IdM Developer Guide

12/1/10

API v1.0.2010.10

This document is intended for software developers interested in developing applications which utilizes Customer Identity Management System as the authentication engine. It includes details on how to integrate with IdM.

Table of Contents

List of Tables

List of Examples

Overview

The IdM Service allows Rackspace Applications to obtain tokens that can be used to access resources in the Rackspace Cloud. This document is intended for:

Service Developers	Service developers are interested in writing client for Rackspace IdM service.

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

Concepts

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

Token

Small bits of text that contain the security information for a login session and identifies the user, the user's groups, and the user's privileges. Tokens may be revoked at anytime and are valid for a finite duration.

Customer

Organizations that are customers of Rackspace

User

Individuals at a customer that have a login

Client

Rackspace applications (e.g.: Control Panel, Billing, etc.) or Customer third party applications. IdM will intially setup all Rackspace applications

General API Information

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

Request/Response Types

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

Table 1. Response Types

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

Example 1. JSON Request with Headers

```
POST /v1.0/token HTTP/1.1
Host: idm.api.rackspace.com
Content-Type: application/json
Accept: application/xml
```

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

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


Example 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://docs.rackspacecloud.com/idm/api/v1.0">
 <access_token
    expires_in="3600"
    id="ab48a9efdfedb23ty3494" />
 <refresh_token
 id="8792gdfskjbadf98y234r" />
 <user
     xmlns="http://docs.rackspacecloud.com/idm/api/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">
     <roles>
         <role type="RackspaceDefined"
               name="Admin" />
     </roles>
 </user>
 <client xmlns="http://docs.rackspacecloud.com/idm/api/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" />
</auth>
```

Content Compression

Request and response body data my be encoded with gzip compression in order to accelerate interactive performance of API calls and responses. This is controlled using the

`Accept-Encoding` header on the request from the client and indicated by the `Content-Encoding` header in the server response. Unless the header is explicitly set, encoding defaults to disabled.

Table 2. Compression Headers

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

API Version

The IdM 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.

List Versions

Verb	URI	Description
GET	https://idm.api.rackspace.com/	Retrieve a list of IdM API versions.

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. XML Versions Response

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

<versions xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
 <version
    id="v0.9"
    status="DEPRECATED"
    href="/v0.9" />
 <version
    id="v1.0"
    status="CURRENT"
    href="/v1.0" />
 <version
    id="v1.1"
    status="BETA"
    href="/v1.1" />
</versions>
```

Get Version Details

Verb	URI	Description
GET	https://idm.api.rackspace.com/v1.0/	Retrieve API Version details

Normal Response Code(s):200

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

You can also obtain additional information about a specific version by performing a **GET** on the base version URL (e.g. https://idm.api.rackspace.com/v1.0). Version request URLs should always end with a trailing slash (/). If the slash is omitted, the server may respond with a 302 redirection request. Format extensions may be placed after the slash (e.g. https://idm.api.rackspace.com/v1.0/.xml).

This operation does not require a request body.

Example 4. XML Version Response

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

<version xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
 wadl="http://docs.rackspacecloud.com/idm/api/v1.0/
application.wadl"
 docURL="http://docs.rackspacecloud.com/idm/api/v1.0/idm-
devguide.pdf"
 status="CURRENT"
 id="v1.0" />
```

The detailed version response contains a pointer to both the human readable and a machine processable description of the API service. The machine processable description is written in the Web Application Description Language (WADL).

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 5. XML Fault Response

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

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

Example 6. 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 7. XML Not Found Fault

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

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

Example 8. 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. 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.

Service Developer Operations

Overview

The operations described in this chapter allow service developers to get and validate access tokens, manage users, manage clients, and password related operations (get password rules, valid password).

Token Operations

Authenticate

Verb	URI	Description
POST	/token	Authenticate to generate a token and a service catalog.

Normal Response Code(s):200, 203

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

Example 9. XML Auth Request

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

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

Example 10. JSON Auth Request

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


Example 11. XML Auth Response

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

<auth xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
 <access_token
    expires_in="3600"
    id="ab48a9efdfedb23ty3494" />
 <refresh_token
 id="8792gdfskjbadf98y234r" />
 <user
     xmlns="http://docs.rackspacecloud.com/idm/api/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">
     <roles>
         <role type="RackspaceDefined"
               name="Admin" />
     </roles>
 </user>
 <client xmlns="http://docs.rackspacecloud.com/idm/api/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" />
</auth>
```


Example 12. JSON Auth Response

```
{
  "accessToken": {
    "id": "ab48a9efdfedb23ty3494",
    "expiresIn": 3600
  },
  "refreshToken": {
    "id": "8792gdfskjbadf98y234r"
  },
  "user" : {
    "roles": {
        "role": [
        {
          "name": "Admin",
          "type": "RackspaceDefined"
        }
      ]},
    "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
  },
  "client": {
    "clientId": "ab4820dhcb39347",
    "customerId": "RCN-000-000-000",
    "name": "Test Application2",
    "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001",
    "iname": "@Rackspace*Rackspace*ControlPanel",
    "status": "ACTIVE",
    "locked": false,
    "softDeleted": false
   }
}
```


Validate Token

Verb	URI	Description
GET	/token/*tokenId*? belongsTo=Username&clientId=ClientId	Check that a token is valid and that it belongs to a particular user and return the permissions relevant to a particular client.

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 13. XML Validate Token Response

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

<auth xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
 <access_token
    expires_in="3600"
    id="ab48a9efdfedb23ty3494" />
 <user
     xmlns="http://docs.rackspacecloud.com/idm/api/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">
     <roles>
         <role type="RackspaceDefined" name="Admin" />
     </roles>
 </user>
 <client xmlns="http://docs.rackspacecloud.com/idm/api/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">
 </client>
 <permissions>
        <permission id="CloudServers-1"
            clientId="CS"
            customerId="RCN-000-000-000" />
    </permissions>
</auth>
```


Example 14. JSON Validate Token Response

```
{
    "access_token": {
        "id": "ab48a9efdfedb23ty3494",
        "expires_in": "3600"
    },
    "user" : {
        "roles": { "role": [
              {
                "name": "Admin",
                "type": "RackspaceDefined"
              }]},
          "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
     },
     "client": {
          "clientId": "ab4820dhcb39347",
          "customerId": "RCN-000-000-000",
          "name": "Test Application2",
          "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001",
          "status": "ACTIVE",
          "locked": false,
          "softDeleted": false
     },
     "permissions": { "permission": [
          {
            "id": "CloudServers-1",
            "clientId": "CS",
            "customerId": "RCN-000-000-000"
          }
        ]
     }
}
```


Revoke Token

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

Get Token Permissions

Verb	URI	Description
GET	/token/*tokenId*/permissions	Gets a list of permissions for the token.

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 15. XML Permission List Response

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

<permissions xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
    <granted>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="addCustomer"/>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="getCustomer"/>
    </granted>
    <defined>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="addCustomer"/>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="getCustomer"/>
    </defined>
</permissions>
```


Example 16. JSON Permission List Response

```
{
  "granted": {
    "permission": [
      {
        "permissionId": "addCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      },
      {
        "permissionId": "getCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      }
    ]
  },
  "defined": {
    "permission": [
      {
        "permissionId": "addCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      },
      {
        "permissionId": "getCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      }
    ]
  }
}
```

Customer Operations

Set Customer Lock Status

Verb	URI	Description
PUT	/customers/*customerId*/actions/lock	Lock or unlock a customer.

Normal Response Code(s):204

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

User Operations

Create the First User

Verb	URI	Description
POST	/users	Create a new customer entry and add the user as an 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. 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 17. XML User Request

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

<user xmlns="http://docs.rackspacecloud.com/idm/api/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 18. JSON User Request

```
{
  "secret": {
    "secretAnswer": "Francis",
    "secretQuestion": "What is your dogs name?"
  },
  "password": {
    "password": "P@ssword1"
  },
  "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",
  "isLocked": false,
  "isSoftDeleted": false
}
```


Example 19. XML User Response

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

<user
    xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    isSoftDeleted="false" isLocked="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!" />
    <roles>
        <role type="RackspaceDefined"
            customerInum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE"
            customerId="RCN-000-000-000" name="Admin">
        </role>
    </roles>
</user>
```


Example 20. JSON User Response

```
{
  "secret": {
    "secretQuestion": "What is your dogs name?",
    "secretAnswer": "sicnarF"
  },
  "password": {
    "password": "C@n+f001me!"
  },
  "roles": {
    "role": [
      {
        "permissions": {},
        "name": "Admin",
        "customerId": "RCN-234-592-018",
        "customerInum": "@FFFF.FFFF.FFFF.FFFF!2398.1FCB",
        "type": "RackspaceDefined"
      }
    ]},
  "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",
  "timeZone": "America/Chicago",
  "region": "SAT",
  "status": "ACTIVE",
  "isLocked": false,
  "isSoftDeleted": false
}
```

Create a User

Verb	URI	Description
POST	/customers/*customerId*/users	Create a new user. If a blank password is passed in, the API generates a random password for the user.

The Username 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. 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), usernameConflict(409), emailConflict(409), idmFault (500), serviceUnavailable(503)

Example 21. XML User Request

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

<user xmlns="http://docs.rackspacecloud.com/idm/api/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 22. JSON User Request

```
{
  "secret": {
    "secretAnswer": "Francis",
    "secretQuestion": "What is your dogs name?"
  },
  "password": {
    "password": "P@ssword1"
  },
  "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",
  "isLocked": false,
  "isSoftDeleted": false
}
```

Example 23. XML User Response

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

<user
    xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
 isSoftDeleted="false" isLocked="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 24. JSON User Response

```
{
  "secret": {
    "secretQuestion": "What is your dogs name?",
    "secretAnswer": "Francis"
  },
  "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",
  "iname": "@Example.Smith*John",
  "timeZone": "America/Chicago",
  "region": "SAT",
  "status": "ACTIVE",
  "isLocked": false,
  "isSoftDeleted": false
}
```

Get a User

Verb	URI	Description
GET	/customers/*customerId*/users/*userId*	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 25. XML User Response

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

<user
    xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
 isSoftDeleted="false" isLocked="false"
    tatus="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">
 <roles>
  <role type="RackspaceDefined"
            name="Admin">
  </role>
 </roles>
</user>
```


Example 26. JSON User Response

```
{
  "roles": {
    "role": [
      {
        "name": "Admin",
        "type": "RackspaceDefined"
      }
    ]},
  "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",
  "isLocked": false,
  "isSoftDeleted": false
}
```

Update a User

Verb	URI	Description
PUT	/customers/*customerId*/users/*userId*	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 27. XML User Request

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

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


Example 28. JSON User Request

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

Example 29. XML User Response

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

<user xmlns="http://docs.rackspacecloud.com/idm/api/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">
</user>
```

Example 30. JSON 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
}
```

Soft Delete a User

Verb	URI	Description
PUT	/customers/*customerId*/users/*userId*/softDelete	Set a User's softDeleted Flag

Normal Response Code(s):200

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

Example 31. XML User Soft Delete Request

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

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

Example 32. JSON User Soft Delete Request

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

Example 33. XML User SoftDeleted Response

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

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

Example 34. JSON User SoftDeleted Response

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

Delete User

Verb	URI	Description
DELETE	/customers/*customerId*/users/*userId*	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.

Get User List

Verb	URI	Description
GET	/customers/*customerId*/users/?offset=*x*&limit=*y*	Gets a page of users where *x* is the offset and *y* is the limit of records to return.

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 35. XML User List Response

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

<users xmlns="http://docs.rackspacecloud.com/idm/api/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 36. JSON 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
}
```


Add Role for a User

Verb	URI	Description
PUT	/customers/*customerId*/users/*userId*/roles/*roleName*	Add a role to a user.

Normal Response Code(s):204

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

Delete Role from a User

Verb	URI	Description
DELETE	/customers/*customerId*/users/*userId*/roles/*roleName*	Delete a role from a user.

Normal Response Code(s):204

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

Get a user's roles

Verb	URI	Description
GET	/customers/*customerId*/users/*userId*/roles/	Get a user's roles

Normal Response Code(s):200

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

Example 37. XML User Roles Response

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

<roles xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
    <role type="RackspaceDefined"
        name="Admin" />
    <role type="CustomerDefined"
        name="HR" />
</roles>
```


Example 38. JSON User Roles Response

```
{
  "role": [
    {
      "name": "Admin",
      "type": "RackspaceDefined"
    },
    {
      "name": "HR",
      "type": "CustomerDefined"
    }
  ]
}
```

Generate/Reset Auth Service Key

Verb	URI	Description
POST	/customers/*customerId*/users/*username*/key	Generate or reset a user's Auth key.

Normal Response Code(s):200, 203

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 39. XML User Auth Key Response

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

<userApiKey xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    apiKey="403958809d0809834e0809808a" />
```

Example 40. JSON User Auth Key Response

```
{
  "apiKey": "403958809d0809834e0809808a"
}
```

Set User Lock

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

Normal Response Code(s):204

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

Example 41. XML User Lock Request

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

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

Example 42. JSON User Lock Request

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

Example 43. XML User Lock Response

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

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

Example 44. JSON User Lock Response

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

Set User Password

Verb	URI	Description
PUT	/customers/*customerId*/users/*username*/password?recovery=*boolean*	Sets a user's password.

This method takes an optional boolean query parameter (recovery). True indicates that the password change is for Forget Password and will not require the old password. False is the normal reset route when an old password is used to reset the password.

Normal Response Code(s):204

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 45. XML User Password Request

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

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

Example 46. JSON User Password Request

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

Example 47. XML User Password Response

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

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

Example 48. JSON User Password Response

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

Reset User Password

Verb	URI	Description
POST	/customers/*customerId*/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 49. XML Reset User Password Response

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

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

Example 50. JSON Reset User Password Response

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

Get a User Password

Verb	URI	Description
GET	/customers/*customerId*/users/*userId*/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 51. XML User Password Response

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

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

Example 52. JSON User Passwrord Response

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

Initiate Forget Password

Verb	URI	Description
POST	/customers/*customerId*/users/*username*/password/recoveryEmail	Send a request to recover a user's password.

Normal Response Code(s):204

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

Example 53. XML Initiate Forgot Password Request

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

<passwordRecovery xmlns="http://docs.rackspacecloud.com/idm/api/
v1.0"
    subject="Password Recovery"
    replyTo="replay@email.com"
    from="email@email.com"
    templateUrl="http://www.someurl.com"
    callbackUrl="http://www.someurl.com">
    <customParams>
        <params value="Steve" name="FirstName" />
    </customParams>
</passwordRecovery>
```

Example 54. JSON Initiate Forgot Password Request

```
{
  "customParams": {
    "param": [
      {
        "name": "FirstName",
        "value": "Steve"
      }
    ]
  },
  "callbackUrl": "http://www.someurl.com",
  "templateUrl": "http://www.someurl.com",
  "from": "email@email.com",
  "replyTo": "replay@email.com",
  "subject": "Password Recovery"
}
```

Get Password Recovery Token

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

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 55. XML Password Recovery Token Response

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

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

Example 56. JSON Password Recovery Token Response

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

Set User Status

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

Normal Response Code(s):204

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

Example 57. XML User Status Request

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

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

Example 58. JSON User Status Request

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

Example 59. XML User Status Response

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

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


Example 60. JSON User Status Response

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

Set User Secret

Verb	URI	Description
PUT	/customers/*customerId*/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 61. XML User Secret Request

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

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

Example 62. JSON User Secret Request

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

Example 63. XML User Secret Response

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

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

Example 64. JSON User Secret Response

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


Customer Application Operations

Add a Customer Application

Verb	URI	Description
POST	/customers/*customerId*/clients	Add a client.

Normal Response Code(s):201

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

Example 65. XML Add Client Request

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

<client xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    name="Test Application2"
    customerId="RCN-000-000-000">
</client>
```

Example 66. JSON Add Client Request

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

Example 67. XML Add Client Response

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

<client xmlns="http://docs.rackspacecloud.com/idm/api/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">
    <credentials clientSecret="3af738fbeiwu23" />
</client>
```


Example 68. JSON Add Client Response

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

Get a Customer Application

Verb	URI	Description
GET	/customers/*customerId*/clients/*clientId*	Get a client.

Normal Response Code(s):200

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

Example 69. XML Get Client Response

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

<client xmlns="http://docs.rackspacecloud.com/idm/api/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">
    <permissions>
        <permission clientId="IDM"
            customerId="RCN-000-000-000"
            resourceId="addCustomer" />
    </permissions>
</client>
```


Example 70. JSON Get Client Response

```
{ "permissions": {
    "permission": [
      {
        "resourceId": "addCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      }
    ]},
  "clientId": "ab4820dhcb39347",
  "customerId": "RCN-000-000-000",
  "name": "Test Application2",
  "inum": "@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0001",
  "iname": "@Rackspace*Rackspace*ControlPanel",
  "status": "ACTIVE",
  "locked": false,
  "softDeleted": false
}
```

Get All Applications for a Customer

Verb	URI	Description
GET	/customers/*customerId*/clients/	Get all clients for *customerId*

Normal Response Code(s):200

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

Example 71. XML Get Clients Response

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

<clients xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
 <client
    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">
 </client>
 <client
    softDeleted="false"
    locked="false"
    status="ACTIVE"
    iname="@Rackspace*Rackspace*CloudServers"
    inum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE!0002"
    name="Test Application2"
    customerId="RCN-000-000-000"
    clientId="632h389cv902bde">
 </client>
</clients>
```


Example 72. JSON Get Clients Response

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

Soft Delete a Customer Application

Verb	URI	Description
DELETE	/customers/*customerId*/clients/*clientId*/softdeleted	Delete a customer application.

Normal Response Code(s):204

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

Example 73. XML Soft Delete Customer Application Request

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

<client xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    softDeleted="true" />
```


Example 74. JSON Soft Delete Customer Application Request

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

Example 75. XML Soft Delete Customer Application Response

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

<client xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    softDeleted="true" />
```

Example 76. JSON Soft Delete Customer Application Response

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

Delete a customer application

Verb	URI	Description
DELETE	/customers/*customerId*/clients/*clientId*	Delete a client.

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.

Get Permissions for a Customer Application

Verb	URI	Description
GET	/customers/*customerId*/ clients/*clientId*/permissions	Get all permissions for a *clientId* of a *customerId*

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 77. XML Permissions Response

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

<permissions xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
    <granted>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="addCustomer"/>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="getCustomer"/>
    </granted>
    <defined>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="addCustomer"/>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="getCustomer"/>
    </defined>
</permissions>
```


Example 78. JSON Permissions Response

```
{
  "granted": {
    "permission": [
      {
        "permissionId": "addCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      },
      {
        "permissionId": "getCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      }
    ]
  },
  "defined": {
    "permission": [
      {
        "permissionId": "addCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      },
      {
        "permissionId": "getCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      }
    ]
  }
}
```

Get Granted Permissions for a Customer Application

Verb	URI	Description
GET	/customers/*customerId*/clients/*clientId*/permissions/granted	Get all permissions for a *clientId* of a *customerId*

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 79. XML Granted Permissions Response

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

<permissions xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
    <granted>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="addCustomer"/>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="getCustomer"/>
    </granted>
</permissions>
```

Example 80. JSON Granted Permissions Response

```
{
  "granted": {
    "permission": [
      {
        "permissionId": "addCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      },
      {
        "permissionId": "getCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      }
    ]
  }
}
```

Get Defined Permissions for a Customer Application

Verb	URI	Description
GET	/customers/*customerId*/clients/*clientId*/permissions/defined	Get all permissions for a *clientId* of a *customerId*

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 81. XML Defined Permissions Response

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

<permissions xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
    <defined>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="addCustomer"/>
        <permission
            clientId="IDM"
            customerId="RCN-000-000-000"
            permissionId="getCustomer"/>
    </defined>
</permissions>
```

Example 82. JSON Defined Permissions Response

```
{
  "defined": {
    "permission": [
      {
        "permissionId": "addCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      },
      {
        "permissionId": "getCustomer",
        "customerId": "RCN-000-000-000",
        "clientId": "IDM"
      }
    ]
  }
}
```

Add a Defined Permission for a Customer Application

Verb	URI	Description
POST	/customers/*customerId*/clients/*clientId*/permissions/defined	Add a resource for a *clientId* of a *customerId*

Normal Response Code(s):201

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

Example 83. XML Add a Defined Permission Request

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

<permission xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    type="application/text"
    clientId="IDM"
    customerId="RCN-000-000-000"
    permissionId="addCustomer">POST /customers</permission>
```

Example 84. JSON Add a Defined Permission Request

```
{
  "value": "POST /customers",
  "permissionId": "addCustomer",
  "customerId": "RCN-000-000-000",
  "clientId": "IDM",
  "type": "application/text"
}
```

Example 85. XML Add a Defined Permission Response

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

<permission xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    type="application/text"
    clientId="IDM"
    customerId="RCN-000-000-000"
    permissionId="addCustomer">POST /customers</permission>
```

Example 86. JSON Add a Defined Permission Response

```
{
  "value": "POST /customers",
  "permissionId": "addCustomer",
  "customerId": "RCN-000-000-000",
  "clientId": "IDM",
  "type": "application/text"
}
```

Get a Defined Permission for a Customer Application

Verb	URI	Description
GET	/customers/*customerId*/clients/*clientId*/permissions/defined/*permissionId*	Get a *permissionId* for a *clientId* of a *customerId*

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 87. XML Get Defined Permission Response

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

<permission xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    type="application/text"
    clientId="IDM"
    customerId="RCN-000-000-000"
    permissionId="addCustomer">POST /customers</permission>
```

Example 88. JSON Get Defined Permission Response

```
{
  "value": "POST /customers",
  "permissionId": "addCustomer",
  "customerId": "RCN-000-000-000",
  "clientId": "IDM",
  "type": "application/text"
}
```

Update a Defined Permission for a Customer Application

Verb	URI	Description
PUT	/customers/*customerId*/ clients/*clientId*/permissions/ defined/*permissionId*	Update a defined permission for a *clientId* of a *customerId*

Normal Response Code(s):201

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

Example 89. XML Update a Defined Permission Request

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

<permission xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    type="application/text"
    clientId="IDM"
    customerId="RCN-000-000-000"
    permissionId="addCustomer">POST /customers</permission>
```


Example 90. JSON Update a Defined Permission Request

```
{
  "value": "POST /customers",
  "permissionId": "addCustomer",
  "customerId": "RCN-000-000-000",
  "clientId": "IDM",
  "type": "application/text"
}
```

Example 91. XML Update a Defined Permission Response

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

<permission xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
    type="application/text"
    clientId="IDM"
    customerId="RCN-000-000-000"
    permissionId="addCustomer">POST /customers</permission>
```

Example 92. JSON Update a Defined Permission Response

```
{
  "value": "POST /customers",
  "permissionId": "addCustomer",
  "customerId": "RCN-000-000-000",
  "clientId": "IDM",
  "type": "application/text"
}
```

Delete a Defined Permission for a Customer Application

Verb	URI	Description
DELETE	/customers/*customerId*/ clients/*clientId*/permissions/ defined/*permissionId*	Delete a defined permission for a *clientId* of a *customerId*

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.

Password Operations

Get Password Rules

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 93. XML Get Password Rules Response

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

<passwordRules
    xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
    <passwordRule
        message="Password must be at least 7 characters."
        name="Mininum Legth"
        id="1" />
    <passwordRule
        message="Password must contain a lowercase."
        name="Lowercase Character"
        id="2" />
</passwordRules>
```

Example 94. JSON Get Password Rules Response

```
{
  "passwordRule": [
    {
      "id": 1,
      "name": "Mininum Legth",
      "message": "Password must be at least 7 characters long."
    },
    {
      "id": 2,
      "name": "Lowercase Character",
      "message": "Password must contain a lowercase character."
    }
  ]
}
```


Password validation check

Verb	URI	Description
GET	/passwordrules/validation/*password*	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 95. XML Password Validation Response

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

<passwordValidation xmlns="http://docs.rackspacecloud.com/idm/api/
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 96. 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
}
```

