access to an Application Request Parameters

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

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

4.1.5. 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 4.4. Check if Token has Role on an Application Request Parameters

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

4.2. User Operations

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

Verb	URI	Description
GET	`/users?username=`*`string`*	Gets a list of users.
POST	`/users`	Adds a new user. If the customer specified does not exist, adds this first user as the admin user.
GET	`/users/{userId}`	Get a user.
PUT	`/users/{userId}`	Update a user.
DELETE	`/users/{userId}`	Delete a user.
PUT	`/users/{userId}/softdeleted`	Soft deletes a user.
PUT	`/users/{userId}/lock`	Lock or unlock a user.
PUT	`/users/{userId}/status`	Set a user's status to ACTIVE or INACTIVE.
PUT	`/users/{userId}/secret`	Sets a user's secret question and answer.
GET	`/users/{userId}/password`	Get a user's password.
PUT	`/users/{userId}/password`	Sets a user's password.
POST	`/users/{userId}/password`	Reset a user's password.
GET	`/users/{userId}/password/recoverytoken`	Get a token that can be used to reset a user's password.
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.
PUT	`/users/{userId}/applications/{applicationId}/roles/{roleId}`	Grant user a role on the application.
DELETE	`/users/{userId}/applications/{applicationId}/roles/{roleId}`	Revoke a user's role on the application.
GET	`/users/{userId}/roles?applicationId=`*`string`*	Get the applications that have been provisioned for a user.

4.2.1. Get Users

Verb	URI	Description
GET	`/users?username=`*`string`*	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 4.7. 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 4.8. 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 4.5. Get Users Request Parameters

Name	Style	Description
`Authorization`	header	You need a valid token for access. String. Required.
`username`	query	Allows you search for a user by username String. Optional.

4.2.2. Add a User

Verb	URI	Description
POST	`/users`	Adds a new user. If the customer specified does not exist, adds this first user as the admin user.

Normal Response Code(s): 200

Error Response Code(s): emailConflict (409), usernameConflict (409), serviceFault (500), badRequest (400), unauthorized (401), forbidden (403), serviceUnavailable (503), itemNotFound (404)

The Username attribute, CustomerId attribute and Password element are required in this request. If a blank password is passed in the Password element, the API will generate a random password for the user. If the customer does not already exist in IdM, the system will create the customer and add this user as the admin user. The prefLanguage attribute defaults to "US_en" and the timeZone attribute defaults to "America/Chicago".

Example 4.9. 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?" />
    <password password="C@n+f001me!" />
</user>
```

Example 4.10. Add User Request: JSON

```
{
  "secret": {
    "secretAnswer": "Francis",
    "secretQuestion": "What is your dogs name?"
  },
  "password": {
    "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 4.11. Add 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" timeZone="America/Chicago"
 region="SAT" 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">
 <secret secretAnswer="Francis"
       secretQuestion="What is your dogs name?" />
    <password password="C@n+f001me!" />
</user>
```

Example 4.12. Add User Response: JSON

```
{
  "secret": {
    "secretQuestion": "What is your dogs name?",
    "secretAnswer": "sicnarF"
  },
  "password": {
    "password": "C@n+f001me!"
  },
  "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",
  "timeZone": "America/Chicago",
  "region": "SAT",
  "status": "ACTIVE",
  "locked": false,
  "softDeleted": false
}
```

Table 4.6. Add a User Request Parameters

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

4.2.3. 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 4.13. 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 4.14. 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 4.7. Get a User Request Parameters

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

This operation does not require a request body.

4.2.4. 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 4.15. 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 4.16. Update User Request: JSON

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

Example 4.17. 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 4.18. 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 4.8. Update a User Request Parameters

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

4.2.5. 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 4.9. Delete a User Request Parameters

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

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

4.2.6. Soft Delete a User

Verb	URI	Description
PUT	`/users/{userId}/softdeleted`	Soft deletes a user.

Normal Response Code(s): 204

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

Example 4.19. Soft Delete User Request: XML

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

<user xmlns="http://idm.api.rackspace.com/v1.0"
    softDeleted="true" />
```

Example 4.20. Soft Delete User Request: JSON

```
{
  "softDeleted": "true"
}
```

Table 4.10. Soft Delete a User Request Parameters

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

This operation does not return a response body.

4.2.7. Lock or Unlock a User

Verb	URI	Description
PUT	`/users/{userId}/lock`	Lock or unlock a user.

Normal Response Code(s): 204

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

Example 4.21. User Lock Request: XML

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

<user xmlns="http://idm.api.rackspace.com/v1.0"
    locked="true" />
```

Example 4.22. User Lock Request: JSON

```
{
  "locked": "true"
}
```

Table 4.11. Lock or Unlock a User Request Parameters

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

This operation does not return a response body.

4.2.8. Set User Status

Verb	URI	Description
PUT	`/users/{userId}/status`	Set a user's status to ACTIVE or INACTIVE.

Normal Response Code(s): 204

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

Example 4.23. Set User Status Request: XML

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

<user xmlns="http://idm.api.rackspace.com/v1.0"
    status="ACTIVE" />
```

Example 4.24. Set User Status Request: JSON

```
{
  "status": "INACTIVE"
}
```

Table 4.12. Set User Status Request Parameters

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

This operation does not return a response body.

4.2.9. 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 4.25. Set User Secret Request: XML

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

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

Example 4.26. Set User Secret Request: JSON

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

Table 4.13. Set User Secret Request Parameters

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

This operation does not return a response body.

4.2.10. Get User Password

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

Normal Response Code(s): 200

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

Example 4.27. Get User Password Response: XML

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

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

Example 4.28. Get User Password Response: JSON

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

Table 4.14. Get User Password Request Parameters

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

This operation does not require a request body.

4.2.11. Set User Password

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

Normal Response Code(s): 200

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

Example 4.29. Set User Password Request: XML

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

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

Example 4.30. Set User Password Request: JSON

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

Example 4.31. Set User Password Response: XML

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

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

Example 4.32. Set User Password Response: JSON

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

Table 4.15. Set User Password Request Parameters

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

4.2.12. Reset User Password

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

Normal Response Code(s): 200

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

Example 4.33. Reset User Password Response: XML

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

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

Example 4.34. Reset User Password Response: JSON

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

Table 4.16. Reset User Password Request Parameters

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

This operation does not require a request body.

4.2.13. Get Password Recovery Token

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

Normal Response Code(s): 200

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

Example 4.35. Get Password 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 4.36. Get Password Recovery Token Response: JSON

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

Table 4.17. Get Password Recovery Token Request Parameters

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

This operation does not require a request body.

4.2.14. 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 4.37. 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 4.38. 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 4.18. Get a List of delegated refresh Tokens for a User Request Parameters

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

This operation does not require a request body.

4.2.15. 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 4.39. 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 4.40. Delegated Refresh Token Response: JSON

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

Table 4.19. Get a Delegated Refresh Token for a User Request Parameters

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

This operation does not require a request body.

4.2.16. 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 4.20. Delete a Delegated Refresh Token for a User Request Parameters

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

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

4.2.17. 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 4.41. Delegated Refresh Token Response: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0">
 <application
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347"/>
 <application
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0002"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="632h389cv902bde"/>
</application>
```

Example 4.42. Delegated Refresh Token Response: JSON

```
{
  "application": [
    {
      "clientId": "ab4820dhcb39347",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
      "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    },
    {
      "clientId": "632h389cv902bde",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
      "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0002",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    }
  ]
}
```

Table 4.21. Get the Applications that a User has Provisioned Request Parameters

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

This operation does not require a request body.

4.2.18. Provision an Application for a User

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

Normal Response Code(s): 200

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

Example 4.43. Provision User Application Request: XML

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

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

Example 4.44. Provision User Application Request: JSON

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

Example 4.45. Provision User Application Response: XML

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

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

Example 4.46. Provision User Application Response: JSON

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

Table 4.22. Provision an Application for a User Request Parameters

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

This operation does not return a response body.

4.2.19. 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), itemNotFound (404)

Table 4.23. Remove Provisioned Application from a User Request Parameters

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

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

4.2.20. Grant User Role on Application

Verb	URI	Description
PUT	`/users/{userId}/applications/{applicationId}/roles/{roleId}`	Grant user a role on the application.

Normal Response Code(s): 204

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

Example 4.47. Grant User Role Request: XML

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

<role name="admin" tenantId="cf_78373" />
```

Example 4.48. Grant User Role Request: JSON

```
{
      "name": "admin",
      "tenantId": "cf_78373"
}
```

Table 4.24. Grant User Role on Application Request Parameters

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

This operation does not return a response body.

4.2.21. Revoke a User Role on Application

Verb	URI	Description
DELETE	`/users/{userId}/applications/{applicationId}/roles/{roleId}`	Revoke a user's role on the application.

Normal Response Code(s): 204

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

Table 4.25. Revoke a User Role on Application Request Parameters

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

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

4.2.22. Get User Roles

Verb	URI	Description
GET	`/users/{userId}/roles?applicationId=`*`string`*	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 4.49. Get User 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"/>
    <role name="technical_reviewer" tenantId="57884" applicationId="swift"/>
    <role name="admin" tenantId="cf_89983" applicationId="billing"/>
</roles>
```

Example 4.50. Get User Roles Response: JSON

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

Table 4.26. Get User Roles Request Parameters

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

4.3. Application Operations

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

Verb	URI	Description
GET	`/applications?name=`*`string`*	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.
PUT	`/applications/{applicationId}/softdeleted`	Soft deletes an application.
GET	`/applications/{applicationId}/definedroles`	Get roles defined by an application.
PUT	`/applications/{applicationId}/definedroles/{roleId}`	Add a role defined by an application.
DELETE	`/applications/{applicationId}/definedroles/{roleId}`	Delete a role defined by an application.
POST	`/applications/{applicationId}/secret`	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.
PUT	`/applications/{applicationId}/applications/{provisionedApplicationId}/roles/{roleId}`	Grant application a role on another application.
DELETE	`/applications/{applicationId}/applications/{provisionedApplicationId}/roles/{roleId}`	Revoke an application's role on another application..
GET	`/applications/{applicationId}/roles?applicationId=`*`string`*	Get the applications that have been provisioned for this application.

4.3.1. Get Applications

Verb	URI	Description
GET	`/applications?name=`*`string`*	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 4.51. Get Applications Response: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0">
 <application
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347"/>
 <application
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0002"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="632h389cv902bde"/>
</application>
```

Example 4.52. Get Applications Response: JSON

```
{
  "application": [
    {
      "clientId": "ab4820dhcb39347",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
      "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    },
    {
      "clientId": "632h389cv902bde",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
      "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0002",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    }
  ]
}
```

Table 4.27. Get Applications Request Parameters

Name	Style	Description
`Authorization`	header	You need a valid token for access. String. Required.
`name`	query	Allows you search for an application by name String. Optional.

4.3.2. 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), itemNotFound (404)

Example 4.53. 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" />
```

Example 4.54. Add Application Request: JSON

```
{
  "customerId": "RCN-000-000-000",
  "name": "Test Application2"
}
```

Example 4.55. Add 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"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347">
    <credentials clientSecret="3af738fbeiwu23" />
</application>
```

Example 4.56. Add Application Response: JSON

```
{
  "credentials": {
    "clientSecret": "3af738fbeiwu23"
  },
  "clientId": "ab4820dhcb39347",
  "customerId": "RCN-000-000-000",
  "name": "Test Application2",
  "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001",
  "status": "ACTIVE",
  "locked": false,
  "softDeleted": false
}
```

Table 4.28. Add an Application Request Parameters

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

4.3.3. 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 4.57. Get 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" />
```

Example 4.58. Get Application Response: JSON

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

Table 4.29. Get an Application Request Parameters

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

This operation does not require a request body.

4.3.4. 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 4.59. 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 4.60. Update Application Request: JSON

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

Example 4.61. 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 4.62. 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 4.30. Update an Application Request Parameters

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

Name	Style	Description
		String. Required.
`applicationId`	template	String.

4.3.5. 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 4.31. Delete an Application Request Parameters

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

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

4.3.6. Soft Delete an Application

Verb	URI	Description
PUT	`/applications/{applicationId}/softdeleted`	Soft deletes an application.

Normal Response Code(s): 204

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

Example 4.63. Soft Delete Application Request: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0"
    softDeleted="true" />
```

Example 4.64. Soft Delete Application Request: JSON

```
{
  "softDeleted": "true"
}
```

Table 4.32. Soft Delete an Application Request Parameters

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

This operation does not return a response body.

4.3.7. Get Roles defined by an Application

Verb	URI	Description
GET	`/applications/{applicationId}/definedroles`	Get roles defined by an application.

Normal Response Code(s): 200

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

Example 4.65. Defined Roles Response: XML

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

<roles xmlns="http://idm.api.rackspace.com/v1.0">
    <role name="network_admin" description="Role for networking related
 functionality, like routing servers, etc"/>
    <role name="technical_reviewer" description="Reviews technical related
 stuff" />
    <role name="admin"  />
</roles>
```

Example 4.66. Defined Roles Response: JSON

```
{
  "role": [
    {
      "name": "network_admin"
      "description": "Role for networking related functionality, like routing
 servers, etc"
    },
    {
      "name": "technical_reviewer"
      "description": "Reviews technical related stuff""
    }
    {
      "name": "admin"
      "description": ""
    }
  ]
}
```

Table 4.33. Get Roles defined by an Application Request Parameters

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

This operation does not require a request body.

4.3.8. Define a Role for an Application

Verb	URI	Description
PUT	`/applications/{applicationId}/definedroles/{roleId}`	Add a role defined by an application.

Normal Response Code(s): 204

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

Example 4.67. Define Role Request: XML

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

<role name="admin" description="Can admin the heck out of anything" />
```

Example 4.68. Define Role Request: JSON

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

Table 4.34. Define a Role for an Application Request Parameters

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

This operation does not return a response body.

4.3.9. Delete a Defined Role for an Application

Verb	URI	Description
DELETE	`/applications/{applicationId}/definedroles/{roleId}`	Delete a role defined by an application.

Normal Response Code(s): 204

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

Table 4.35. Delete a Defined Role for an Application Request Parameters

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

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

4.3.10. Reset Application Secret

Verb	URI	Description
POST	`/applications/{applicationId}/secret`	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 4.69. Reset Application Secret Response: XML

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

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

Example 4.70. Reset Application Secret Response: JSON

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

Table 4.36. Reset Application Secret Request Parameters

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

This operation does not require a request body.

4.3.11. 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 4.71. Get Provisioned Applications Response: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0">
 <application
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347"/>
 <application
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0002"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="632h389cv902bde"/>
</application>
```

Example 4.72. Get Provisioned Applications Response: JSON

```
{
  "application": [
    {
      "clientId": "ab4820dhcb39347",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
      "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    },
    {
      "clientId": "632h389cv902bde",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
      "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0002",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    }
```

```
   ]
}
```

Table 4.37. Get the Applications that an Application has Provisioned Request Parameters

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

This operation does not require a request body.

4.3.12. 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)

Example 4.73. Provision Application Request: XML

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

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

Example 4.74. Provision Application Request: JSON

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

Table 4.38. Provision an Application for an Application Request Parameters

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

This operation does not return a response body.

4.3.13. 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 4.39. Remove Provisioned Application from an Application Request Parameters

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

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

4.3.14. Grant Application Role on another Application

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

Normal Response Code(s): 204

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

Example 4.75. Grant Application Role Request: XML

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

<role name="admin" tenantId="cf_78373" />
```

Example 4.76. Grant Application Role Request: JSON

```
{
      "name": "admin",
      "tenantId": "cf_78373"
}
```

Table 4.40. Grant Application Role on another Application Request Parameters

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

This operation does not return a response body.

4.3.15. Revoke an Application Role on another Application

Verb	URI	Description
DELETE	`/applications/{applicationId}/applications/{provisionedApplicationId}/roles/{roleId}`	Revoke an application's role on another application..

Normal Response Code(s): 204

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

Table 4.41. Revoke an Application Role on another Application Request Parameters

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

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

4.3.16. Get Application Roles

Verb	URI	Description
GET	`/applications/{applicationId}/roles?applicationId=`*`string`*	Get the 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 4.77. Get Application 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"/>
    <role name="technical_reviewer" tenantId="57884" applicationId="swift"/>
    <role name="admin" tenantId="cf_89983" applicationId="billing"/>
</roles>
```

Example 4.78. Get Application Roles Response: JSON

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

Table 4.42. Get Application Roles Request Parameters

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

4.4. Customer Identity Profile Operations

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

Verb	URI	Description
GET	`/customeridentityprofiles/{customerId}`	Gets a customer's identity profile.
DELETE	`/customeridentityprofiles/{customerId}`	Delete a customer's identity profile.
PUT	`/customeridentityprofiles/{customerId}/locked`	Lock or unlock a customer's identity profile.
PUT	`/customeridentityprofiles/{customerId}/passwordrotationpolicy`	Sets the password rotation policy for a customer.
GET	`/customeridentityprofiles/{customerId}/users`	Gets a list of users that this customer owns.
GET	`/customeridentityprofiles/{customerId}/applications`	Gets a list of applications that this customer owns.

4.4.1. 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 4.79. 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"
    locked="false"
    iname="@Rackspace*Customer"
    inum="@FFFF.FFFF.FFFF.FFFF!0ABE.34EC"
    customerId="RCN-123-549-034" />
```

Example 4.80. Get Customer Identity Profile Response: JSON

```
{
  "customerId": "RCN-123-549-034",
  "inum": "@FFFF.FFFF.FFFF.FFFF!0ABE.34EC",
  "iname": "@Rackspace*Customer",
  "locked": false,
  "softDeleted": false
}
```

Table 4.43. Get a Customer's Identity Profile Request Parameters

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

This operation does not require a request body.

4.4.2. 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 4.44. Delete a Customer's Identity Profile Request Parameters

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

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

4.4.3. Lock or Unlock Customer's Identity Profile

Verb	URI	Description
PUT	`/customeridentityprofiles/{customerId}/locked`	Lock or unlock 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 4.81. Customer Identity Profile Lock Request: XML

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

<customeridentityprofile xmlns="http://idm.api.rackspace.com/v1.0"
    locked="true"  />
```

Example 4.82. Customer Identity Profile Lock Request: JSON

```
{
    "locked": true
}
```

Table 4.45. Lock or Unlock Customer's Identity Profile Request Parameters

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

This operation does not return a response body.

4.4.4. 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 4.83. 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 4.84. Password Rotation Policy Request: JSON

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

Table 4.46. Set Password Rotation Policy Request Parameters

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

This operation does not return a response body.

4.4.5. Get Users

Verb	URI	Description
GET	/customeridentityprofiles/{customerId}/users	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 4.85. 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 4.86. 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 4.47. Get Users Request Parameters

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

This operation does not require a request body.

4.4.6. Get Applications

Verb	URI	Description
GET	`/customeridentityprofiles/{customerId}/applications`	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 4.87. Get Applications Response: XML

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

<application xmlns="http://idm.api.rackspace.com/v1.0">
 <application
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="ab4820dhcb39347"/>
 <application
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0002"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="632h389cv902bde"/>
</application>
```

Example 4.88. Get Applications Response: JSON

```
{
  "application": [
    {
      "clientId": "ab4820dhcb39347",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
      "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    },
    {
      "clientId": "632h389cv902bde",
      "customerId": "RCN-000-000-000",
      "name": "Test Application2",
      "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0002",
      "status": "ACTIVE",
      "locked": false,
      "softDeleted": false
    }
  ]
}
```

Table 4.48. Get Applications Request Parameters

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

This operation does not require a request body.

4.5. 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.

4.5.1. 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 4.89. 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 4.90. 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 4.49. Get Password Rules Request Parameters

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

This operation does not require a request body.

4.5.2. 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 4.91. Password Validation Request: XML

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

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

Example 4.92. Password Validation Request: JSON

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

Example 4.93. 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 4.94. 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 4.50. Password Validation Check Request Parameters

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