



IdM Developer Guide

1/27/11

API v1.0.2011.12

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

Overview	1
Concepts	2
Token	2
Customer	2
User	2
Client	2
General API Information	3
Request/Response Types	3
Content Compression	4
API Version	5
List Versions	5
Get Version Details	6
Faults	7
Service Developer Operations	9
Overview	9
Token Operations	9
Authenticate	9
Validate Token	12
Revoke Token	15
Get Token Permissions	15
Customer Operations	16
Set Customer Lock Status	16
Reset client secret	17
User Operations	17
Create the First User	17
Create a User	20
Get a User	23
Get a User by Mosso ID	25
Get a User by NAST ID	27
Get a User by UserName	29
Update a User	31
Soft Delete a User	32
Delete User	33
Get User List	33
Add Role for a User	36
Delete Role from a User	36
Get a user's roles	36
Generate/Reset Auth Service Key	37
Set User Lock	37
Set User Password	38
Reset User Password	39
Get a User Password	40
Initiate Forget Password	40
Get Password Recovery Token	41
Set User Status	42
Set User Secret	43
Authentication Operations	44
Authenticate a user by Mosso ID	44
Authenticate a user by NAST ID	45

Authenticate a user by Username	46
Customer Application Operations	48
Add a Customer Application	48
Get a Customer Application	49
Get All Applications for a Customer	50
Soft Delete a Customer Application	52
Delete a customer application	53
Get Permissions for a Customer Application	53
Get Granted Permissions for a Customer Application	55
Get Defined Permissions for a Customer Application	56
Add a Defined Permission for a Customer Application	57
Get a Defined Permission for a Customer Application	58
Update a Defined Permission for a Customer Application	59
Delete a Defined Permission for a Customer Application	60
Password Operations	61
Get Password Rules	61
Password validation check	62
Admin Operations	63
Get Base URLs	63
Add Base URL	65
Get a Base URL	66
Delete a Base URL	67
Get Base Url Refs	67
Add Base URL Ref	69
Get Base URL Ref	70
Delete a Base URL Ref	70
Get Service catalog for user	71

List of Tables

1. Response Types	3
2. Compression Headers	5
3. Fault Types	8

List of Examples

1. JSON Request with Headers	3
2. XML Response with Headers	4
3. XML Versions Response	6
4. XML Version Response	6
5. XML Fault Response	7
6. JSON Fault Response	7
7. XML Not Found Fault	7
8. JSON Not Found Fault	7
9. XML Auth Request	9
10. JSON Auth Request	9
11. XML Auth Response	10
12. JSON Auth Response	11
13. XML Validate Token Response	13
14. JSON Validate Token Response	14
15. XML Permission List Response	15
16. JSON Permission List Response	16
17. XML Response	17
18. JSON Response	17
19. XML User Request	18
20. JSON User Request	18
21. XML User Response	19
22. JSON User Response	20
23. XML User Request	21
24. JSON User Request	22
25. XML User Response	22
26. JSON User Response	23
27. XML User Response	24
28. JSON User Response	25
29. XML User Response	26
30. JSON User Response	27
31. XML User Response	28
32. JSON User Response	29
33. XML User Response	30
34. JSON User Response	31
35. XML User Request	31
36. JSON User Request	32
37. XML User Response	32
38. JSON User Response	32
39. XML User Soft Delete Request	33
40. JSON User Soft Delete Request	33
41. XML User SoftDeleted Response	33
42. JSON User SoftDeleted Response	33
43. XML User List Response	34
44. JSON User List Response	35
45. XML User Roles Response	36
46. JSON User Roles Response	37
47. XML User Auth Key Response	37
48. JSON User Auth Key Response	37
49. XML User Lock Request	38

50. JSON User Lock Request	38
51. XML User Lock Response	38
52. JSON User Lock Response	38
53. XML User Password Request	39
54. JSON User Password Request	39
55. XML User Password Response	39
56. JSON User Password Response	39
57. XML Reset User Password Response	40
58. JSON Reset User Password Response	40
59. XML User Password Response	40
60. JSON User Password Response	40
61. XML Initiate Forgot Password Request	41
62. JSON Initiate Forgot Password Request	41
63. XML Password Recovery Token Response	42
64. JSON Password Recovery Token Response	42
65. XML User Status Request	42
66. JSON User Status Request	42
67. XML User Status Response	42
68. JSON User Status Response	43
69. XML User Secret Request	43
70. JSON User Secret Request	43
71. XML User Secret Response	43
72. JSON User Secret Response	43
73. Mosso Credentials	44
74. XML Response	44
75. JSON Response	45
76. NAST Credentials	45
77. XML Response	46
78. JSON Response	46
79. User Data	47
80. XML Response	47
81. JSON Response	47
82. XML Add Client Request	48
83. JSON Add Client Request	48
84. XML Add Client Response	48
85. JSON Add Client Response	49
86. XML Get Client Response	49
87. JSON Get Client Response	50
88. XML Get Clients Response	51
89. JSON Get Clients Response	52
90. XML Soft Delete Customer Application Request	52
91. JSON Soft Delete Customer Application Request	53
92. XML Soft Delete Customer Application Response	53
93. JSON Soft Delete Customer Application Response	53
94. XML Permissions Response	54
95. JSON Permissions Response	55
96. XML Granted Permissions Response	56
97. JSON Granted Permissions Response	56
98. XML Defined Permissions Response	57
99. JSON Defined Permissions Response	57
100. XML Add a Defined Permission Request	58

101. JSON Add a Defined Permission Request	58
102. XML Add a Defined Permission Response	58
103. JSON Add a Defined Permission Response	58
104. XML Get Defined Permission Response	59
105. JSON Get Defined Permission Response	59
106. XML Update a Defined Permission Request	59
107. JSON Update a Defined Permission Request	60
108. XML Update a Defined Permission Response	60
109. JSON Update a Defined Permission Response	60
110. XML Get Password Rules Response	61
111. JSON Get Password Rules Response	62
112. XML Password Validation Response	63
113. JSON Password Validation Response	63
114. XML Get BaseURLs Response	64
115. JSON Get Password Rules Response	65
116. Data to be passed in	66
117. XML Response	66
118. JSON Response	67
119. XML response	68
120. JSON response	69
121. Data to be passed in	70
122. XML Response	70
123. JSON Response	70
124. XML Response	71
125. JSON Response	72

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 initially 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 may 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/ <i>tokenId</i> ? 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/ <i>tokenId</i>	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/ <i>tokenId</i> /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/ <i>customerId</i> /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)

Reset client secret

Verb	URI	Description
POST	/customers/ <i>customerId</i> / clients/ <i>clientId</i> /secret	Reset client 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 17. XML Response

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

<clientCredentials xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
  clientSecret="c582fe94dda4c0da19bf234c504db0b63df96daa" />
```

Example 18. JSON Response

```
{ "clientSecret": "eca2d413a1790dc6473842ab5a49387a29996757" }
```

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 19. 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+f00lme!" />
</user>
```

Example 20. 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 21. 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+f00lme!" />
  <roles>
    <role type="RackspaceDefined"
      customerInum="@FFFF.FFFF.FFFF.FFFF!EEEE.EEEE"
      customerId="RCN-000-000-000" name="Admin">
    </role>
  </roles>
</user>
```

Example 22. JSON User Response

```
{
  "secret": {
    "secretQuestion": "What is your dogs name?",
    "secretAnswer": "sicnarF"
  },
  "password": {
    "password": "C@n+f00lme!"
  },
  "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/ <i>customerId</i> /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 23. 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+f00lme!" />
</user>
```

Example 24. 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 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"
  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+f00lme!" />
</user>
```

Example 26. JSON User Response

```
{
  "secret": {
    "secretQuestion": "What is your dogs name?",
    "secretAnswer": "Francis"
  },
  "password": {
    "password": "C@n+f00lme!"
  },
  "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/ <i>customerId</i> /users/ <i>userId</i>	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 27. 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 28. 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
}
```

Get a User by Mosso ID

Verb	URI	Description
GET	/mosso/mossoId	Get a user with mossoId.

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 29. XML User Response

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

<user xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
  username="gauser"
  customerId="RCN-QATestingCustomer"
  customerInum="@!FFFF.FFFF.FFFF.FFFF!EEEE.999"
  email="gauser@example.com"
  personId="RPN-gauser-000-10001"
  firstName="gauser"
  middleName="none"
  lastName="Tester"
  displayName="gauser"
  prefLanguage="en-us"
  inum="@!FFFF.FFFF.FFFF.FFFF!EEEE.999!10001"
  iname="@Rackspace.TestCustomer*QATester"
  status="ACTIVE"
  locked="false"
  softDeleted="false"
  country="USA"
  timeZone="America/Chicago"
  mossoId="999112358"
  nastId="nastid-gauser"/>
```

Example 30. JSON User Response

```
{
  "secret": null,
  "password": null,
  "apiKey": null,
  "roles": null,
  "username": "gauser",
  "customerId": "RCN-QATestingCustomer",
  "customerInum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.999",
  "email": "gauser@example.com",
  "personId": "RPN-gauser-000-10001",
  "firstName": "gauser",
  "middleName": "none",
  "lastName": "Tester",
  "displayName": "gauser",
  "prefLanguage": "en-us",
  "inum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.999!10001",
  "iname": "@Rackspace.TestCustomer*QATester",
  "region": null,
  "status": "ACTIVE",
  "locked": false,
  "softDeleted": false,
  "country": "USA",
  "timeZone": "America/Chicago",
  "mossoId": 999112358,
  "nastId": "nastid-gauser",
  "created": null,
  "updated": null
}
```

Get a User by NAST ID

Verb	URI	Description
GET	/nast/nastId	Get a user with a nastId.

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 31. XML User Response

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

<user xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
  username="gauser"
  customerId="RCN-QATestingCustomer"
  customerInum="@!FFFF.FFFF.FFFF.FFFF!EEEE.999"
  email="gauser@example.com"
  personId="RPN-gauser-000-10001"
  firstName="gauser"
  middleName="none"
  lastName="Tester"
  displayName="gauser"
  prefLanguage="en-us"
  inum="@!FFFF.FFFF.FFFF.FFFF!EEEE.999!10001"
  iname="@Rackspace.TestCustomer*QATester"
  status="ACTIVE"
  locked="false"
  softDeleted="false"
  country="USA"
  timeZone="America/Chicago"
  mossoId="999112358"
  nastId="nastid-gauser"/>
```

Example 32. JSON User Response

```
{
  "secret": null,
  "password": null,
  "apiKey": null,
  "roles": null,
  "username": "gauser",
  "customerId": "RCN-QATestingCustomer",
  "customerInum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.999",
  "email": "gauser@example.com",
  "personId": "RPN-gauser-000-10001",
  "firstName": "gauser",
  "middleName": "none",
  "lastName": "Tester",
  "displayName": "gauser",
  "prefLanguage": "en-us",
  "inum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.999!10001",
  "iname": "@Rackspace.TestCustomer*QATester",
  "region": null,
  "status": "ACTIVE",
  "locked": false,
  "softDeleted": false,
  "country": "USA",
  "timeZone": "America/Chicago",
  "mossoId": 999112358,
  "nastId": "nastid-gauser",
  "created": null,
  "updated": null
}
```

Get a User by UserName

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

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 33. XML User Response

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

<user xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
  username="gauser"
  customerId="RCN-QATestingCustomer"
  customerInum="@!FFFF.FFFF.FFFF.FFFF!EEEE.999"
  email="gauser@example.com"
  personId="RPN-gauser-000-10001"
  firstName="gauser"
  middleName="none"
  lastName="Tester"
  displayName="gauser"
  prefLanguage="en-us"
  inum="@!FFFF.FFFF.FFFF.FFFF!EEEE.999!10001"
  iname="@Rackspace.TestCustomer*QATester"
  status="ACTIVE"
  locked="false"
  softDeleted="false"
  country="USA"
  timeZone="America/Chicago"
  mossoId="999112358"
  nastId="nastid-gauser"/>
```

Example 34. JSON User Response

```
{
  "secret": null,
  "password": null,
  "apiKey": null,
  "roles": null,
  "username": "qauser",
  "customerId": "RCN-QATestingCustomer",
  "customerInum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.999",
  "email": "qauser@example.com",
  "personId": "RPN-qauser-000-10001",
  "firstName": "qauser",
  "middleName": "none",
  "lastName": "Tester",
  "displayName": "qauser",
  "prefLanguage": "en-us",
  "inum": "@!FFFF.FFFF.FFFF.FFFF!EEEE.999!10001",
  "iname": "@Rackspace.TestCustomer*QATester",
  "region": null,
  "status": "ACTIVE",
  "locked": false,
  "softDeleted": false,
  "country": "USA",
  "timeZone": "America/Chicago",
  "mossoId": 999112358,
  "nastId": "nastid-qauser",
  "created": null,
  "updated": null
}
```

Update a User

Verb	URI	Description
PUT	/customers/ <i>customerId</i> /users/ <i>userId</i>	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 35. 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 36. JSON User Request

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

Example 37. 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 38. 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/ <i>customerId</i> /users/ <i>userId</i> /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 39. 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 40. JSON User Soft Delete Request

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

Example 41. XML User SoftDeleted Response

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

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

Example 42. JSON User SoftDeleted Response

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

Delete User

Verb	URI	Description
DELETE	/customers/ <i>customerId</i> /users/ <i>userId</i>	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/ <i>customerId</i> /users/?offset= <i>x</i> &limit= <i>y</i>	Gets a page of users where <i>x</i> is the offset and <i>y</i> 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 43. 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 44. 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/ <i>customerId</i> /users/ <i>userId</i> /roles/ <i>roleName</i>	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/ <i>customerId</i> /users/ <i>userId</i> /roles/ <i>roleName</i>	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/ <i>customerId</i> /users/ <i>userId</i> /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 45. 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 46. JSON User Roles Response

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

Generate/Reset Auth Service Key

Verb	URI	Description
POST	/customers/ <i>customerId</i> /users/ <i>username</i> /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 47. 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 48. JSON User Auth Key Response

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

Set User Lock

Verb	URI	Description
PUT	/customers/ <i>customerId</i> /users/ <i>username</i> /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 49. XML User Lock Request

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

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

Example 50. JSON User Lock Request

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

Example 51. XML User Lock Response

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

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

Example 52. JSON User Lock Response

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

Set User Password

Verb	URI	Description
PUT	/customers/ <i>customerId</i> /users/ <i>username</i> /password?recovery= <i>boolean</i>	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 53. 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 54. JSON User Password Request

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

Example 55. XML User Password Response

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

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

Example 56. JSON User Password Response

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

Reset User Password

Verb	URI	Description
POST	/customers/ <i>customerId</i> /users/ <i>username</i> /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 57. 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 58. JSON Reset User Password Response

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

Get a User Password

Verb	URI	Description
GET	/customers/ <i>customerId</i> /users/ <i>userId</i> /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 59. XML User Password Response

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

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

Example 60. JSON User Password Response

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

Initiate Forget Password

Verb	URI	Description
POST	/customers/ <i>customerId</i> /users/ <i>username</i> /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 61. XML Initiate Forgot Password Request

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

<passwordRecovery xmlns="http://docs.rackspacecloud.com/idm/api/
v1.0"
  templateUrl="http://cp.rackspace.com/files/
passwordRecovery.txt"
  callbackUrl="http://cp.rackspace.com/passwordRecovery.html"
  from="Rackspace Control Panel"
  replyTo="cpsupport@rackspace.com"
  subject="Rackspace Password Reset">
  <customParams>
    <customParam name="%%FirstName%%" value="Steve" />
    <customParam name="%%LastName%%" value="Smith" />
  </customParams>
</passwordRecovery>
```

Example 62. JSON Initiate Forgot Password Request

```
{
  "customParams" : [
    {
      "name": "%%FirstName%%",
      "value": "Steve"
    },
    {
      "name": "%%LastName%%",
      "value": "Smith"
    }
  ],
  "templateUrl": "http://cp.rackspace.com/files/
passwordRecovery.txt",
  "callbackUrl": "http://cp.rackspace.com/
passwordRecovery.html",
  "from": "Rackspace Control Panel",
  "replyTo": "cpsupport@rackspace.com",
  "subject": "Rackspace Password Reset"
}
```

Get Password Recovery Token

Verb	URI	Description
POST	/customers/ <i>customerId</i> /users/ <i>username</i> /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 63. 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 64. JSON Password Recovery Token Response

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

Set User Status

Verb	URI	Description
PUT	/customers/ <i>customerId</i> /users/ <i>username</i> /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 65. XML User Status Request

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

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

Example 66. JSON User Status Request

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

Example 67. XML User Status Response

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

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

Example 68. JSON User Status Response

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

Set User Secret

Verb	URI	Description
PUT	/customers/ <i>customerId</i> /users/ <i>username</i> /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 69. 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 70. JSON User Secret Request

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

Example 71. 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 72. JSON User Secret Response

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

Authentication Operations

Authenticate a user by Mosso ID

Verb	URI	Description
POST	/auth/mosso	Authenticate a user by mosso Id. Pass in the mosso credentials shown below.

Normal Response Code(s):200

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

Example 73. Mosso Credentials

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

<mossoCredentials xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
  mossoId="999112358"
  key="gauser-000-001" />
```

Example 74. XML Response

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

<cloudAuth xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
  <token id="DFW-c6dfc4456f6545f8b8265ff5b6bda775"
    expires_in="16640"/>
  <serviceCatalog xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
    <service name="cloudFiles">
      <endpoint region="DFW"
        v1Default="true"
        publicURL="https://storage.clouddrive.com/v1/KovacsNastId"
        internalURL="https://storage-snet.clouddrive.com/v1/KovacsNastId"/>
    </service>
    <service name="cloudServers">
      <endpoint v1Default="false"
        publicURL="https://servers.api.rackspacecloud.com/v1.0/90000"/>
    </service>
  </serviceCatalog>
</cloudAuth>
```

Example 75. JSON Response

```
{ "token": {  
  "id": "DFW-c6dfc4456f6545f8b8265ff5b6bda775",  
  "expires_in": 16110  
}}
```

Authenticate a user by NAST ID

Verb	URI	Description
POST	/auth/nast	Authenticate a user by nast Id. Pass in the nast credentials shown below.

Normal Response Code(s):200

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

Example 76. NAST Credentials

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<nastCredentials xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"  
  nastId="nastid-qauser"  
  key="qauser-000-001" />
```

Example 77. XML Response

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

<cloudAuth xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
  <token id="DFW-c6dfc4456f6545f8b8265ff5b6bda775"
    expires_in="16640"/>
  <serviceCatalog xmlns="http://docs.rackspacecloud.com/idm/api/
v1.0">
    <service name="cloudFiles">
      <endpoint region="DFW"
        v1Default="true"
        publicURL="https://storage.clouddrive.com/v1/
KovacsNastId"
        internalURL="https://storage-snet.clouddrive.com/v1/
KovacsNastId"/>
    </service>
    <service name="cloudServers">
      <endpoint v1Default="false"
        publicURL="https://servers.api.rackspacecloud.com/
v1.0/90000"/>
    </service>
  </serviceCatalog>
</cloudAuth>
```

Example 78. JSON Response

```
{ "token": {
  "id": "DFW-c6dfc4456f6545f8b8265ff5b6bda775",
  "expires_in": 16110
}}
```

Authenticate a user by Username

Verb	URI	Description
POST	/auth	Authenticate a user by username. Pass in the username credentials shown below.

Normal Response Code(s):200

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

Example 79. User Data

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

<usernameCredentials xmlns="http://docs.rackspacecloud.com/idm/
api/v1.0"
  username="gauser"
  key="gauser-000-001" />
```

Example 80. XML Response

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

<cloudAuth xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
  <token id="DFW-c6dfc4456f6545f8b8265ff5b6bda775"
    expires_in="16640"/>
  <serviceCatalog xmlns="http://docs.rackspacecloud.com/idm/api/
v1.0">
    <service name="cloudFiles">
      <endpoint region="DFW"
        v1Default="true"
        publicURL="https://storage.clouddrive.com/v1/
KovacsNastId"
        internalURL="https://storage-snet.clouddrive.com/v1/
KovacsNastId"/>
    </service>
    <service name="cloudServers">
      <endpoint v1Default="false"
        publicURL="https://servers.api.rackspacecloud.com/
v1.0/90000"/>
    </service>
  </serviceCatalog>
</cloudAuth>
```

Example 81. JSON Response

```
{ "token": {
  "id": "DFW-c6dfc4456f6545f8b8265ff5b6bda775",
  "expires_in": 16110
}}
```

Customer Application Operations

Add a Customer Application

Verb	URI	Description
POST	/customers/ <i>customerId</i> /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 82. 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 83. JSON Add Client Request

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

Example 84. 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 85. 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/ <i>customerId</i> /clients/ <i>clientId</i>	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 86. 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 87. 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/ <i>customerId</i> /clients/	Get all clients for <i>customerId</i>

Normal Response Code(s):200

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

Example 88. 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 89. 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/ <i>customerId</i> / clients/ <i>clientId</i> /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 90. 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 91. JSON Soft Delete Customer Application Request

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

Example 92. 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 93. JSON Soft Delete Customer Application Response

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

Delete a customer application

Verb	URI	Description
DELETE	/customers/ <i>customerId</i> /clients/ <i>clientId</i>	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/ <i>customerId</i> /clients/ <i>clientId</i> /permissions	Get all permissions for a <i>clientId</i> of a <i>customerId</i>

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 94. 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 95. 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/ <i>customerId</i> / clients/ <i>clientId</i> /permissions/granted	Get all permissions for a <i>clientId</i> of a <i>customerId</i>

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 96. 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 97. 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/ <i>customerId</i> / clients/ <i>clientId</i> /permissions/defined	Get all permissions for a <i>clientId</i> of a <i>customerId</i>

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 98. 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 99. 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/ <i>customerId</i> /clients/ <i>clientId</i> /permissions/defined	Add a resource for a <i>clientId</i> of a <i>customerId</i>

Normal Response Code(s):201

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

Example 100. 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 101. JSON Add a Defined Permission Request

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

Example 102. 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 103. 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/ <i>customerId</i> / clients/ <i>clientId</i> /permissions/ defined/ <i>permissionId</i>	Get a <i>permissionId</i> for a <i>clientId</i> of a <i>customerId</i>

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 104. 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 105. 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/ <i>customerId</i> / clients/ <i>clientId</i> /permissions/ defined/ <i>permissionId</i>	Update a defined permission for a <i>clientId</i> of a <i>customerId</i>

Normal Response Code(s):201

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

Example 106. 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 107. JSON Update a Defined Permission Request

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

Example 108. 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 109. 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/ <i>customerId</i> / clients/ <i>clientId</i> /permissions/ defined/ <i>permissionId</i>	Delete a defined permission for a <i>clientId</i> of a <i>customerId</i>

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 110. XML Get Password Rules Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<passwordRuleResults
  xmlns="http://docs.rackspacecloud.com/idm/api/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 111. 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."
    }
  ]
}
```

Password validation check

Verb	URI	Description
GET	/passwordrules/validation/ <i>password</i>	Check Password Validation for <i>password</i>

Normal Response Code(s):200

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

Example 112. 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 113. JSON Password Validation Response

```
{ "passwordRuleResults": {
  "passwordRuleResults": [
    {
      "passed": true,
      "ruleId": 1, "ruleName":
      "Mininum 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
}
```

Admin Operations

Get Base URLs

Verb	URI	Description
GET	/baseurls	Get base urls

Normal Response Code(s):200

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

Example 114. XML Get BaseURLs Response

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

<baseURLs xmlns="http://docs.rackspacecloud.com/idm/api/v1.0">
  <baseURL
    id="111111"
    userType="NAST"
    serviceName="cloudFiles"
    region="DFW"
    publicURL="https://storage.clouddrive.com/v1"
    internalURL="https://storage-snet.clouddrive.com/v1"
    default="true"/>
  <baseURL
    id="222222"
    userType="MOSSO"
    serviceName="cloudServers"
    publicURL="https://servers.api.rackspacecloud.com/v1.0"
    default="true"/>
</baseURLs>
```

Example 115. JSON Get Password Rules Response

```
{ "baseURL": [
  {
    "id": 111111,
    "userType": "NAST",
    "serviceName": "cloudFiles",
    "region": "DFW",
    "publicURL": "https://storage.clouddrive.com/v1",
    "internalURL": "https://storage-snet.clouddrive.com/v1",
    "adminURL": null,
    "default": true
  },
  {
    "id": 222222,
    "userType": "MOSSO",
    "serviceName": "cloudServers",
    "region": null,
    "publicURL": "https://servers.api.rackspacecloud.com/v1.0",
    "internalURL": null,
    "adminURL": null,
    "default": true
  }
]}
```

Add Base URL

Verb	URI	Description
POST	/baseurls	Add base urls. Pass the base url data (shown below).

Normal Response Code(s):200

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

Example 116. Data to be passed in

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

<baseURL xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
  id="111114"
  userType="NAST"
  serviceName="cloudFiles"
  region="DFW"
  publicURL="https://storage.clouddrive.com/v1"
  internalURL="https://storage-snet.clouddrive.com/v1"
  adminURL="https://storage-snet.clouddrive.com/v1"
  default="true"/>
```

Get a Base URL

Verb	URI	Description
GET	/baseurls/ <i>baseUrlId</i>	Get a base url

Normal Response Code(s):200

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

Example 117. XML Response

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

<Response xmlns="http://10.127.7.164/v1.0/baseurls/111111"
  adminURL="https://storage-snet.clouddrive.com/v1"
  default="true"
  id="111111"
  internalURL="https://storage-snet.clouddrive.com/v1"
  publicURL="https://storage.clouddrive.com/v1"
  region="DFW"
  serviceName="cloudFiles"
  userType="NAST"/>
```

Example 118. JSON Response

```

{
  "id": 111111,
  "userType": "NAST",
  "serviceName": "cloudFiles",
  "region": "DFW",
  "publicURL": "https://storage.clouddrive.com/v1",
  "internalURL": "https://storage-snet.clouddrive.com/v1",
  "adminURL": null,
  "default": true
}

```

Delete a Base URL

Verb	URI	Description
DELETE	/baseurls/ <i>baseUrlId</i>	Delete a base url

Normal Response Code(s):200

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

Get Base Url Refs

Verb	URI	Description
GET	/users/ <i>username</i> /baseurlrefs	Gets a base url reference

Normal Response Code(s):200

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

Example 119. XML response

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

<baseURLRefs xmlns="http://docs.rackspacecloud.com/auth/api/v1.1">
  <baseURLRef
    href="https://auth.api.rackspacecloud.com/v1.1/baseURLs/1"
    id="1"
    v1Default="true" />
  <baseURLRef
    href="https://auth.api.rackspacecloud.com/v1.1/baseURLs/2"
    id="2" />
  <baseURLRef
    href="https://auth.api.rackspacecloud.com/v1.1/baseURLs/3"
    id="3"
    v1Default="true" />
  <baseURLRef
    href="https://auth.api.rackspacecloud.com/v1.1/baseURLs/4"
    id="4" />
  <baseURLRef
    href="https://auth.api.rackspacecloud.com/v1.1/baseURLs/5"
    id="5"
    v1Default="true" />
</baseURLRefs>
```

Example 120. JSON response

```
{
  "baseURLRefs" : [
    {
      "id" : 1,
      "href" : "https://auth.api.rackspacecloud.com/v1.1/baseURLs/1",
      "v1Default" : true
    },
    {
      "id" : 2,
      "href" : "https://auth.api.rackspacecloud.com/v1.1/baseURLs/2"
    },
    {
      "id" : 3,
      "href" : "https://auth.api.rackspacecloud.com/v1.1/baseURLs/3",
      "v1Default" : true
    },
    {
      "id" : 4,
      "href" : "https://auth.api.rackspacecloud.com/v1.1/baseURLs/4"
    },
    {
      "id" : 5,
      "href" : "https://auth.api.rackspacecloud.com/v1.1/baseURLs/5",
      "v1Default" : true
    }
  ]
}
```

Add Base URL Ref

Verb	URI	Description
PUT	/users/ <i>username</i> /baseurlrefs	Add base url Ref. Pass the base url ref data (shown below).

Normal Response Code(s):200

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

Example 121. Data to be passed in

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

<baseURLRef xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
  id="111114"
  href="https://storage-snet.clouddrive.com/v1"
  v1Default="true"/>
```

Get Base URL Ref

Verb	URI	Description
GET	/users/username/baseurlrefs/baseUrlId	Get a base url.

Normal Response Code(s):200

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

Example 122. XML Response

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

<baseURLRef xmlns="http://docs.rackspacecloud.com/idm/api/v1.0"
  id="111111"
  href="https://idm.rackspace.com/v1.0/baseurls/111111"
  v1Default="true" />
```

Example 123. JSON Response

```
{
  "id": 111111,
  "href": "https://idm.rackspace.com/v1.0/baseurls/111111",
  "v1Default": true
}
```

Delete a Base URL Ref

Verb	URI	Description
DELETE	/users/username/baseurls/baseUrlId	Delete a base url ref

Normal Response Code(s):200

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

Get Service catalog for user

Verb	URI	Description
GET	/users/ <i>username</i> /servicecatalog	Get Service Catalog for users.

Normal Response Code(s):200

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

Example 124. XML Response

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

<serviceCatalog xmlns="http://docs.rackspacecloud.com/idm/api/
v1.0">
  <service name="cloudFiles">
    <endpoint region="DFW"
      v1Default="true"
      publicURL="https://storage.clouddrive.com/v1/
KovacsNastId"
      internalURL="https://storage-snet.clouddrive.com/v1/
KovacsNastId"/>
  </service>
  <service name="cloudServers">
    <endpoint v1Default="false"
      publicURL="https://servers.api.rackspacecloud.com/
v1.0/90000"/>
  </service>
</serviceCatalog>
```

Example 125. JSON Response

```
{ "service": [
  {
    "endpoint": [
      {
        "region": "DFW",
        "v1Default": true,
        "publicURL": "https://storage.clouddrive.com/v1/
KovacsNastId",
        "internalURL": "https://storage-snet.clouddrive.com/v1/
KovacsNastId",
        "adminURL": null
      }
    ],
    "name": "cloudFiles"
  },
  {
    "endpoint": [
      {
        "region": null,
        "v1Default": false,
        "publicURL": "https://servers.api.rackspacecloud.com/
v1.0/90000",
        "internalURL": null,
        "adminURL": null
      }
    ],
    "name": "cloudServers"
  }
]
```