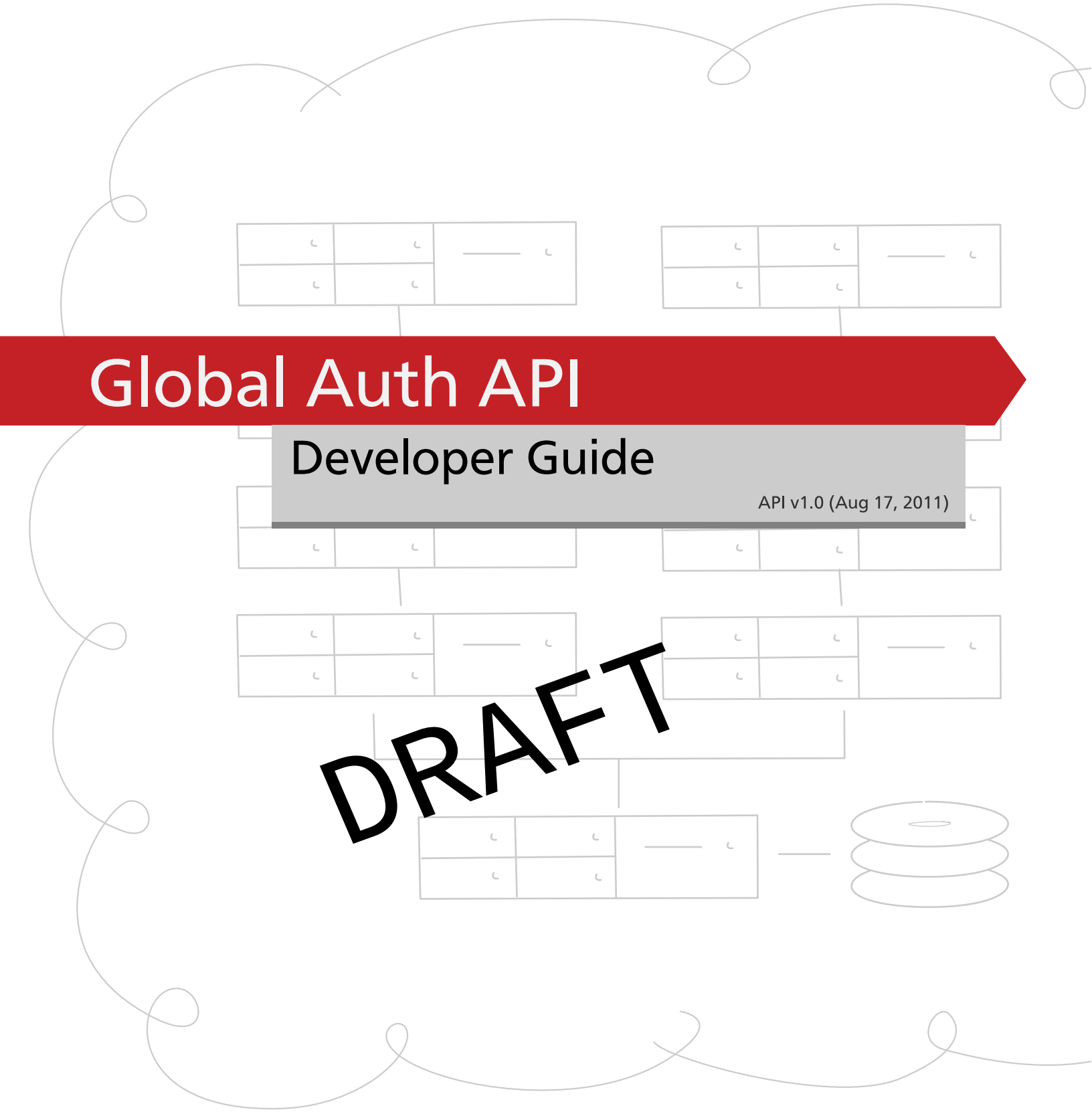
Global Auth API

Developer Guide

API v1.0 (Aug 17, 2011)



rackspace® HOSTING

docs.rackspace.com/api

Global Auth API Developer Guide

API v1.0 (2011-08-17)

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. Overview

The Global Auth API allows Rackspace clients to obtain tokens that can be used to access resources in Rackspace. It also allows clients manage identities. This document is intended for:

Service Developers	Service developers are interested in writing clients for Global Auth API service.

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

2. Concepts

The Global Auth system has several key concepts that are important to understand:

2.1. 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.

2.2. User

This is a type of client associated with a person that has access to Rackspace Applications.

2.3. 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.

2.4. 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.

2.5. Tenant

This is a container that contains a collection of a customers products and services that Rackspace provides.

2.6. 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.

3. General API Information

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.

3.1. 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 3.1. Response Types

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

Example 3.1. XML Request with Headers

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

3.2. API Version

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 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.

3.2.1. Service

Verb	URI	Description
GET	https://{Hostname TBD}/	Retrieve the service profile.

Normal Response Code(s):200

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

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

This operation does not require a request body.

Example 3.3. XML Service Profile Response

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

    <link rel="self" href="https://idm.api.rackspace.com" />
    <link rel="describedby" type="text/html" href="http://serviceregistry.
rackspace.com/services/idm" />

    <short-description>Provides auth related capabilities.</short-description>

    <detailed-description>
 The Global Auth API allows Rackspace clients to obtain tokens that can be
 used to access resources in Rackspace.
 It also allows clients manage identities.
    </detailed-description>
```

```
    <versions>
      <version id="v1.0" status="CURRENT" updated="2011-03-09T16:42:17Z">

        <media-types>
          <media-type base="application/xml" type="application/vnd.rackspace.
idm-v1+xml">
            <link rel="describedby" type="application/xml" href="https://idm.
api.rackspace.com/idm-v1.0.xsd" />
          </media-type>
        </media-types>
        <link rel="self" href="https://idm.api.rackspace.com/v1.0/"/>
        <link rel="describedby" type="application/vnd.sun.wadl
+xml" href="https://idm.api.rackspace.com/v1.0" />
      </version>
    </versions>

</service-profile>
```

3.2.2. Get Version Wadl

Verb	URI	Description
GET	https://{Hostname TBD}/v1.0/	Retrieve Wadl for specific version

Normal Response Code(s):200

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

You can obtain the wadl for a specific version of the service by performing a **GET** on the base version URL (e.g. https://idm.api.rackspace.com/v1.0).

This operation does not require a request body.

3.3. 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 3.4. XML Fault Response

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

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

Example 3.5. 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 (e.g. idmFault) may change depending on the type of error. The following is an example of an itemNotFound error.

Example 3.6. XML Not Found Fault

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

<itemNotFound xmlns="http://idm.api.rackspace.com/v1.0"
    code="404">
    <message>Item not found.</message>
    <details>Error Details...</details>
</itemNotFound>
```

Example 3.7. JSON Not Found Fault

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

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

Table 3.2. Fault Types

Fault Element	Associated Error Code	Expected in All Requests
idmFault	500, 400	✓
serviceUnavailable	503	✓
unauthorized	401	✓
badRequest	400	✓
passwordValidation	400	
userDisabled	403	
forbidden	403	
itemNotFound	404	
clientConflict	409	
customerConflict	409	
emailConflict	409	
usernameConflict	409	

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

3.4. 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 3.3. 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 section below for Authentication).

Example 3.8. XML Request with Headers

```
POST /v1.0/token HTTP/1.1
Host: {host name TBD}
Content-Type: application/json
Accept: application/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 3.9. 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>
```

3.5. 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 3.10. Authorization Header Request Format

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

4. Service API (Client Operations)

4.1. Overview

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

4.2. Service API

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

Note

The following table of calls are the service apis.

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

4.3. Authenticate

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

Normal Response Code(s):200, 203

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

Example 4.1. XML User Auth Request

```
<?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. JSON User Auth Request

```
{
  "grant_type": "PASSWORD",
  "client_id": "8972348923400fdshasdf",
  "client_secret": "0923899flewriudsb",
  "username": "testuser",
```

```
  "password": "P@ssword1"
}
```

Example 4.3. XML User Auth Response

```
<?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. JSON User Auth Response

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

Example 4.5. XML Application Auth Request

```
 <?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 4.6. JSON Application Auth Request

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

Example 4.7. XML Application Auth Response

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

<auth xmlns="http://idm.api.rackspace.com/v1.0">
 <access_token
    expires_in="3600"
    id="ab48a9efdfedb23ty3494" />
 <client
        customerId="RACKSPACE"
        clientId="ab4820dhcb39347" />
</auth>
```

Example 4.8. JSON Application Auth Response

```
{
  "accessToken": {
    "id": "ab48a9efdfedb23ty3494",
    "expiresIn": 3600
  },
  "client": {
    "clientId": "ab4820dhcb39347",
    "customerId": "RACKSPACE",
   }
}
```

5. Admin API (Service Developer Operations)

5.1. Overview

The operations described in this chapter allow service developers to get and validate access tokens, manage users and manage applications.

5.2. Admin API

Note

The following are tables of the admin api.

5.2.1. Tokens

Verb	URI	Description
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 has access to an application.
GET	/tokens/*tokenId*/applications/*applicationId*/roles/*roleId*	Check if token has role on application.

5.2.2. Users

Verb	URI	Description
POST	/users	Adds a new user. If the customer specified does not exist, adds this first user as the admin user.
GET	/users/*username*	Get a user.
DELETE	/users/*username*	Delete a user.
PUT	/users/*username*	Update a user.
PUT	/users/*username*/softdeleted	Soft deletes a user.
PUT	/users/*username*/lock	Set a user's lock.
PUT	/users/*username*/status	Set a user's status to ACTIVE or INACTIVE.
PUT	/users/*username*/secret	Set a user's secret question and answer.
GET	/users/*username*/password	Get a user's password.
PUT	/users/*username*/password	Set a user's password.
POST	/users/*username*/password	Reset a user's password.
GET	/users/*username*/password/recoverytoken	Get a token that can be used to reset a user's password.
GET	/users/*username*/delegatedrefreshtokens	Get a list of a user's delegated refresh tokens.

Verb	URI	Description
GET	/users/*username*/delegatedrefreshtokens/*tokendId*	Get a user's delegated refresh token.
DELETE	/users/*username*/delegatedrefreshtokens/*tokendId*	Delete a user's delegated refresh token.
GET	/users/*username*/applications	Get the applications that have been provisioned for a user.
PUT	/users/*username*/applications/*applicationId*	Provision an application for a user.
DELETE	/users/*username*/applications/*applicationId*	Removes a provisioned application from a user.
GET	/users/*username*/roles?applicationId=*applicationId*	Get all the roles that a user has. Optionally filter by applicationId.
PUT	/users/*username*/applications/*applicationId*/roles/*roleId*	Grant user a role on an application.
DELETE	/users/*username*/applications/*applicationId*/roles/*roleId*	Revoke a user's role on an application.

5.2.3. Applications

Verb	URI	Description
POST	/applications	Add an application.
GET	/applications/*applicationId*	Get an application.
PUT	/applications/*applicationId*	Update an application.
DELETE	/applications/*applicationId*	Delete an application.
DELETE	/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/*applicationId*	Provision an application for this application.
DELETE	/applications/*applicationId*/applications/*applicationId*	Removes a provisioned application from this application.
GET	/applications/*applicationId*/roles?applicationId=*applicationId*	Get all the roles that an application has. Optionally filter by applicationId.
PUT	/applications/*applicationId*/applications/*applicationId*/roles/*roleId*	Grant application a role on a provisioned application.
DELETE	/applications/*applicationId*/applications/*applicationId*/roles/*roleId*	Revoke an application's role on a provisioned application.

5.2.4. Racker

Verb	URI	Description
GET	/rackers/*rackerId*/roles	Get a racker's roles.

5.2.5. Identity Profile

Verb	URI	Description
GET	/identityprofiles/*customerId*	Gets a customer's identity profile.

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

5.2.6. Password

Verb	URI	Description
GET	/passwordrules	Get password rules.
POST	/passwordrules/validation	Check password validation for password.

5.3. Token Operations

5.3.1. Authenticate

All client operations are supported in the admin API

5.3.2. Validate Token

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

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

Normal Response Code(s):200, 203

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

This operation does not require a request body.

Example 5.1. XML Validate Token Response

```
<?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 5.2. JSON Validate Token Response

```
{
    "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"}]

}
```

5.3.3. Revoke Token

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
DELETE	/token/*tokenId*	Revoke token.

Normal Response Code(s):204

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

This operation does not require a request body.

5.3.4. Check if Token has access to an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

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

Normal Response Code(s):204 404

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

This operation does not require a request body.

5.3.5. Check if Token has Role on an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

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

Normal Response Code(s):204 404

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

This operation does not require a request body.

5.4. User Operations

5.4.1. Add a User

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

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

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".

Normal Response Code(s):201

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

Example 5.3. XML Add User Request

```
<?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 5.4. JSON Add User Request

```
{
  "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 5.5. XML Add User Response

```
<?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 5.6. JSON Add User Response


```
{
  "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
}
```


5.4.2. Get a User

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/users/*username*	Get a user.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.7. XML Get User Response


```
            <?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 5.8. JSON Get User Response

```
          {
  "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
}
```

5.4.3. Delete User

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
DELETE	/users/*username*	Delete a user.

Normal Response Code(s):204

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

This operation does not require a request body.

5.4.4. Update a User

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/users/*username*	Update a user.

Normal Response Code(s):200, 203

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

Example 5.9. XML Update User Request

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

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

Example 5.10. JSON Update User Request

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

Example 5.11. XML Update User Response

```
<?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 5.12. JSON Update User Response

```
{
  "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
}
```

5.4.5. Soft Delete a User

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/users/*username*/softdeleted	Soft deletes a user.

Normal Response Code(s):200

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

Example 5.13. XML User Soft Delete Request

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

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

Example 5.14. JSON User Soft Delete Request

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

Example 5.15. XML User Soft Delete Response

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

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

Example 5.16. JSON User Soft Delete Response

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

5.4.6. Set User Lock

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/users/*username*/lock	Set a user's lock

Normal Response Code(s):200

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

Example 5.17. XML Set User Lock Request

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

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

Example 5.18. JSON Set User Lock Request

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

Example 5.19. XML Set User Lock Response

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

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

Example 5.20. JSON Set User Lock Response

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

5.4.7. Set User Status

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/users/*username*/status	Set a user's status to ACTIVE or INACTIVE

Normal Response Code(s):200

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

Example 5.21. XML Set User Status Request

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

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

Example 5.22. JSON Set User Status Request

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

Example 5.23. XML Set User Status Response

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

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

Example 5.24. JSON Set User Status Response

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

5.4.8. Set User Secret

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/users/*username*/secret	Sets a user's secret question and answer.

Normal Response Code(s):200

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

Example 5.25. XML Set User Secret Request

```
<?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 5.26. JSON Set User Secret Request

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


Example 5.27. XML Set User Secret Response


```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 5.28. JSON Set User Secret Response


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


5.4.9. Get User Password

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/users/*username*/password	Get a user's password.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.29. XML Get User Password Response


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

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


Example 5.30. JSON Get User Passwrord Response


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

5.4.10. Set User Password

This operation requires an authorization header. See Section 3.5, "Authorization Header" for details.

Verb	URI	Description
PUT	/users/*username*/password	Sets a user's password..

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.31. XML Set User Password Request

```
<?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 5.32. JSON Set User Password Request

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

Example 5.33. XML User Password Response

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

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

Example 5.34. JSON User Password Response

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

5.4.11. Reset User Password

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
POST	/users/*username*/password	Reset a user's password.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.35. XML Reset User Password Response

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

<userPassword xmlns="http://idm.api.rackspace.com/v1.0"
    password="7ud$dnF" />
```

Example 5.36. JSON Reset User Password Response

```
{
  "password": "7ud$dnF"
}
```

5.4.12. Get Password Recovery Token

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
POST	/users/*username*/password/recoverytoken	Get a token that can be used to reset a user's password.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.37. XML Password Recovery Token Response

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

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

Example 5.38. JSON Password Recovery Token Response

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

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

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/users/*username*/delegatedrefreshtokens	Get a list of a user's delegated refresh tokens.

Normal Response Code(s):200

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

Example 5.39. XML Delegated Tokens Response

```
<?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 5.40. JSON Delegated Tokens Response

```
{ "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"
    }]
}
```


5.4.14. Get a Delegated Refresh Token for a User

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/users/*username*/delegatedrefreshtokens/*tokendId*	Get a user's delegated refresh token

Normal Response Code(s):200

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

Example 5.41. XML Delegated Token Response


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

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


Example 5.42. JSON Delegated Token Response


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


5.4.15. Delete a Delegated Refresh Token for a User

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
DELETE	/users/*username*/delegatedrefreshtokens/*tokendId*	Deletes a user's delegated refresh tokens

Normal Response Code(s):204

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

This operation does not require a request body.

5.4.16. Get the Applications that a User has Provisioned

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/users/*username*/applications	Get the applications that have been provisioned for a user.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.43. XML Get User Applications Response

```
<?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 5.44. JSON Get User Applications Response

```
{
  "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
    }
  ]
}
```

5.4.17. Provision an Application for a User

This operation requires an authorization header. See Section 3.5, "Authorization Header" for details.

Verb	URI	Description
PUT	/users/*username*/applications/*applicationId*	Provision an application for a user.

Normal Response Code(s):200

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

Example 5.45. XML Provision User Application Request

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

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

Example 5.46. JSON Provision User Application Request

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

5.4.18. Remove Provisioned Application from User

This operation requires an authorization header. See Section 3.5, "Authorization Header" for details.

Verb	URI	Description
DELETE	/users/*username*/applications/*applicationId*	Removes a provisioned application from a user.

Normal Response Code(s):204

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

This operation does not require a request body.

5.4.19. Get User Roles

This operation requires an authorization header. See Section 3.5, "Authorization Header" for details.

Verb	URI	Description
GET	/users/*username*/roles?applicationId=*applicationId*	Get all the roles that a user has. Optionally filter by applicationId.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.47. XML Get User Application Roles Response

```
              <?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 5.48. JSON Get User Application Roles Response

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

5.4.20. Grant User Role on Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/users/*username*/applications/*applicationId*/roles/*roleId*	Grant user a role on the application.

Normal Response Code(s):200

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

Example 5.49. XML Grant User Role Request

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

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

Example 5.50. JSON Grant User Role Request

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

Example 5.51. XML Grant User Role Response

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

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

Example 5.52. JSON Grant User Role Response

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

5.4.21. Revoke a User Role on Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
DELETE	/users/*username*/applications/*applicationId*/roles/*roleId*	Revoke a user's role on the application.

Normal Response Code(s):204

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

This operation does not require a request body.

5.5. Application Operations

5.5.1. Add an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
POST	/applications	Add an application.

Normal Response Code(s):201

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

Example 5.53. XML Add Application Request

```
<?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 5.54. JSON Add Application Request

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

Example 5.55. XML Add Application Response

```
<?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 5.56. JSON Add Application Response


```
{
  "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
}
```


5.5.2. Get an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/applications/*applicationId*	Get an application.

Normal Response Code(s):200

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

Example 5.57. XML Get Application Response


```
<?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 5.58. JSON Get Application Response

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


5.5.3. Update an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/applications/*applicationId*	Update an application.

Normal Response Code(s):200

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

Example 5.59. XML Update Application Request


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

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


Example 5.60. JSON Update Application Request


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


Example 5.61. XML Update Application Response


```
<?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 5.62. JSON Update Application Response


```
{
  "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
}
```


5.5.4. Delete an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
DELETE	/applications/*applicationId*	Delete an application.

Normal Response Code(s):204

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

This operation does not require a request body.

5.5.5. Soft Delete an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/applications/*applicationId*/softdeleted	Soft deletes an application.

Normal Response Code(s):200

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

Example 5.63. XML Soft Delete Application Request

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

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

Example 5.64. JSON Soft Delete Application Request

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

Example 5.65. XML Soft Delete Application Response

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

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

Example 5.66. JSON Soft Delete Application Response

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

5.5.6. Get all Roles for an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/applications/*applicationId*/roles	Get roles defined by an application.

Normal Response Code(s):200 404

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

This operation does not require a request body.

Example 5.67. XML Roles Defined by Application Response

```
              <?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 5.68. JSON Roles Defined by Application Response


```
            {
  "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": ""
    }
  ]
}
```


5.5.7. Define a Role for an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

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

Normal Response Code(s):200

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

Example 5.69. XML Define Role Request


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

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


Example 5.70. JSON Define Role Request


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


Example 5.71. XML Define Role Response


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

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

Example 5.72. JSON Define Role Response

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

5.5.8. Delete a Defined Role for an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

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

Normal Response Code(s):204

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

This operation does not require a request body.

5.5.9. Reset Application Secret

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
POST	/applications/*applicationId*/secret	Reset application secret.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.73. XML Reset Application Secret Response

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

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

Example 5.74. JSON Reset Application Secret Response

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

5.5.10. Get the Applications that an Application has Provisioned

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

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

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.75. XML Get Applications Response

```
<?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 5.76. JSON Get Applications Response

```
{
  "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
    }
  ]
}
```

5.5.11. Provision an Application for an Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/applications/*applicationId*/ applications/*applicationId*	Provision an application for this application.

Normal Response Code(s):204

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

This operation does not require a request body.

Example 5.77. XML Provision Application Request

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

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

Example 5.78. JSON Provision Application Request

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

5.5.12. Remove Provisioned Application from Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
DELETE	/applications/*applicationId*/ applications/*applicationId*	Removes a provisioned application from this application

Normal Response Code(s):204

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

This operation does not require a request body.

5.5.13. Get Application Roles

This operation requires an authorization header. See Section 3.5, "Authorization Header" for details.

Verb	URI	Description
GET	/applications/*applicationId*/roles? applicationId=*applicationId*	Get all the roles that an application has. Optionally filter by applicationId.

Normal Response Code(s):200 404

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

This operation does not require a request body.

Example 5.79. XML Get Application Roles Response

```
              <?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 5.80. JSON Get Application Roles Response

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

```
    {
      "name": "admin",
      "tenantId": "cf_89983",
      "applicationId": "billing"
    }
  ]
}
```


5.5.14. Grant an Application a Role on another Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
POST	/applications/*applicationId*/ applications/*applicationId*/roles/*roleId*	Grant application a role on the provisioned application.

Normal Response Code(s):200

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

Example 5.81. XML Grant Application Role Request


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

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


Example 5.82. JSON Grant Application Role Request


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


Example 5.83. XML Grant Application Role Response


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

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


Example 5.84. JSON Grant Application Role Response


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

5.5.15. Revoke an Application Role on another Application

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
DELETE	/applications/*applicationId*/ applications/*applicationId*/roles/*roleId*	Revoke an application's role on a provisioned application.

Normal Response Code(s):204

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

This operation does not require a request body.

5.6. Identity Profile Operations

5.6.1. Get a Customer's Identity Profile

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/identityprofiles/*customerId*	Gets a customer's identity profile.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.85. XML Identity Profile Response

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

<identityprofile 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 5.86. JSON Identity Profile Response

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

5.6.2. Delete a Customer's Identity Profile

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
DELETE	/identityprofiles/*customerId*	Delete a customer's identity profile.

Normal Response Code(s):204

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

This operation does not require a request body.

5.6.3. Lock or unlock Customer's Identity Profile

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/identityprofiles/*customerId*/locked	Lock or unlock a customer's identity profile.

Normal Response Code(s):200

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

Example 5.87. XML Identity Profile Lock Request

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

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

Example 5.88. JSON Identity Profile Lock Request

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

Example 5.89. XML Identity Profile Lock Response

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

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

Example 5.90. JSON Identity Profile Lock Response

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

5.6.4. Set Customer's Password Rotation Policy

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
PUT	/identityprofiles/*customerId*/passwordrotationpolicy	Sets the password rotation policy for a customer.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.91. XML Password Rotation Policy Request

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

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

Example 5.92. JSON Password Rotation Policy Request

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

Example 5.93. XML Password Rotation Policy Response

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

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


Example 5.94. JSON Password Rotation Policy Response


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


5.6.5. Get User List

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/identityprofiles/*customerId*/users	Gets a list of users that this customer owns.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.95. XML Get User List Response


```
<?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 5.96. JSON Get User List Response

```
{"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
}
```

5.6.6. Get Application List

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/identityprofiles/*customerId*/applications	Gets a list of applications that this customer owns.

Normal Response Code(s):200

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

This operation does not require a request body.

Example 5.97. XML Get User List Response

```
<?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 5.98. JSON Get User List Response

```
{
  "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
    }
```

```
  ]
}
```

5.7. Password Operations

5.7.1. Get Password Rules

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/passwordrules	Get Password Rules

Normal Response Code(s):200

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

Example 5.99. XML Get Password Rules Response

```
<?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 5.100. JSON Get Password Rules Response

```
{
  "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."
    }
  ]
```

```
}
```

5.7.2. Password validation check

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
POST	/passwordrules/validation	Check Password Validation for *password*

Normal Response Code(s):200

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

Example 5.101. XML Password Validation Request

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

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

Example 5.102. JSON Password Validation Request

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

Example 5.103. XML Password Validation Response

```
<?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 5.104. JSON Password Validation Response

```
{"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
}
```

5.8. Racker Operations

5.8.1. Get the Rackspace Roles for a Racker

This operation requires an authorization header. See Section 3.5, “Authorization Header” for details.

Verb	URI	Description
GET	/rackers/*rackerId*/roles	Get a racker's roles

Normal Response Code(s):200

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

Example 5.105. XML Racker Roles Response

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

<roles xmlns="http://idm.api.rackspace.com/v1.0">
    <role name="dl_foundation" />
    <role name="dl_cloud" />
</roles>
```

Example 5.106. JSON Racker Roles Response

```
{
  "role": [
    {
      "name": "dl_foundation"
    },
    {
      "name": "dl_cloud"
    }
```

```
  ]
}
```