Rackspace Cloud Keep API 1.0

Use the following links to go directly to user and reference information for using the Rackspace Cloud Keep service API.

Deleted: jump

- Getting Started Guide
- Developer Guide
- API Reference

About the API

The Rackspace Cloud Keep service provides a REST API that enables secure life-cycle management of keys and credentials, called *secrets*, on behalf of customers. It is based on the OpenStack Key Manager service (code named Barbican), a community-led open-source platform.

Formatted: Font: Italic

You can use the API to securely store and retrieve credentials systematically and enable users to have keys generated on their behalf based on their requested encryption algorithm and bit length.

Deleted: Using the API, y

Rackspace Cloud Keep provides the following APIs:

- Secrets API
- Containers API
- Consumers API
- Quotas API

Moved down [4]: <#>Quotas API¶

Moved (insertion) [4]

Early Access program

Deleted: Program

Rackspace Cloud Keep is available to customers through an Early Access (EA) program. The EA program is an opportunity for customers to work in partnership with Rackspace to ensure that the implemented features align with business needs and can be used with maximum efficiency.

As the service progresses towards General Availability, additional features and event notifications will be added, and it is possible that existing implementations and formats might change.

To learn more, visit the Cloud Keep Signup page.

API contract changes

The API contract is not locked and might change during the Early Access program.

Rackspace will notify customers in release notes when and if the contract does change.

Additional resources

We welcome feedback, comments, and bug reports.

- Send Cloud Keep feedback and support questions to keep@rackspace.com.
- For general issues about Rackspace products, visit the Rackspace Product Feedback page.
- For product updates and announcement through Twitter, see http://twitter.com/rackspace.
- To learn about using Rackspace Cloud SDKs, see SDKs and tools.
- To get information about the APIs for other Rackspace Cloud services, see the Rackspace API Documentation.

Getting Started Guide

Use this Getting Started Guide to learn how to authenticate, send API requests, and complete basic operations with the Cloud Keep API.

For more information about Rackspace Cloud Keep concepts and API operations, see the Developer Guide and the API Reference.

Prerequisites

Comment [KH1]: Update with the new boilerplate.

To run the examples in this guide, you must have the following prerequisites:

- Rackspace Cloud account
- Username and password to access the account
- API key to access Rackspace Cloud services
- Account number
- Command line tool or browser client to communicate with the API service

If you don't have an account, sign up for one. You can find your account number and API key on the Account Settings page in the Cloud Control Panel.

Get your credentials

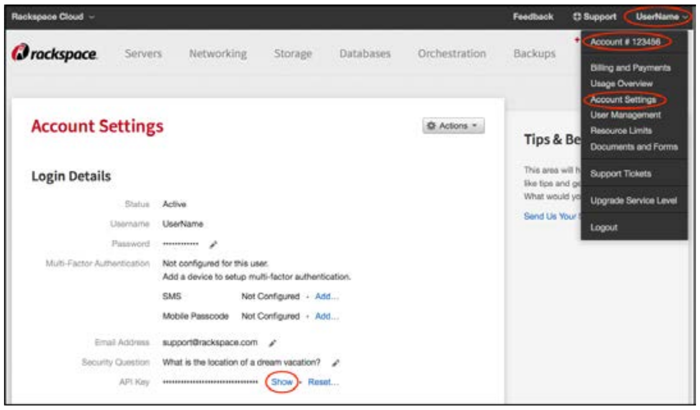
To communicate with Rackspace Cloud services by using the REST API, you need your Rackspace Cloud account username, API key, and account number. To get this information, log in to the Cloud Control Panel.

Deleted: +

Deleted: +

Note: In the API service documentation, the account number is referred to as your *tenant ID* or *tenant name*.

After you log in, click your username on the upper-right side of the top navigation pane. Then, select Account Settings to open the page.



Account Settings

Login Details

Status	Active
Username	UserName
Password	••••••••••••
Multi-Factor Authentication	Not configured for this user. Add a device to setup multi-factor authentication. SMS Not Configured · Add... Mobile Passcode Not Configured · Add...
Email Address	support@rackspace.com
Security Question	What is the location of a dream vacation?
API Key	•••••••••••••••••••••••••••••••• Show · Reset...

Save your API key

1. On the Account Settings page, find the **API Key** field in the Login Details section.
2. Click Show to see the value and copy it to a text editor of your choice.
3. Click Hide to secure the API key value in the browser.

Important: Protect your API key. Do not expose the value in code samples, screen captures, or insecure client-server communications. Also, ensure that the value is not included in source code that is stored in public repositories.

Save your account number

1. On the Account Settings page, scroll down to the Account Details section.
2. Copy and save the account number.

Send requests to the API

This Getting Started Guide shows you how to send requests by using cURL.

Note: You can also use Rackspace Cloud API services by using the following methods:

- If you are developing applications or automation, try using Rackspace SDKs, the Rackspace CLI, or OpenStack client applications.

Formatted: Font: Bold

Deleted: make sure

- For API development, testing, and workflow management in a graphical environment, try interacting with the API by using an application such as Postman or RESTClient for Firefox.

Comment [KH2]: Insert a comma here

cURL is a command-line tool that you can use to interact with REST interfaces. cURL lets you transmit and receive HTTP requests and responses from the command line or a shell script, which enables you to work with the API directly. cURL is available for Linux distributions, Mac OS® X, and Microsoft Windows®. For information about cURL, see http://curl.haxx.se/.

To run the cURL request examples shown in this guide, copy each example directly to the command line or a script.

The following example shows a cURL command for sending an authentication request to the Rackspace Cloud Identity service.

Example: cURL command for sending a JSON request

```
$ curl https://identity.api.rackspacecloud.com/v2.0/tokens  \
      -X POST \
      -d '{"auth":{"RAX-KSKEY:apiKeyCredentials":{"username":"yourUserName","apiKey":"$apiKey"}}}' \
      -H "Content-Type: application/json" \
      | python -m json.tool
```

Deleted: type

In this example, `$apiKey` is an environment variable that stores your API key value. Environment variables make it easier to reference account information in API requests, to reuse the same cURL commands with different credentials, and to keep sensitive information like your API key from being exposed when you send requests to Rackspace Cloud API services. For details about creating environment variables, see Configure environment variables.

Note: The carriage returns in the cURL request examples use a backslash (\) as an escape character. The escape character allows continuation of the command across multiple lines.

The cURL examples in this guide use the following command-line options.

Option	Description
-d	Sends the specified data in a POST request to the HTTP server. Use this option to send a JSON request body to the server.

-H	Specifies an extra HTTP header in the request. You can specify any number of extra headers. Precede each header with the `-H` option.
	
	Common headers in Rackspace API requests are as follows:
	
	• `Content-Type`: Required for operations with a request body. Specifies the format of the request body. Following is the syntax for the header where format is `json`:
	
	`Content-Type: application/json`
	
	• `X-Auth-Token`: Required. Specifies the authentication token.
	
	• `X-Auth-Project-Id`: Optional. Specifies the project ID, which can be your account number or another value.
	
	• `Accept`: Optional. Specifies the format of the response body. Following is the syntax for the header where the format is `json`, which is the default:
	
	`Accept: application/json`

-i	Includes the HTTP header in the output.
-s	Specifies silent or quiet mode, which makes cURL mute. No progress or error messages are shown. If your cURL command is not generating any output, try replacing the `-s` option with `-i`.
-T	Transfers the specified local file to the remote URL.
-X	Specifies the request method to use when communicating with the HTTP server. The specified method is used instead of the default method, which is GET.

Comment [KH3]: Be sure to indent the code blocks in this cell under the bullet items.

Formatted: Indent: Left: 0.28"

Formatted: Indent: Left: 0.28"

For commands that return a response, use json.tool to pretty-print the output by appending the following command to the cURL call:

```
| python -m json.tool
```

To use json.tool, import the JSON module. For information about json.tool, see JSON encoder and decoder.

If you run a Python version earlier than 2.6, import the simplejson module and use simplejson.tool. For information about simplejson.tool, see simplejson encoder and decoder.

If you do not want to pretty-print JSON output, omit this code.

Authenticate to the Rackspace Cloud

Whether you use cURL, a REST client, or a command-line client (CLI) to send requests to the Rackspace Cloud Keep API, you need an authentication token to include in the `X-Auth-Token` header of each API request.

With a valid token, you can send API requests to any of the API service endpoints that you are authorized to use. The authentication response includes a token expiration date. When a token expires, you can send another authentication request to get a new one.

Note: For more information about authentication tokens, see the following topics in the Rackspace Cloud Identity developer documentation.

- Authentication token operations
  The examples in the Getting Started Guide show how to authenticate by using username and API key credentials, which is a more secure way to communicate with API services. The authentication token operations reference describes other types of credentials that you can use for authentication.

- Manage tokens and token expiration

Follow these steps to authenticate to the Rackspace Cloud by using cURL:

- Send an authentication request
- Review the authentication response
- Configure environment variables

Send an authentication request

From a command prompt, send a POST tokens request to the Rackspace Cloud Identity service. Include your username and API key as shown in the following example.

Example: Authentication request

```
$ curl https://identity.api.rackspacecloud.com/v2.0/tokens  \
     -X POST \
     -d '{"auth":{"RAX-KSKEY:apiKeyCredentials":{"username":"yourUserName","apiKey":"$apiKey"}}}' \
     -H "Content-type: application/json" \
     | python -m json.tool
```

Review the authentication response

If your credentials are valid, the Identity service returns an authentication response that includes the following information:

- An authentication token
- A service catalog with information about the services that you can access
- User information and role assignments

Note: For detailed information about the authentication response, see Annotated authentication request and response in the Rackspace Cloud Identity documentation.

In the following example response, the ellipsis (...) represents other service endpoints, which are not shown. The values shown in this and other examples vary because the information returned is specific to your account.

Example: Authentication response

```
{
        "access": {
                "token": {
                        "id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
                        "expires": "2014-11-24T22:05:39.115Z",
                        "tenant": {
                                "id": "110011",
                                "name": "110011"
                        },
                        "RAX-AUTH:authenticatedBy": [
                                "APIKEY"
                        ]
                 },
                "serviceCatalog": [
                        {
                                "name": "cloudDatabases",
                                "endpoints": [
                                        {
                                        "publicURL": "https://syd.databases.api.rackspacecloud.com/v1.0/110011",
                                        "region": "SYD",
                                        "tenantId": "110011"
                                        },
                                        {
                                                "publicURL": "https://dfw.databases.api.rackspacecloud.com/v1.0/110011",
                                                "region": "DFW",
                                                "tenantId": "110011"
                                        },
                                        {
                                                "publicURL": "https://ord.databases.api.rackspacecloud.com/v1.0/110011",
                                                "region": "ORD",
                                                "tenantId": "110011"
                                        },
                                        {
                                                "publicURL": "https://iad.databases.api.rackspacecloud.com/v1.0/110011",
                                                "region": "IAD",
                                                "tenantId": "110011"
                                        },
                                        {
                                                "publicURL": "https://hkg.databases.api.rackspacecloud.com/v1.0/110011",
                                                "region": "HKG",
                                                "tenantId": "110011"
                                        }
                                ],
                                "type": "rax:database"
                        },
                ...

                {
                        "name": "cloudDNS",
                        "endpoints": [
                                {
                                        "publicURL": "https://dns.api.rackspacecloud.com/v1.0/110011",
                                        "tenantId": "110011"
                                }
                        ],
                        "type": "rax:dns"
                },
                {
                        "name": "rackCDN",
                        "endpoints": [
                                {
                                        "internalURL": "https://global.cdn.api.rackspacecloud.com/v1.0/110011",
                                        "publicURL": "https://global.cdn.api.rackspacecloud.com/v1.0/110011",
                                        "tenantId": "110011"
                                }
                        ],

                        "type": "rax:cdn"
                }
        ],
        "user": {
                "id": "123456",
                "roles": [
                        {
                                "description": "A Role that allows a user access to keystone Service methods",
                                "id": "6",
                                "name": "compute:default",
                                "tenantId": "110011"
                        },
                        {
                                "description": "User Admin Role.",
                                "id": "3",
                                "name": "identity:user-admin"
                        }
                ],
                "name": "jsmith",
                "RAX-AUTH:defaultRegion": "ORD"
        }
    }
}
```

If the request was successful, you can find the authentication token and other information in the authentication response. You'll need these values to submit requests to the API. See [Configure environment variables]().

If the request failed, review the response message and the following error message descriptions to determine next steps.

- If you see the following message, review the authentication request for syntax or coding errors. If you are using cURL, see the section on [using cURL]().

  ```
  400 Invalid request body: unable to parse Auth data. Please review XML or JSON formatting
  ```

- If you see the following message, verify the authentication credentials submitted in the authentication request. If necessary, contact your Rackspace Cloud Administrator or Rackspace Support to get valid credentials.

  ```
  401 Unable to authenticate user with credentials provided.
  ```

- If you see the following message, verify that your user has the correct service catalog entries and roles to access the service. If necessary, contact your Rackspace Cloud Administrator or Rackspace Support to verify authorization.

  ```
  403 Forbidden. Not Authorized.
  ```

Note: For more information about authentication and authorization errors, see the [Cloud Identity API Reference documentation]().

Configure environment variables

The authentication response returns the following values that you need to include when you make service requests to the Rackspace Cloud Keep API.

Formatted: Indent: Left: 0", Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Indent: Left: 0", Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Indent: Left: 0", Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Deleted: /or

Deleted: your

token ID

> The authentication token ID value is required to confirm your identity each time you access the service. You include it in the `X-Auth-Token` header for each API request. The `expires` attribute indicates the date and time that the token will expire, unless it is revoked prior to the expiration. To get a new token, submit another authentication request. For more information, see Manage tokens and token expiration.

tenant ID

> The tenant ID value provides your account number. For most Rackspace Cloud service APIs, the tenant ID is appended to the API endpoint in the service catalog automatically.

endpoints

> The endpoints object provides the URLs that you can use to access the API service. For guidance on choosing an endpoint, see Service access.

To make it easier to include these and other values in API requests, use the `export` command to create environment variables that can be substituted for the actual values. For example, you can create an `ENDPOINT` variable to store the URL for accessing an API service. To reference the value in an API request, prefix the variable name with a $, for example `$ENDPOINT`.

Note: The environment variables created with the `export` command are valid only for the current terminal session. If you start a new session, run the export commands again.

To reuse the variables across sessions, update the configuration file for your shell environment to include the export statements. For details about using and managing environment variables on different systems, see the Environment variables wiki.

Create environment variables

1. In the `token` section of the authentication response, copy the token `id` and tenant `id` values from the token object. The following example shows sample values only.

```
{
    "access": {
        "token": {
```

```
                    "id": "AA2345631l1NVdD6D1OCauKA0e9fioquZqVlS-
hpbCqQ5Yx1zLOREGf4efBh10CfB5AvjC1yld4ZNJUouE7DA0QB0n5nRbdDsYADA-
ORICIqHNqOVS_kYmedqDh75c_PLe123456789101",
                    "expires": "2015-11-18T18:35:45.013Z",
                    "tenant": {
                          "id": "123456",
                          "name": "123456"
                    },
                    "RAX-AUTH:authenticatedBy": [
                          "APIKEY"
                    ]
  },
```

2. Export the token ID to an environment variable that can be supplied in the X-Auth-Token header required in each API request. Replace `token-id` with the authentication token id value returned in the authentication response.

```
$ export AUTH_TOKEN="token-id"
```

Formatted: code_body Char, Font color: Auto, Border: : (No border)

3. Export the tenant ID to an environment variable that can be supplied in requests that require you to specify a tenant ID or tenant name. Replace `tenant-id` with the authentication token tenant id value returned in the authentication response.

```
$ export TENANT_ID="tenant-id"
```

Formatted: code_body Char, Font color: Auto, Border: : (No border)

4. In the service catalog section of the authentication response, copy the publicURL value for the Rackspace Cloud Keep API, version, and region that you want to access.

   The following example shows the endpoint available for the Rackspace Cloud Keep API.

Deleted: Example:

```
    {
      "name": "cloudKeep",
      "endpoints": [
        {
          "region": "IAD",
          "tenantId": "123456",
          "publicURL": "https://iad.keep.api.rackspacecloud.com"
        }
      ],
      "type": "key-manager"
    }
```

Note: For some services, the publicURL value for Rackspace Cloud Keep API consists of the service access endpoint URL with the tenant ID for your account appended after the /.

5. Export the URL to an environment variable, as shown in the following example. Replace `publicURL` with the publicURL value listed in the service catalog.

```
$ export ENDPOINT="publicURL"
```

Create and manage secrets

You can use the examples in this section to create and manage secrets by using Rackspace Cloud Keep API operations. Example requests are provided in cURL, followed by the response.

Before running the examples, review the Rackspace Cloud Keep concepts to understand the API workflow and use cases.

Note: These examples use the $ENDPOINT and $AUTH_TOKEN environment variables to specify the API endpoint and authentication token for accessing the service. Be sure to configure these variables before running the code samples.

For more information about all Cloud Keep operations, see the API reference.

Store a secret

You can store a secret by submitting a POST request against the secrets resource and including the secret in the ``*payload*`` parameter. You specify the secret payload type in the payload_content_type parameter:

- For text-based secrets, set the payload_content_type parameter to text/plain.
- For binary secrets, set the payload_content_type parameter to application/octet-stream.

Note: Note The secrets resource encrypts and stores client-provided secret information and metadata. Submitting a POST request creates secret metadata. If the payload is provided with the POST request, it is encrypted and stored, and then linked with this metadata. If

Deleted: Replace

Formatted: code_body Char, Font color: Auto, Border: : (No border)

Deleted: include

Comment [KH4]: For some reason, the tagging is appearing, instead of the correct formatting.

Comment [KH5]: Delete the word *Note*.

Deleted: then

no payload is included with the POST request, it must be provided in a subsequent PUT request.

The following example shows how to store a secret in the format of an AES key by submitting a POST request with the base64-encoded secret payload specified against the secrets resource.

Example: Store a secret, request

```
$ curl -X POST -H 'Content-Type: application/json' -H 'Accept: application/json'\
-H 'X-Auth-Token: '$AUTH_TOKEN -d\
  '{
     "name": "AES key",
     "expiration": "2020-02-28T19:14:44.180394",
     "algorithm": "aes",
     "bit_length": 256,
     "mode": "cbc",
     "payload": "gF6+lLoF3ohA9aPRpt+6bQ==",
     "payload_content_type": "application/octet-stream",
     "payload_content_encoding": "base64"
   }' $API_ENDPOINT/v1/secrets
```

If the request is successful, you will receive a response like the following one.

Example: Store a secret, response

```
"{"secret_ref": "https://iad.keep.api.rackspacecloud.com/v1/secrets/578391c7-92fa-484f-8546-3562b170e5"}"
```

The preceding example shows the secret ID (578391c7-92fa-484f-8546-3562b170e5), which will be returned in a successful response from the https://iad.keep.api.rackspacecloud.com endpoint.

For subsequent API calls that require a secret ID, you should set an environment variable as follows:

```
$ export SECRET_ID=578391c7-92fa-484f-8546-3562b170e5
```

Note: Note You can also store a secret by submitting a POST request without specifying the secret payload and then submitting a subsequent PUT request with the payload. This storage mode enables you to upload a binary file to Cloud Keep directly for encrypted storage. For more information, see Create and store a secret by using two requests.

Comment [KH6]: the payload?

Deleted: with

Comment [KH7]: This paragraph is part of the preceding note, but it should not be. Remove it from the note and make it a regular paragraph.

Formatted: Indent: Left: 0"

Deleted: wth

Deleted: ¶

Deleted: :

Deleted: The

Deleted: above

Deleted: Id

Deleted: endpoint

Deleted: requiring

Comment [KH8]: Delete the word *Note*.

Deleted: first

Deleted: a

Deleted: read

Deleted: -step storage

Create and store a secret by using two requests

When you cannot easily provide secret data inside the JSON data in the initial POST request, you can use a subsequent PUT request to provide the data.

1. Create the secret metadata in Cloud Keep by sending a POST request, as shown in the following example:

```
$ curl -i -X POST -H 'Content-Type: application/json' -H 'X-Auth-Token: '$AUTH_TOKEN -d \ '{"name":
"Binary Key File"}' $API_ENDPOINT/v1/secrets
```

If the call is successful, you receive a 201 Created response, as shown in the following example:

```
HTTP/1.1 201 Created
Date: Tue, 01 Mar 2016 23:04:16 GMT
Location: https://iad.keep.api.rackspacecloud.com:443/v1/secrets/e56fbf98-e670-41b4-96a1-8ed095df2345
Via: 1.1 Repose (Repose/7.3.1.0)
Date: Tue, 01 Mar 2016 23:04:16 GMT
x-trans-id:
09skcXF1ZXN0SWQiOijkms8yYTQ4MS1hOWEBABABOOEYEtYWZlOC0pl97sYzRhYTI3NmUiLCJvcml
naW4iOm51bGx9
X-NewRelic-App-Data:
0skclp8xDgoTVVBaBAYGXlwTGhE1AwE2QgNWEVlbQFtcCxY0QwgcFFUZRAQFEV1HQ0ZNUhsBGVZ
XBAUGUF9WNCJUS81UNAAMLH1cBTRMDBQFRV1JYUFUAAAgABQBV9k8jsV1FVj8=
x-openstack-request-id: req-01be5361-b1a8-4da5-bff3-683cb10f62dc
Content-Type: application/json; charset=UTF-8
Content-Length: 105
Server: Jetty(9.2.z-SNAPSHOT)

{"secret_ref": "https://iad.keep.api.rackspacecloud.com/v1/secrets/943c8f98-e980-4cc4-0da1-8ed0993bcf55"}
```

The secret metadata is now stored in Cloud Keep with a secret ID of 943c8f98-e980-4cc4-0da1-8ed0993bcf55. Note that you have only created secret metadata, not the actual secret itself. You need to remember the secret ID for the subsequent PUT request that will store the payload itself. You can export it to an environment variable, as follows:

```
$ export SECRET_ID=943c8f98-e980-4cc4-0da1-8ed0993bcf55
```

[KH: The preceding code block is not indented, which causes the numbering below to restart.]

Comment [KH9]: Note that I have moved this subsection from the end of the section to here. I think that it makes more sense to provide all the information about storing secrets together.

Deleted: ~~-step storage~~

Deleted: ~~You can use a two-step secret storage process when secret data cannot be easily provided inside the JSON data in a one-step secret storage~~

Deleted: ~~To follow the two-step process, first c~~

Deleted: ~~successfull~~

Deleted: ~~which~~

1. Create a file with random data to be used as a secret key file. The following command creates a 5 KB file in the current directory that contains random data:

```
$ dd if=/dev/random of=secret_key_file count=5 bs=1024
```

2. Submit a PUT request that includes the secret key file that you just created, as shown in the following example:

```
$ curl -i -X PUT -H 'Content-Type: application/octet-stream'\
    -H 'X-Auth-Token: '$AUTH_TOKEN \
    -T ./secret_key_file $API_ENDPOINT/v1/secrets/$SECRET_ID
```

3. Cloud Keep encrypts and stores the contents of the secret key file, associates it with the previously created metadata, and responds with an empty `204 No Content` message, as shown in the following example:

```
HTTP/1.1 204 No Content
Date: Tue, 01 Mar 2016 23:13:10 GMT
Via: 1.1 Repose (Repose/7.3.1.0)
Date: Tue, 01 Mar 2016 23:13:10 GMT
x-trans-id:
ekdmc8F1ZXN0SWQiOiIzMTMyNTQ4ZS00NDA1LTQ2OTgtOTYzOS0093jcmksz5DA1ZTYiLCJvcmlnaW
4iOm51bGx9
X-NewRelic-App-Data:
kjm83ghzn0oTVVBaBAYGXlwTGhE1AwE2QgNWEVlbQFtcCxY0QwgcFFUZRAQFEV1HQ0sCWlYIB15c
VBtXUFFaTwRXCgQVWgdWAkhbB1QABFBdUwcEUFMaHwBIUUwFAQFRXAUGA1tfUFEEVQlUABQ
BAwFVFUMEBFBaVgMAWVBQDQQAVVJTFR1RBwhCU24=
x-openstack-request-id: req-c90c5678-c3df-9279-a94c-94c9f5c062e3
Server: Jetty(9.2.z-SNAPSHOT)
```

Now you can use a GET request to retrieve the secret, as explained in Retrieve a secret.

Retrieve a secret

After you have created and stored a secret, you can submit a GET request to retrieve either the secret metadata or the actual decrypted secret, depending on the URL that is used in the GET request.

- To retrieve only the secret metadata, submit the request to the `/v1/secrets/$SECRET_ID` resource.
- To retrieve the decrypted secret, submit the request to the `/v1/secrets/$SECRET_ID/payload` resource.

Deleted: Next, c

Deleted: 5KB

Deleted: in the current directory:

Deleted: Next, s

Comment [KH10]: This should not be formatted as a step. Instead, it should be a paragraph under the preceding step.

The following example retrieves the secret metadata by submitting a GET request against the endpoint URL with the secret ID specified.

Example: Retrieve secret metadata, request

```
$ curl -X GET $API_ENDPOINT/v1/secrets/$SECRET_ID  \
    -H "X-Auth-Token: $AUTH_TOKEN" | python -m json.tool
```

If the call is successful, the response looks like the following example, assuming that your API_ENDPOINT is https://iad.keep.api.rackspacecloud.com. :

Example: Retrieve secret metadata, response

```
{
  "algorithm": "aes",
  "bit_length": 256,
  "content_types": {
    "default": "application/octet-stream"
  },
  "created": "2016-02-29T19:25:31.993225",
  "creator_id": "9a756651fa1046c983d8afaefd4f0c71",
  "expiration": "2020-02-28T19:14:44.180394",
  "mode": "cbc",
  "name": "AES key",
  "secret_ref": "https://iad.keep.api.rackspacecloud.com/v1/secrets/578391c7-
92fa-484f-8546-3562b170e5",
  "secret_type": "opaque",
  "status": "ACTIVE",
  "updated": "2016-02-29T19:25:31.999733"
}
```

The following example shows how to retrieve the secret payload by submitting a GET request against the endpoint URL with the secret ID specified.

Example: Retrieve decrypted secret, request

```
$ curl -X GET $API_ENDPOINT/v1/secrets/$SECRET_ID/payload \
    -H "X-Auth-Token: $AUTH_TOKEN"
```

Deleted: ¶
Example: Retrieve secret metadata request¶

Moved down [1]: Example: Retrieve decrypted secret request¶

Moved (insertion) [1]

If the call is successful, you receive a response that contains the decrypted secret.

Deleted: containing

Retrieve a list of stored secrets

Perform a GET request on the secrets resource to retrieve a list of secrets that are associated with your tenant.

Comment [KH11]: This term is not linked in any of the preceding sections. I would say that if you want to link to the concept section, do it at the first occurrence of the term in the beginning of this section, rather than here.

By default, the API request returns the first 11 secrets associated with the tenant. You can use the `limit` and `offset` request parameters to specify the number of secrets returned in a single request and to set the starting point for the list.

Comment [KH12]: In the "Get secrets" section of the API Reference, the description of the `limit` parameter says that the default is 10. That seems more likely to be the correct number!

Formatted: code_body Char, Font color: Auto, Border: : (No border)

Formatted: code_body Char, Font color: Auto, Border: : (No border)

Deleted: limit

The following example specifies limit and offset values to return five secrets, starting with the first.

Example: Retrieve list of secrets, request

```
$ curl -X GET $API_ENDPOINT/v1/secrets?limit=5\&offset=0 \
    -H "Accept: application/json" \
    -H "X-Auth-Token: $AUTH_TOKEN" \
    -H "Content-Type: application/json" \
    | python -m json.tool
```

If the operation is successful, the response returns a list of secrets as shown in the following example.

Note: If additional secrets have been stored, the returned data contains next and previous links so that you can page through the data.

Example: Retrieve list of secrets, response

```
{
"next": "https://iad.keep.api.rackspacecloud.com/v1/secrets?limit=5&offset=5",
"secrets": [
  {
    "algorithm": "aes",
    "bit_length": 256,
    "content_types": {
      "default": "application/octet-stream"
    },
    "created": "2016-02-29T19:25:31.993225",
    "creator_id": "9a756651fa1046c983d8afaefd4f0c71",
    "expiration": "2020-02-28T19:14:44.180394",
    "mode": "cbc",
    "name": "AES key",
    "secret_ref": "https://iad.keep.api.rackspacecloud.com/v1/secrets/578391c7-
92fa-484f-8546-3562b170e5",
    "secret_type": "opaque",
    "status": "ACTIVE",
    "updated": "2016-02-29T19:25:31.999733"
  },
  {
    "algorithm": "aes",
    "bit_length": 256,
    "content_types": {
      "default": "application/octet-stream"
    },
    "created": "2016-02-29T19:23:44.974085",
    "creator_id": "9a756651fa1046c983d8afaefd4f0c71",
    "expiration": "2020-02-28T19:14:44.180394",
    "mode": "cbc",
    "name": "AES key",
    "secret_ref": "https://iad.keep.api.rackspacecloud.com/v1/secrets/94f88434-
512c-4b4b-be3e-9469fc6824e2",
    "secret_type": "opaque",
    "status": "ACTIVE",
    "updated": "2016-02-29T19:23:44.981439"
  },
  {
    "algorithm": "aes",
    "bit_length": 256,
    "content_types": {
      "default": "application/octet-stream"
    },
    "created": "2016-02-29T19:18:15.639569",
    "creator_id": "9a756651fa1046c983d8afaefd4f0c71",
    "expiration": "2020-02-28T19:14:44.180394",
    "mode": "cbc",
    "name": "AES key",
    "secret_ref": "https://iad.keep.api.rackspacecloud.com/v1/secrets/98bccb6e-
67c6-87a9-b7a4-98198b4f1732",
    "secret_type": "opaque",
    "status": "ACTIVE",
    "updated": "2016-02-29T19:18:15.648516"
  },
  {
    "algorithm": "aes",
    "bit_length": 256,
    "content_types": {
      "default": "application/octet-stream"
    },
    "created": "2016-02-29T19:13:37.888706",
    "creator_id": "9a756651fa1046c983d8afaefd4f0c71",
    "expiration": "2018-02-28T19:14:44.180394",
    "mode": "cbc",
    "name": "AES key",
    "secret_ref": "https://iad.keep.api.rackspacecloud.com/v1/secrets/ba87c04f-
797c-4f69-8fb6-d01db61d9573",
    "secret_type": "opaque",
    "status": "ACTIVE",
    "updated": "2016-02-29T19:13:37.896416"
  },
  {
    "algorithm": "aes",
    "bit_length": 256,
    "content_types": {
      "default": "application/octet-stream"
    },
    "created": "2016-02-29T15:54:34.474305",
    "creator_id": "9a756651fa1046c983d8afaefd4f0c71",
    "expiration": "2018-02-28T19:14:44.180394",
    "mode": "cbc",
    "name": "AES key",
    "secret_ref": "https://iad.keep.api.rackspacecloud.com/v1/secrets/19f7f7aa-
817c-49cc-b252-441df7b44a4c",
    "secret_type": "opaque",
    "status": "ACTIVE",
    "updated": "2016-02-29T15:54:34.485707"
  }
],
"total": 23
}
```

Developer Guide

This guide is intended to assist software developers who want to develop applications by using the REST application programming interface (API) for the Rackspace Cloud Keep service.

To use the information provided here, you should have a general understanding of the Rackspace Cloud Keep service and have a Rackspace Cloud account with access to the Cloud Keep service. You should also be familiar with the following technologies:

- *RESTful* web services
- *HTTP*/1.1
- JSON data serialization format

Comment [KH13]: Do not italicize.

Comment [KH14]: Do not italicize.

Concepts

Review the following key concepts and architectural overview to learn how Rackspace Cloud Keep enables secure life-cycle management for keys and credentials.

Secret

Deleted: s

A *secret* is a single item that is stored within Rackspace Cloud Keep. A secret is any data that requires security-conscious storage, such as a key, credential, or configuration file. The typical use case for a secret is an encryption key that you want to keep safe.

Formatted: Font: Italic

Deleted: singular

Deleted: It

Deleted: , etc

Deleted: wish

Deleted: store away from prying eyes

Following are some examples of a secret:

Comment [KH15]: This look like it is formatted as a definition list, but it should not be. Just make this a regular paragraph.

Deleted: S

Deleted: may include

- Private key
- Certificate
- Password
- SSH key

Deleted: Key

Deleted: Keys

The secret schema represents the actual data that is presented to the Rackspace Cloud Keep service. Secrets themselves can be any format.

Deleted: secret or key

The following example shows the specification for a secret that has been added to Rackspace Cloud Keep:

```
{
  "uuid": "e2b633c7-fda5-4be8-b42c-9a2c9280284d",
  "name": "AES key",
  "expiration": "2018-02-28T19:14:44.180394",
  "secret": "b7990b786ee9659b43e6b1cd6136de07d9c5…",
  "secret_type": "application/aes-256-cbc",
 }
```

A secret consists of the following elements:

Element	Description
`uuid`	Unique identifier for the secret. This value is assigned by the API.
`name`	Human-readable name for the secret.
`expiration`	The expiration date for the secret in ISO 8601 format. After the secret expires, it is no longer returned by the API or agent.
`secret`	The base64-encoded value of the secret.
`secret_type`	*(optional)* The secret type. The possible secret types are as follows: • symmetric: Used for storing byte arrays such as keys suitable for symmetric encryption • public: Used for storing the public key of an asymmetric key pair • private: Used for storing the private key of an asymmetric key pair • passphrase: Used for storing plain-text passphrases • certificate: Used for storing cryptographic certificates such as X.509 certificates

Deleted: -

Deleted: Once

Deleted: has

Deleted: d

Deleted: will

Deleted: be

Comment [KH16]: I'm a little confused by the types listed here. In the examples shown earlier, the secret_type values don't match what is shown here. Are these just general type values, and the actual values will be specific examples of these general types? I guess I would just want it to be clear that these are not actual values.

Formatted: Font: Italic

Deleted: .

Deleted: keypair

Deleted: .

Deleted: keypair

Deleted: .

Deleted: .

Deleted: .

You can use one of the following methods to store a secret:

- Submit a POST request against the secrets resource. Include the secret metadata in the JSON body and include the secret itself in the payload parameter.

- Submit a POST request without a payload parameter against the secrets resource and then include the payload in a subsequent PUT request. This mode enables you to upload a binary file to the Rackspace

Cloud Keep database directly for encrypted storage.

Note: Note Submitting a POST request creates secret metadata. If the payload is provided with the POST request, then it is encrypted and stored, and then linked with this metadata. If no payload is included with the POST request, it must be provided with a subsequent PUT request. The secret resource encrypts and stores client-provided secret information and metadata.

Comment [KH17]: Delete the word *Note*.

Comment [KH18]: I don't think you need this note here. It is provided in the section about storing secrets.

Container

The containers resource simplifies secrets management in environments that have large numbers of secrets.

Deleted: is the organizational center piece of Rackspace Cloud Keep that

A *container* is a logical object that you can use to store reference links to secrets resources that are related by relationship or type. For example, you can create a single container to group secrets for a private key, certificate, and bundle for an SSL certificate. Containers simplify the task of managing large numbers of secrets resources.

Formatted: Font: Italic

Deleted: that can be used

Deleted: references

Rackspace Cloud Keep supports the following types of containers:

- Generic
- Certificate
- RSA

Comment [KH19]: This look like it is formatted as a definition list, but it should not be. Just make this a regular paragraph.

Deleted: 3

Each of these types have explicit restrictions as to what type of secrets should be held within. These will be broken down in their respective sections.

Comment [KH20]: This paragraph isn't really necessary, and it is misleading because generic containers don't have explicit restrictions. Plus. the subsequent sections describe any restrictions.

This guide assumes that you are running Rackspace Cloud Keep in a local development environment.

Comment [KH21]: It is not clear to me why this information is here. What does running Cloud Keep in a local development environment have to do with containers? Make the connection more explicit. Also, do we expect this only in regards to containers? Or should this statement be made earlier in the doc?

Generic containers

Deleted: Containers

A generic container is used to hold any type or number of secrets. There are no restrictions on the type or number of secrets that can be held within a generic container.

Deleted: for any type of container that a user may wish to create.

Deleted: amount

An example use case for a generic container is storing multiple passwords in the same container reference, as shown in the following example:

```
{
  "type": "generic",
  "status": "ACTIVE",
  "name": "Test Environment User Passwords",
  "consumers": [],
  "container_ref": "https://{cloudkeep_host}/v1/containers/{container_uuid}",
  "secret_refs": [
    {
      "name": "test_admin_user",
      "secret_ref": "https://{cloudkeep_host}/v1/secrets/{secret1_uuid}"
    },
    {
      "name": "test_audit_user",
      "secret_ref": "https://{cloudkeep_host}/v1/secrets/{secret2_uuid}"
    }
  ],
  "created": "2015-03-30T21:10:45.417835",
  "updated": "2015-03-30T21:10:45.417835"
}
```

Deleted: of a

Deleted: would be

Deleted: having

Deleted: stored

Certificate containers

Deleted: Containers

A certificate container is used to store the following secrets that are relevant to certificates:

Deleted: for storing

- certificate
- private_key (optional)
- private_key_passphrase (optional)
- intermediates (optional)

Comment [KH22]: The implication is that the container can hold any of the secrets in this list, not that it must. Given that, the (optional) qualifiers do not seem necessary.

```
{
  "type": "certificate",
  "status": "ACTIVE",
  "name": "Example.com Certificates",
  "consumers": [],
  "container_ref": "https://{cloudkeep_host}/v1/containers/{container_uuid}",
  "secret_refs": [
    {
      "name": "certificate",
      "secret_ref": "https://{cloudkeep_host}/v1/secrets/{cert_uuid}"
    },
    {
      "name": "private_key",
      "secret_ref": "https://{cloudkeep_host}/v1/secrets/{pk_uuid}"
    },
    {
      "name": "private_key_passphrase",
      "secret_ref": "https://{cloudkeep_host}/v1/secrets/{pass_uuid}"
    },
    {
      "name": "intermediates",
      "secret_ref": "https://{cloudkeep_host}/v1/secrets/{inters_uuid}"
    }

  ],
  "created": "2015-03-30T21:10:45.417835",
  "updated": "2015-03-30T21:10:45.417835"
}
```

The payload for the secret referenced as the `certificate` is expected to be a PEM formatted x509 certificate.

Deleted: "

Deleted: "

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

The payload for the secret referenced as the `intermediates` is expected to be a PEM formatted PKCS7 certificate chain.

Deleted: "

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: "

RSA containers

Deleted: Containers

An RSA container is used to store RSA public keys, private keys, and private key pass phrases.

Deleted: for storing

```
{
  "type": "rsa",
  "status": "ACTIVE",
  "name": "John Smith RSA",
  "consumers": [],
  "container_ref": "https://{cloudkeep_host}/v1/containers/{container_uuid}",
  "secret_refs": [
    {
      "name": "private_key",
      "secret_ref": "https://{cloudkeep_host}/v1/secrets/{pk_uuid}"
    },
    {
      "name": "private_key_passphrase",
      "secret_ref": "https://{cloudkeep_host}/v1/secrets/{pass_uuid}"
    },
    {
      "name": "public_key",
      "secret_ref": "https://{cloudkeep_host}/v1/secrets/{pubkey_uuid}"
    }

  ],
  "created": "2015-03-30T21:10:45.417835",
  "updated": "2015-03-30T21:10:45.417835"
}
```

Quotas

All users authenticated with Rackspace Cloud Keep are able to read the effective quota values that apply to their project. Rackspace Cloud Keep can derive the project that a user belongs to by reading the project scope from the authentication token.

Service administrators can read, set, and delete quota configurations for each project known to Rackspace Cloud Keep. These operations are available to an authenticated user that has the service administrator role. This role is defined in the Rackspace Cloud Keep `policy.json` configuration file. The name for a service administrator role is `keep:service-admin`.

Quotas can be enforced for the following Rackspace Cloud Keep resources: secrets, containers, and consumers. The configured quota value can be as follows:

- -1, which means there is no limit to the number of resources that you can create
- 0, which means that the resource has been disabled
- A positive integer, which defines the maximum number of a resource that is allowed for a project

If no value is specified, Rackspace Cloud Keep uses the default setting (see the following section), and the quota value is set to None.

Default quotas

When no quotas have been set for a project, the default quotas are enforced for that project. Default quotas are specified in the Rackspace Cloud Keep configuration file (`barbican.conf`). The defaults provided in the standard configuration file are as follows:

```
# default number of secrets allowed per project
```

Comment [KH23]: I think that it might be better to start this section with the paragraph that begins, "Quotas can be enforced...," followed by the bullet list and the last paragraph. Then, the paragraphs that are now first and second would be last. This way, quota is defined first, then the user is told how to find them.

Comment [KH24]: I know that *project* is a main idea in OpenStack, but it is not that widely used as a concept in Rackspace Cloud products. I think that it should probably be defined here, to provide some context. Or,

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: ¶

Deleted: "

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: "

Deleted: ing

Deleted: meaning

Deleted: is

Deleted: positive

Deleted: defining

Comment [KH25]: Is this the actual value that is shown? If so, show it in monospace.

Deleted: Quotas

Deleted: project

Deleted: project

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: .

```
quota_secrets = -1

# default number of containers allowed per project
quota_containers = -1

# default number of consumers allowed per project
quota_consumers = -1
```

The default quotas are returned via a GET request on the quotas resource when no explicit project quotas have been set for the current project.

Consumer

A *consumer* is registered as an interested party for a container. You can view all of the registered consumers of a container by performing a GET request on the `{container_ref}/consumers` resource. Before a container is deleted, all consumers should be notified of the deletion.

Deleted: a way to to
Formatted: Font: Italic
Deleted: A
Deleted: register
Deleted: can be viewed
Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto
Deleted: delete
Deleted: Information

General API information

The Rackspace Cloud Keep API is defined as a RESTful HTTP service that uses all aspects of the HTTP protocol, including methods, URIs, media and content types, and response codes. Review the topics in this section to learn more about these API components and how to access and use the API for this service.

Deleted: /

The Rackspace Cloud Keep API supports JSON data serialization request and response formats.

Note: All requests to authenticate against and operate the service are performed using SSL over HTTP (HTTPS) on TCP port 443.

Authentication

Each REST request to the Rackspace Cloud Keep API service requires the inclusion of a specific authorization token, supplied in the X-Auth-Token HTTP header of each API request. You get a token by submitting an authentication request with valid account credentials to the following Rackspace Cloud Identity API service endpoint:

```
https://identity.api.rackspacecloud.com/v2.0
```

For details, see the following information:

- Authenticate to the Rackspace Cloud
- Rackspace Cloud Identity API developer documentation

Role-based access control

Like many other services, the Rackspace Cloud Keep service protects its APIs by enforcing policy rules that are defined in a policy file. The Cloud Keep service stores a reference to a policy JSON file in its configuration file, /etc/barbican/barbican.conf. Typically, this file is named policy.json and is stored in /etc/barbican/policy.json.

Each Rackspace Cloud Keep API operation has a line in the policy file that dictates which level of access applies, as shown in the following example:

```
API_NAME: RULE_STATEMENT or MATCH_STATEMENT
```

The RULE_STATEMENT``value can be another ``RULE_STATEMENT or a MATCH_STATEMENT as shown in the following example:

```
RULE_STATEMENT: RULE_STATEMENT or MATCH_STATEMENT
```

The MATCH_STATEMENT parameter specifies a set of identifiers that must match between the token provided by the user of the API and the parameters or target entities of the API in question. For example, the following code sample indicates that to create a new secret via a **POST** request, you must have either the admin or creator role in your token.

```
"secrets:post": "role:admin or role:creator"
```

Warning: The Rackspace Cloud Keep service scopes the ownership of a secret at the project level. This means that many operations in the API perform an additional check to ensure that the project ID of the token matches the project ID stored as the secret owner.

Comment [KH26]: In other Cloud service API docs, this section provides information about the roles that are available for users of the service. Even if Cloud Keep is not enabled with the Rackspace RBAC, should the roles that are mentioned (role:admin and role:creator) be explained?

Oh, I see that roles are defined in the next section! Hm. I think that it would make more sense for this section to *follow* the "Default policy" section. That way the idea of a policy, and the different roles, are defined before they are referred to here.

Deleted: B

Deleted: Access

Deleted: Control

Deleted: supports the protection of

Deleted: Rackspace

Deleted: call

Comment [KH27]: API_OPERATION ?

Comment [KH28]: Insert a space between *STATEMENT* and *value* to fix the formatting here.

Deleted: .

Deleted: caller

Formatted: Font: Bold

Deleted: calls

Deleted: will

Deleted: _id

Deleted: _id

Policy

The Rackspace Cloud Keep service provides a sample `policy.json` file that you can use as the starting point for a customized policy that meets your particular needs. The sample policy defines the following distinct roles:

key-manager:service-admin

The cloud administrator in charge of the Rackspace Cloud Keep service. This user has access to all management APIs, such as the project quotas.

admin

Project administrator. This user has full access to all resources owned by the project for which the admin role is scoped.

creator

Users with this role are allowed to create new resources but are not allowed to delete any existing resources. They are also allowed full access to existing secrets owned by the project in scope.

observer

Users with this role are allowed to access existing resources but are not allowed to upload new secrets or delete existing secrets.

audit

Users with this role are allowed access only to the resource metadata. Users with this role cannot decrypt secrets.

Regionalized service access endpoints

The Rackspace Cloud Keep service is a regionalized service. The user of the service is therefore responsible for selecting the appropriate regional endpoint to ensure access to servers, networks, or other Cloud services.

Tip: To help you decide which regionalized endpoint to use, read about special considerations for choosing a data center.

Deleted: Default Policy

Deleted: The policy engine in OpenStack is very flexible and allows for customized policies that make sense for your particular cloud.

Deleted: comes with

Deleted: which

Deleted: be used

Deleted: 5

Deleted: like

Deleted: to

Deleted: are unable to

Comment [KH29]: Make this the link text.

Deleted: at http://www.rackspace.com/knowledge_center/article/about-regions

If you are working with cloud services that are in one of the Rackspace data centers, using the ServiceNet endpoint in the same data center has no network costs and provides a faster connection. ServiceNet is the data center internet network. In the service catalog that Rackspace Cloud Identity returns in the authentication response, this endpoint is listed as `internalURL`.

If you are working with services that are not in one of the Rackspace data centers, you must use a public endpoint to connect. In your authentication response, public endpoints are listed as `publicURL`. If you are working with services in multiple data centers or have a mixed environment where you have services both in your data centers and in Rackspace data centers, use a public endpoint because it is accessible from all the servers in the different environments.

Note: Copy the base URLs directly from the catalog rather than trying to construct them manually.

The regional endpoints in the service catalog contain your account ID. Your account ID, also known as project ID or tenant ID, refers to your Rackspace account number.

Tip: If you do not know your account ID or which data center you are working in, you can find that information in your Cloud Control Panel at mycloud.rackspace.com.

Rackspace Cloud Keep contract version

The Rackspace Cloud Keep version defines the contract and build information for the API.

The contract version denotes the data model and behavior that the API supports. The requested contract version is included in all request URLs.

Example: Request URL

```
https://iad.keep.api.rackspacecloud.com/v1/secrets
```

Different contract versions of the API might be available at any given time and are not guaranteed to be compatible with one another.

Deleted: your

Deleted: service catalog

Deleted: it

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: You should c

Deleted: Rackspace Cloud Identity returns a

Deleted: service catalog, which includes

Deleted: with

Moved down [2]: Different contract versions of the API might be available at any given time and are not guaranteed to be compatible with one another.

Moved down [3]: This document pertains to contract version 1.0.

Moved (insertion) [2]

This document pertains to contract version 1.0.

Moved (insertion) [3]

Request and response types

The Rackspace Cloud Keep API supports JSON data serialization formats.

Response format

Deleted: Format

Format	Accept header	Query extension	Default
JSON	application/json	.json	Yes

Deleted: Header

Deleted: Extension

Specify the request format by using the `Content-Type` header, which is required for operations that have a request body.

You can specify the response format in requests by using the `Accept` header. If no response format is specified, JSON is the default response format.

Limits

All accounts are configured with default thresholds, or limits, that manage capacity and prevent abuse of the system.

Rate limits control the frequency at which the user can issue specific API requests.

For Rackspace Cloud Keep, the rate limit is 600 requests per minute.

Note: If the default limits are too low for your particular application, contact Rackspace Cloud support to request an increase. All requests require reasonable justification.

Common headers

The following table describes the common headers used by the API.

Common headers

Header	Description
X-Auth-Token	Authentication token. Required.
X-Project-Id	A unique ID for the user to which the value of X-Auth-Token grants access. The unique ID is your account ID.
Accept	Media type. Initially, only text and binary types are supported.
Accept-Encoding	Specifies that the agent accepts gzip-encoded response bodies.
Content-Length	For POST or PUT requests, the length in bytes of the message document being submitted.
Content-Type	application/json
Date	Current date and time.
Host	Host name of the API.

Date and time format

For the display and consumption of date and time values, Rackspace Cloud services use a date format that complies with ISO 8601.

```
YYYY-MM-DDThh:mm:ssZ
```

The system time is expressed as UTC. For example, the UTC-5 format for May 19, 2016 at 8:07:08 a.m. is as follows:

```
2016-05-19T08:07:08-05:00
```

Deleted: Table:

Comment [KH30]: What does "Initially" mean here? The first time you use the API? For the EA release? Can we be more specific?

Formatted: Space After: 18 pt

Deleted: ¶
The system time is expressed as UTC.¶
Example: |product name| date and time format

Deleted: yyyy

Deleted: dd

Deleted: HH

The following table describes the date and time format codes.

Date and time format codes

YYYY	Four-digit year
MM	Two-digit month
DD	Two-digit day
T	Separator for the date and time
hh	Two-digit hour (00-23)
mm	Two-digit minute
ss	Two-digit second
Z	RFC 822 time zone (offset from GMT). If Z is not replaced with the offset from GMT, it indicates a 00:00 offset.

Comment [KH31]: This should not be the header row. Insert a header row.

Deleted: yyyy

Deleted: /

Deleted: HH

Deleted: 8601

Deleted: timezone

Response codes

The Rackspace Cloud Keep REST API returns an HTTP response code that denotes the success or failure of the operation.

Deleted: call

- Successful response codes are returned only if all configured providers were successful in processing the request.
- Error response codes are accompanied by an application/json response body that contains the error messages.

The following table lists possible responses with their associated codes and descriptions.

List of common response codes

Response	Associated response code	Description
OK	200	The request to retrieve a resource was successful.
Created	201	The request to create a resource was successful.
Accepted	202	The request was accepted for asynchronous processing.
No Content	204	The request was successful, and the service did not return any content. For example, a successful DELETE operation returns this code.
Bad Request	400	The request was not processed because of an error in the input values. Recheck the parameters to the service and resubmit.
Unauthorized	401	The request was not processed because of an authentication failure. This response is often the result of a bad user ID or password.
Forbidden	403	The request was not processed because of an authorization failure. Check that your service catalog contains the Cloud Keep endpoint and that your user has the correct role-based access control (RBAC) roles.
Service Unavailable	503	The service is currently unavailable. For example, it might be offline for scheduled platform maintenance. Try again later.

Example: Error message

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "title": "Unsupported limit",
```

Deleted: Table:

Deleted: has been

Deleted: has

Deleted: call

Deleted: will

Deleted: due to

Deleted: due to

Deleted: userid

Deleted: due to

Deleted: RBAC

Deleted: down

Deleted: example

```
  "description": "The given limit cannot be negative, and cannot be greater than 50.",
  "code": 1092,
  "link": {
    "rel": "help",
    "href": "http://docs.example.com/messages#limit",
    "text": "API documentation for the limit parameter"
  }
}
```

API Reference

Learn about the available Rackspace Cloud Keep API resources and methods and see request and response examples.

Note: This document refers to version **v1** of the API. When you submit a request to the Rackspace Cloud Keep API, append the version number to the API endpoint in the URI, for example, $ENDPOINT/v1/containers/{containerID}/consumers. In the examples, replace any `{version}` placeholder values with v1.

Secrets API operations

A secret is any data that requires security-conscious storage, for example, an encryption key, credential, or configuration file. You can use the Secrets API operations to store, view, and manage secrets data in a Rackspace Cloud Keep project.

Note: The examples use the `$ENDPOINT` and `$AUTH_TOKEN` environment variables to specify the API endpoint and authentication token values for accessing the service. Before you run the code samples, be sure to configure your environment variables.

Retrieve secrets

```
GET /{version}/secrets
```

This operation retrieves all of the secrets for a given tenant.

The following table shows the possible response codes for this operation.

Response code	Name	Description
200	OK	This status code is returned when the secrets have been successfully retrieved for the tenant.
401	Unauthorized	This status code is returned when the user was not successfully

Formatted: Font: Bold

Deleted:

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto, Border: : (No border)

Deleted: Operations

Deleted: se

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: Make

Deleted: you

Deleted: C

Deleted: before running the code samples

Deleted: Get

Deleted: Secrets

Deleted: method

Deleted: :

Deleted: Code

Deleted: succesfully

Response code	Name	Description
		authenticated.
403	Forbidden	This status code is returned when the user does not have the correct RBAC role or roles

Request

The following table shows the URI parameters for the request.

Name	Type	Description
offset	integer	The starting index within the total list of the secrets that you want to retrieve.
limit	integer	The maximum number of secrets to return (up to 100). The default limit is 10.

This operation does not accept a request body.

Example: Retrieve secrets, cURL request

```
curl -H 'Accept: application/json' \
    -H 'X-Auth-Token: $AUTH-TOKEN' \
    $ENDPOINT/v1/secrets?offset={offset}&limit={limit}
```

Response

The following table shows the response attributes.

Name	Type	Description
secrets	list	Contains a list of dictionaries filled with secrets data
total	integer	The total number of secrets available to the user

Deleted: Code

Deleted: (s)

Deleted: :

Deleted: would like

Deleted: take

Deleted: Delete secret

Deleted: atttributes

Deleted: for the request

Deleted: :

Name	Type	Description
next	string	A HATEOAS URL to retrieve the next set of secrets based on the offset and limit parameters. This attribute is available only when the total number of secrets is greater than the values of the offset and limit parameters combined.
previous	string	A HATEOAS URL to retrieve the previous set of secrets based on the offset and limit parameters. This attribute is available only when the request offset is greater than 0.

The following response example shows the results of sending an API request where the offset is 0 and the limit is 2.

Example: Retrieve secrets, JSON response

```
{
  "secrets": [
    {
      "status": "ACTIVE",
      "secret_ref":
"https://iad.keep.api.rackspacecloud.com/v1/secrets/15108db8-4505-4c5b-96b9-
a9838951f28f",
      "updated": "2014-08-25T20:43:01.510569",
      "name": "secretname",
      "algorithm": "aes",
      "created": "2014-08-25T20:43:01.421625",
      "content_types": {
        "default": "application/octet-stream"
      },
      "mode": "cbc",
      "bit_length": 256,
      "expiration": null
    },
    {
      "status": "ACTIVE",
      "secret_ref":
"https://iad.keep.api.rackspacecloud.com/v1/secrets/485950f0-37a5-4ba4-b1d6-
413f79b849ef",
      "updated": "2014-08-25T21:18:35.821340",
      "name": "secretname",
      "algorithm": "aes",
      "created": "2014-08-25T21:18:35.719952",
```

Deleted: url

Formatted: code_body Char, Font: +Body (Calibri)

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: only

Formatted: code_body Char, Font: +Body (Calibri)

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: url

Formatted: code_body Char, Font: +Body (Calibri)

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: only

Deleted: s

Deleted: resutls

Deleted: Get

```
      "content_types": {
        "default": "application/octet-stream"
      },
      "mode": "cbc",
      "bit_length": 256,
      "expiration": null
    }
  ],
  "total": 2,
  "next":
"https://iad.keep.api.rackspacecloud.com/v1/secrets?limit=2&offset=3",
  "previous":
"https://iad.keep.api.rackspacecloud.com/v1/secrets?limit=2&offset=0"
}
```

Retrieve secret metadata

Deleted: Get

```
GET /{version}/secrets/{secret_id}
```

This operation retrieves the metadata for the specified secret.

Deleted: method

The following table shows possible response codes for this operation.

Deleted: :

Response code	Name	Description
200	Success	This status code is returned when the secret metadata has been successfully retrieved.
404	Not Found	This error code is returned when the secret ID is invalid.

Deleted: Code

Deleted: id

Request

The following table shows the URI parameter for the request.

Name	Type	Description

Name	Type	Description
{secret_id}	String	This parameter specifies the unique identifier of a secret that has been stored.

This operation does not accept a request body.

Example: Retrieve secret metadata, cURL request

```
curl -H 'Accept: application/json'
    -H 'X-Auth-Token: $AUTH-TOKEN'\
    $ENDPOINT/v1/secrets/{secret_id}
```

Response

The following table shows the response attributes.

Name	Type	Description
status	integer	Returns the current state of the secret resource.
secret_ref	URI	Returns a HATEOAS URL to retrieve information about the specified secret.
updated	date	Returns the date and time that the consumer was last updated.
name	string	Returns the name assigned to the secret resource when it was created.
algorithm	string	Returns the algorithm type used to generate the secret.
created	date	Returns the date and time that the secret was created.
content_types	dict	Returns a dictionary of content type information for the resource. Supported formats are plain text format (text/plain) and binary format (application/octet-stream). Content types are specified when the resource is created.

Formatted: Font: 11 pt

Formatted: Normal

Deleted: Get

Deleted: /secretID

Deleted: secretID

Deleted: atttributes

Deleted: for this request

Deleted: url

Deleted: the

Deleted: or

Name	Type	Description
mode	string	*(Optional)* Returns the type or mode of the algorithm associated with the secret information.
bit_length	integer	Returns the bit length of the secret resource, if available.
expiration	date	Returns the expiration date for the secret resource.

Deleted: /

Deleted: This parameter is optional.

Example: Retrieve secret metadata, JSON response

Deleted: Get

Deleted: information

```
{
  "status": "ACTIVE",
  "secret_ref": "https://iad.keep.api.rackspacecloud.com/v1/secrets/485950f0-
37a5-4ba4-b1d6-413f79b849ef",
  "updated": "2014-05-02T06:29:25.415271",
  "name": "AES key",
  "algorithm": "aes",
  "created": "2014-05-02T06:29:25.415261",
  "content_types": {
    "default": "application/octet-stream"
  },
  "mode": "cbc",
  "bit_length": 256,
  "expiration": "2014-05-28T19:14:44.180394"
}
```

Create a secret

Deleted: Secret

```
POST /{version}/secrets
```

This method creates and stores a secret.

Note: The **POST** request always creates and stores secret metadata. If a payload is provided with the **POST** request, it is encrypted and stored, and then linked with this metadata. If

Formatted: Font: Bold

Formatted: Font: Bold

no payload is provided in the request, it must be provided in a subsequent **PUT** request.

The following table shows possible response codes for this operation.

Response code	Name	Description
201	Success	This status code is returned when the secret has been successfully created.
400	Error	This error code is returned if the `payload` parameter is empty. This response indicates that the `payload` JSON attribute was provided but no value was assigned to it.
400	Error	This error code is returned if the secret has invalid data. This response might include schema violations such as mime-type not specified.
400	Error	This error code is returned if the value specified in the `payload_content_type` parameter is not supported. This error is caused when no crypto plug-in supports the `payload_content_type` requested.
413	Error	This error code is returned when the secret specified in the `payload` parameter is too large.

Request

The following table shows the body parameters for the request.

Name	Type	Description
name	string	*(Optional)* Specifies the human-readable name for the secret. If no name is supplied, the UUID is displayed for this parameter on subsequent GET calls.
expiration	integer	*(Optional)* Specifies the expiration date for the secret in ISO 8601 format. ISO 8601 formats dates by using the following representation: yyyy-mm-ddThh:mm:ss[.mmm]. For example, September 27,

Deleted: POST

Formatted: Font: Bold

Deleted: :

Deleted: Code

Deleted: "

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: "

Deleted: '

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: '

Deleted: ,

Deleted: may

Deleted: "

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: "

Deleted: It

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: "

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: "

Deleted: :

Deleted: This parameter is optional.

Deleted: will be

Comment [KH32]: In previous parameter tables, the name is not bolded. Be consistent throughout.

Deleted: String

Deleted: *(Optional)*

Deleted: -

Deleted: -

Deleted: Integer

Deleted: *(Optional)*

Name	Type	Description
		2015 is represented as 2015-09-27. After the secret has expired, it is no longer returned by the API or agent. If this parameter is not supplied, the secret has no expiration date.
algorithm	string	*(Optional)* Specifies the algorithm type used to generate the secret. Cloud Keep does not validate the information provided for this parameter because it is client and application specific.
bit_length	integer	*(Optional)* Specifies the bit length of the secret. The value must be a positive integer. Cloud Keep does not validate the information provided for this parameter because it is client and application specific.
mode	string	*(Optional)* Specifies the type or mode of the algorithm associated with the secret information. Cloud Keep does not validate the information provided for this parameter because it is client and application specific.
payload	string	*(Optional)* Specifies the secret's unencrypted plain text. If this parameter is specified, the `payload_content_type` parameter must also be specified. If this parameter is not specified, you can provide the payload information via a subsequent **PUT** request. If the payload is not provided, only the secret metadata is retrievable from Cloud Keep and any attempt to retrieve decrypted data for that secret fails. Deferring the secret information to a **PUT** request is useful for secrets that are in binary format and are not suitable for base64 encoding.
payload_content_type	string	*(Optional)* Specifies the type or format in which the secret data is provided. This parameter is required if the `payload` parameter is specified. The following values are supported:

Deleted: 2012

Deleted: 2012

Deleted: Once

Deleted: This parameter is optional.

Deleted: This parameter is optional. Barbican

Deleted: String

Deleted: *(Optional)*

Deleted: /

Deleted: Must

Deleted: This parameter is optional. Barbican

Deleted: Integer

Deleted: *(Optional)*

Deleted: /

Deleted: /

Deleted: This parameter is optional. Barbican

Deleted: String

Deleted: *(Optional)*

Deleted: /

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: as well

Formatted: Font: Bold

Deleted: will be

Deleted: Barbican

Deleted: will

Formatted: Font: Bold

Deleted: String

Deleted: *(Optional)*

Deleted: /

Deleted: in

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: String

Deleted: *(Optional)*

Name	Type	Description
		`text/plain` - This value is used to store plain text secrets. Other options are `text/plain; charset=utf-8`. If the charset value is omitted, UTF-8 is used. Note that Cloud Keep normalizes some formats before storing them as secret metadata; for example, `text/plain; charset=utf-8` is converted to `text/plain`. Retrieved metadata might not exactly match what was originally specified in the request. When the `payload_content_type` parameter is set to `text/plain`, you cannot specify a value for the `payload_content_encoding` parameter. `application/octet- stream` - This value is used to store binary secrets from a base64-encoded payload. If this value is used, you must also include the `payload_content_encoding` parameter.
payload_content_encoding	string	*(Optional)* Specifies the encoding format used to provide the payload data. Cloud Keep might translate and store the secret data into another format. This parameter is required if the `payload_content_type` parameter is set to `application/octet- stream`. The only supported value for this parameter is `base64`, which specifies base64- encoded payloads.

Example: Create a secret, JSON request

```
curl -X POST $ENDPOINT/v1/secrets -H 'Content-Type: application/json'\
    -H 'Accept: application/json -H 'X-Auth-Token: $AUTH-TOKEN' -d \
    '{
      "name": "key",
      "expiration": "2014-09-01T19:14:44.180394",
      "algorithm": "aes",
      "bit_length": 256,
      "mode": "cbc",
      "payload": "secretsecretsecret",
      "payload_content_type": "text/plain"
    }'
```

Deleted: - "
Formatted
Deleted: "
Deleted: "
Formatted
Deleted: "
Deleted: will be
Deleted: assumed
Deleted: Barbican
Deleted: ,
Deleted: "
Formatted
Deleted: "
Deleted: "
Deleted: "
Deleted: may
Formatted
Deleted: POST or PUT
Formatted
Deleted: "
Formatted
Deleted: "
Formatted
Deleted: - "
Formatted
Deleted: "
Formatted
Deleted: Barbican
Deleted: may
Formatted
Deleted: "
Formatted
Deleted: "
Deleted: "
Formatted
Deleted: "
Deleted: String
Deleted: *(Optional)*
Comment [KH33]: Earlier sections use "cURL
Deleted: Example:Create
Deleted: Secret
Deleted: :

Response

The following table shows the response attribute.

Name	Type	Description
secret_ref	URI	Returns a HATEOAS URL to retrieve information about the specified secret. The reference URL concatenates the URI for the retrieve secrets API operation and the system-generated secret ID that is assigned automatically when the secret is created. In the example, the secret ID value is 485950f0-37a5-4ba4-b1d6-413f79b849ef.

Example: Create a secret, JSON response

```
{
  "secret_ref":"https://iad.keep.api.rackspacecloud.com/v1/secrets/485950f0-37a5-4ba4-b1d6-413f79b849ef"
}
```

Update a secret

```
PUT /{version}/secrets/{secret_id}
```

This operation stores the payload for an existing secret that was created without a payload. To provide secret information after the secret is created, submit a **PUT** request to the URI that contains the secret ID of the secret that you want to update. The **PUT** request should include the payload and the appropriate `Content-Type` and `Content-Encoding` definitions.

Note: You can send a **PUT** request only once after a **POST** operation that does not include a payload. Also, you cannot modify any other attributes for a secret resource by using the **PUT** operation.

The following table shows the possible response codes for this operation.

Deleted: for this request

Deleted: url

Deleted: the

Deleted: ‘

Deleted: `

Deleted: and the

Deleted: secretID

Deleted: *secretID*

Deleted: Secret

Deleted: Secret

Deleted: method

Formatted: Font: Bold

Formatted: Font: Bold

Deleted: , as well as

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: only make

Formatted: Font: Bold

Formatted: Font: Bold

Deleted: note that

Formatted: Font: Bold

Deleted: :

Response code	Name	Description
204	No Content	This status code is returned when the secret has been successfully updated.
400	Error	This error code is returned when no crypto plug-in supports the payload content type requested in the `Content-Type` header.
400	Error	This error code is returned when no value was provided for the `payload` parameter.
404	Error	This error code is returned when the supplied UUID doesn't match a secret in the datastore for the specified tenant.
409	Error	This error code is returned when the secret already has encrypted data associated with it.
413	Error	This error code is returned when the secret specified in the `payload` parameter is too large. The current size limit is 10,000 bytes.

Request

The following table shows the URI parameters for the request.

Name	Type	Description
secret_id	string	The unique identifier of a secret that has been stored.
secret_data_file	binary	A file that contains the binary data to be stored as the secret payload.

This operation does not accept a request body.

Example: Update a secret, cURL request

Deleted: Code

Deleted: _

Deleted: _

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: "

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: "

Deleted: "

Formatted: code_body Char, Font: +Body (Calibri)

Deleted: "

Deleted: :

Deleted: {

Deleted: }

Deleted: String

Deleted: This parameter specifies t

Deleted: containing

Deleted: {

Deleted: D

Deleted: ataFile

Deleted: }

Deleted: Binary

Comment [KH34]: cURL or JSON ? Whichever one you use, be consistent throughout the whole doc. I won't mark it again.

```
curl -X PUT -H 'Content-Type: application/octet-stream' \
    -H 'X-Auth-Token: $AUTH-TOKEN' \
    -T {secret_data_file} $ENDPOINT/v1/secrets/{secret_id}
```

The -T option to cURL is used to send the contents of the specified file as the body of the request. For more information, see https://curl.haxx.se/docs/manual.html.

Response

The operation returns an HTTP 204 Accepted response code, if successful. It does not return a response body.

Delete a secret

```
DELETE /{version}/secrets/{secret_id}
```

This operation deletes the specified secret.

The following table shows the possible response codes for this operation.

Code	Description
204	Successful request
401	Invalid X-Auth-Token or the token doesn't have permissions to this resource
404	Secret not found

Request

The following table shows the URI parameter for the request.

Parameter name	Type	Description	Default

Deleted: ~~secretDataFile~~

Deleted: ~~a83018d1-e657-4957-9ddd-42a479753e6b~~

Deleted: curl

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: D

Deleted: :

Comment [KH35]: The preceding response code tables have three columns: "Response code," "Name," and "Description." All of the response code tables should have the same format.

Comment [KH36]: Preceding response code tables use sentences to describe the codes. Do the same in this table.

Deleted: :

Comment [KH37]: None of the URI parameter tables in the preceding sections have a "Default" column. Is it necessary here?

Parameter name	Type	Description	Default
secret_id	string	The UUID of the secret.	None

This operation does not accept a request body.

Example: Delete a secret, cURL request

```
curl -X DELETE -H 'X-Auth-Token: $AUTH-TOKEN' \
    $ENDPOINT/v1/secrets/{secret_id}
```

Response

The operation returns an HTTP 204 Accepted response code, if successful. It does not return a response body.

Containers API operations

A container is a logical object that you can use to store reference links to secrets resources that are related by relationship or type. For example, you can create a single container to group secrets for a private key, certificate, and bundle for an SSL certificate.

Use the Containers API operations to create and manage groups of related secrets for Rackspace Cloud Keep projects.

Note: These examples use the `$ENDPOINT` and `$AUTH_TOKEN` environment variables to specify the API endpoint and authentication token values for accessing the service. Before you run the code samples, be sure to configure your environment variables.

Create a container

```
POST /{version}/containers
```

This operation creates a container. You can create the following types of containers: generic, RSA, and certificate.

Comment [KH37]: None of the URI parameter tables in the preceding sections have a "Default" column. Is it necessary here?

Deleted: ID

Deleted: for

Deleted: doesn't take

Deleted: secretID

Deleted: Operations

Deleted: can be used

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: Make sure you

Deleted: C

Deleted: before running the code samples

Deleted: C

Deleted: ¶
There are three different types of containers that can be created:

Deleted: rsa

Generic

This type of container holds references to any type or number of secrets. Each secret reference is accompanied by a name. Unlike other container types, no specific restrictions are enforced on the contents name attribute.

RSA

This type of container is used to hold references to only the following types of secrets, which are enforced by their names: public_key, private_key, and private_key_passphrase.

Certificate

This type of container is used to hold a reference to a certificate and optionally private_key, private_key_passphrase, and intermediates.

The following table shows the possible response codes for this operation:

Code	Description
201	Successful creation of the container
401	Invalid X-Auth-Token or the token doesn’t have permissions to this resource
403	Forbidden. The user has been authenticated, but is not authorized to create a container. This can be based on the user’s role or the project’s quota.

Request

There are no URL parameters for this request.

The following table shows the body parameters for the request.

Name	Type	Description
type	string	Type of container. Possible values are generic, rsa, and certificate.

Deleted: any number of

Comment [KH38]: I want this to correspond to one of the attributes listed in the request parameters table. Would it be the `secret_refs.name` parameter?

Deleted: designed

Deleted: three different

Deleted: . These secrets are enforced by the their accompanied names

Deleted: designed

Comment [KH39]: See my comments for the table under "Delete a secret."

Deleted: the

Deleted: :

Deleted: Options:

Name	Type	Description
name	string	(*Optional*) Human-readable name for identifying your container.
secret_refs	list	A list of dictionaries that contain references to secrets.
secret_refs.name	string	The name assigned to the secret resource when it was created.
secret_refs.secret_ref	URI	A HATEOAS URL to retrieve information about the specified secret.
secret_id	string	The UUID for the secret to be added to the container.

Example: Create a container, cURL request

```
curl -X POST -H 'X-Auth-Token $AUTH-TOKEN -d \
 '{
    "type": "generic",
    "name": "container name",
    "secret_refs": [
        {
            "name": "private_key",
            "secret_ref": "$ENDPOINT/v1/secrets/{secret_id}"
        }
    ]
 }' $ENDPOINT/v1/containers
```

Response

The following table shows the response attributes.

Parameter name	Type	Description
container_ref	URI	Returns a HATEOS URL to retrieve information about the container resource.
container_id	string	The UUID value assigned to the container. In the following example, the container ID is /6ad67bc0-17fd-45ce-b84a-a9be44fe06.

Deleted: *optional*

Formatted: Font: Italic

Deleted: containing

Deleted: url

Deleted: secretID

Deleted: Container

Deleted: secretID

Deleted: atttributes

Deleted: for this request

Deleted: url

Deleted: ID

Deleted: shown belowcontainerID=

Example: Create a container, JSON response

```
{
  "container_ref":
"https://iad.keep.api.rackspacecloud.com/v1/containers/6ad67bc0-17fd-45ce-
b84a-a9be44fe069b"
}
```

Retrieve containers

```
GET /{version}/containers
```

This operation returns a list of the containers in a project. Returned containers are ordered by creation date, oldest to newest.

The following table shows the possible response codes for this operation.

Response code	Name	Description
200	OK	This status code is returned when the containers have been successfully retrieved for the tenant.
401	Unauthorized	This status code is returned when the user was not successfully authenticated.
403	Forbidden	This status code is returned when the user does not have the correct RBAC role or roles.

Request

The following table shows the URI parameters for the request.

Name	Type	Description

Deleted: Container

Deleted: Get

Deleted: Containers

Deleted: R

Deleted: 's containers

Deleted: ¶

Deleted: ;

Deleted: :

Deleted: Code

Deleted: succesfully

Deleted: (s)

Deleted: :

Name	Type	Description
offset	integer	The starting index within the total list of the containers that you want to retrieve.
limit	integer	The maximum number of containers to return (up to 100). The default limit is 10.

Deleted: would like

This operation does not accept a request body.

Example: Retrieve containers, cURL request

Deleted: Get

```
curl -H 'Accept: application/json' \
    -H 'X-Auth-Token:$AUTH-TOKEN' \
    $ENDPOINT/v1/containers?offset={offset}&limit={limit}
```

Response

The following table shows the response attributes.

Deleted: atttributes

Deleted: for the req

Deleted: uest:

Name	Type	Description
containers	list	Returns a list of dictionaries with information about each container that has been created in Cloud Keep.
total	integer	The total number of containers available to the user.
next	string	A HATEOAS URL to retrieve the next set of containers based on the offset and limit parameters. This attribute is available only when the total number of containers is greater than the offset and limit parameter values combined.
previous	string	A HATEOAS URL to retrieve the previous set of containers based on the offset and limit parameters. This attribute is available only when the request offset is greater than 0.

Deleted: url

Deleted: only

Deleted: url

Deleted: only

Example: Retrieve containers, JSON response

Deleted: Get

```
{
 "containers": [
   {
     "consumers": [],
     "container_ref":
"https://iad.keep.api.rackspacecloud.com/v1/containers/6ad67bc0-17fd-45ce-
b84a-a9be44fe069b",
     "created": "2015-03-26T21:10:45.417835",
     "name": "container name",
     "secret_refs": [
       {
         "name": "private_key",
         "secret_ref":
"https://iad.keep.api.rackspacecloud.com/v1/secrets/485950f0-37a5-4ba4-b1d6-
413f79b849ef"
       }
     ],
     "status": "ACTIVE",
     "type": "generic",
     "updated": "2015-03-26T21:10:45.417835"
   }
 ],
 "total": 1
}
```

Retrieve container information

```
GET /{version}/containers/{container_id}
```

This operation retrieves information about the specified container.

The following table shows the possible response codes for this operation.

Code	Description
200	Successful Request
401	Invalid X-Auth-Token or the token doesn’t have permissions to this resource

Deleted: Get

Deleted: Container

Deleted: Information

Deleted: method

Deleted: a

Deleted: :

Comment [KH40]: Make this like other response code tables, with the third column and descriptions in sentences.

Code	Description
404	Container not found or unavailable

Comment [KH40]: Make this like other response code tables, with the third column and descriptions in sentences.

Request

The following table shows the URI parameter for the request.

Deleted: s

Deleted: :

Name	Type	Description
container_id	string	Specifies the unique identifier of a container that has been stored.

Deleted: {

Deleted: }

Deleted: S

Deleted: *(Required)*

Deleted: This parameter s

This operation does not accept a request body.

Example: Retrieve container information, cURL request

Deleted: Get

Deleted: Container

Deleted: I

```
curl -H 'Accept: application/json' -H 'X-Auth-Token:$AUTH-TOKEN' \
  $ENDPOINT/v1/containers/{container_id}
```

Deleted: containerID

Response

The following table shows the response attributes.

Deleted: attributes

Deleted: for this request

Name	Type	Description
type	string	Indicates the container type: generic, rsa, or certificate.
status	string	Returns the current state of the container.
name	string	The name assigned to the container when it was created.
consumers	dict	Returns a list of dictionaries with information about the consumers included in the container.

Formatted: Font: Not Italic

Formatted: Font: Not Italic

Formatted: Font: Not Italic

Deleted: for

Deleted: specified

Deleted: specified

Deleted: specified

Name	Type	Description
container_ref	URI	A HATEOS URL to retrieve information about the specified container.
secret_refs	dict	Returns a dictionary with information about the secrets included in the container.
secret_refs.name	string	The name assigned to the secret resource when it was created.
secret_refs.secret_ref	URI	A HATEOAS URL to retrieve information about the specified secret.
created	date	The date and time that the container was created.
updated	date | The date and time that the container was last updated.	

Deleted: url

Deleted: url

Comment [KH41]: Some funky formatting here. . .

Example: Retrieve container information, JSON response

Deleted: Get

```
{
  "type": "generic",
  "status": "ACTIVE",
  "name": "container name",
  "consumers": [],
  "container_ref":
"https://iad.keep.api.rackspacecloud.com/v1/containers/6ad67bc0-17fd-45ce-
b84a-a9be44fe069b",
  "secret_refs": [
   {
     "name": "private_key",
     "secret_ref":
"https://iad.keep.api.rackspacecloud.com/v1/secrets/485950f0-37a5-4ba4-b1d6-
413f79b849ef"
   }
  ],
  "created": "2015-03-26T21:10:45.417835",
  "updated": "2015-03-26T21:10:45.417835"
}
```

Delete a container

```
DELETE /{version}/containers/{container_id}
```

This operation deletes a container.

Note: To prevent unexpected service issues, notify all consumers registered for the container before you delete it. Use the retrieve consumers operation to get a list of consumers.

The following table shows the possible response codes for this operation.

Code	Description
204	Successful deletion of a container
401	Invalid X-Auth-Token or the token doesn't have permissions to this resource
404	Container not found or unavailable

Request

The following table shows the URI parameters for the request.

Parameter name	Type	Description	Default
container_id	string	The UUID for the container	None

This operation does not require a response body.

Example: Delete a container, cURL request

```
curl -X DELETE -H 'X-Auth-Token: $AUTH-TOKEN' \
   $ENDPOINT/v1/containers/{container_id}
```

Deleted: D

Deleted: ensure that you

Deleted: :

Comment [KH42]: Update table to be like others.

Deleted: :

Comment [KH43]: Delete this column, unless you are adding "Default" columns to the other parameter tables.

Deleted: ID

Deleted: containerID

Response

The operation returns an HTTP 204 Accepted response code, if successful. It does not return a response body.

Consumers API operations

The consumers resource provides a method to register as an interested party for a container.

Use the Consumers API operations to view and manage registered consumers.

Note: These examples use the `$ENDPOINT` and `$AUTH_TOKEN` environment variables to specify the API endpoint and authentication token values for accessing the service. Before you run the code samples, be sure to configure your environment variables.

Retrieve consumers for a container

```
GET /{version}/containers/{container_ref}/consumers
```

This operation lists a container's consumers. The list of consumers can be filtered by the parameters passed in via the URL.

The following table shows the possible response codes for this operation.

Response Code	Name	Description
200	OK	This status code is returned when the consumers have been successfully retrieved for the tenant.
401	Unauthorized	This status code is returned when the user was not successfully authenticated.
403	Forbidden	This status code is returned when the user does not have the correct RBAC role or roles.

Deleted: Operations

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: Make sure you

Deleted: C

Deleted: before running the code samples

Deleted: Get

Deleted: 's consumers

Deleted: Lists

Deleted: ¶

Deleted: :

Deleted: succesfully

Deleted: (s)

Request

The following table shows the URI parameters for the request.

Name	Type	Description
container_id	integer	The UUID for the container for which you want to retrieve consumers.
offset	integer	*(Optional)* The starting index within the total list of the consumers that you want to retrieve.
limit	integer	*(Optional)* The maximum number of records to return (up to 100). The default limit is 10.

Example: Retrieve consumers for a container, cURL request

```
curl -H 'Accept: application/json' -H 'X-Auth-Token:$AUTH-TOKEN'\
    $ENDPOINT/v1/containers/{container_id}/consumers/?offset={offset}&limit={limit}
```

Response

The following table shows the response attributes.

Name	Type	Description
total	integer	Returns the number of consumers in the specified container.
consumers	dict	Returns a dictionary of consumer information for the specified container.
consumers.status	string	Returns the current state of the consumer.
consumers.URL	string	Returns the URL for the user or service using the container.
consumers.updated	date	The date and time that the consumer was last updated.

Deleted: :

Deleted: you would like to

Deleted: {

Deleted: ID

Deleted: }

Deleted: {

Deleted: }

Deleted: *(Optional)*

Deleted: would like

Deleted: {

Deleted: }

Deleted: *(Optional)*

Deleted: Get

Deleted: containerID

Deleted: for this request

Deleted: consumers resource

Deleted: for

Deleted: specified

Deleted: for the containers resource.

Name	Type	Description
consumers.name	string	The name of the consumer set by the user.
consumers.created	date	The date and time that the consumer was created.
consumers.next	URI	A HATEOAS URL to retrieve the next set of consumers based on the offset and limit parameters. This attribute is available only when the total number of consumers is greater than offset and limit parameter values combined.
consumers.previous	string	A HATEOAS URL to retrieve the previous set of consumers based on the offset and limit parameters. This attribute is available only when the request offset is greater than 0.

Deleted: consumers resource.

Deleted: url

Deleted: only

Deleted: url

Deleted: only

Example: Retrieve consumers for a container, JSON response

Deleted: Get

Deleted: specified

```
{
 "total": 3,
 "consumers": [
  {
    "status": "ACTIVE",
    "URL": "consumerurl",
    "updated": "2015-10-15T21:06:33.123878",
    "name": "consumername",
    "created": "2015-10-15T21:06:33.123872"
  },
  {
    "status": "ACTIVE",
    "URL": "consumerURL2",
    "updated": "2015-10-15T21:17:08.092416",
    "name": "consumername2",
    "created": "2015-10-15T21:17:08.092408"
  },
  {
    "status": "ACTIVE",
    "URL": "consumerURL3",
    "updated": "2015-10-15T21:21:29.970370",
    "name": "consumername3",
    "created": "2015-10-15T21:21:29.970365"
  }
 ]
}
```

Example: Retrieve consumers for container with offset and limit parameters, JSON response

Deleted: Get

```
{
  "total": 3,
  "next": "https://iad.keep.api.rackspacecloud.com/v1/containers/6ad67bc0-17fd-45ce-b84a-a9be44fe069b/consumers?limit=1&offset=2",
  "consumers": [
   {
     "status": "ACTIVE",
     "URL": "consumerURL2",
     "updated": "2015-10-15T21:17:08.092416",
     "name": "consumername2",
     "created": "2015-10-15T21:17:08.092408"
   }
  ],
  "previous": "https://iad.keep.api.rackspacecloud.com/v1/containers/6ad67bc0-17fd-45ce-b84a-a9be44fe069b/consumers?limit=1&offset=0"
}
```

Create a consumer

```
POST /{version}/containers/{container_id}/consumers
```

This operation creates a consumer for the specified container.

Deleted: C

The following table shows the possible response codes for this operation.

Deleted: :

Code	Description
201	Successful creation of the consumer
401	Invalid X-Auth-Token or the token doesn’t have permissions to this resource
403	Forbidden. The user has been authenticated, but is not authorized to create a consumer. This can

Comment [KH44]: See my earlier comments about response code tables.

Code	Description
	be based on the user's role or the project's quota.

Comment [KH44]: See my earlier comments about response code tables.

Deleted: the

Request

The following table shows the URI parameters for this request.

Name	Type	Description	Default
container_id	string	The UUID for the container.	None

Comment [KH45]: Delete this column, unless you are adding "Default" columns to the other parameter tables.

Deleted: Parameter n

Deleted: containerID

The following table shows the body parameters for the request.

Deleted: There are no URL parameters for this request.¶

Deleted: :

Name	Type	Description	Default
name	string	The name of the consumer set by the user.	None
URL	string	The URL for the user or service using the container.	None

Comment [KH46]: Delete this column.

Deleted: Parameter n

Deleted: url

Example: Create a consumer, cURL request

```
curl -X POST -H 'X-Auth-Token $AUTH-TOKEN' \
    -H 'Content-Type: application/json' \
    -d '{
         "name": "your consumer name",
          "URL": "{consumerURL}"
      }' \
    $ENDPOINT/v1/containers/{container_id}/consumers
```

Deleted: containerID

Response

The following table shows the response attributes.

Deleted: for this request

Name	Type	Description

Name	Type	Description
status	string	Returns the current state of the specified consumer.
updated	date	The date and time that the consumer was last updated.
name	string	The name of the container that the user is registering for.
consumers	dict	Returns a dictionary of information for the consumer resource.
consumers.URL	string	Returns the URL for the user or service using the container. In the response example, the consumer URL is https://consum.er.
consumers.name	string	The name of the consumer set by the user.

Deleted: for

Deleted: . for the containers resource

Example: Create a consumer, JSON response

Deleted: Consumer

```
{
   "status": "ACTIVE",
   "updated": "2015-10-15T17:56:18.626724",
   "name": "your container name",
   "consumers": [
      {
         "URL": "https://consum.er",
         "name": "your consumer name"
      }
}
```

Delete a consumer

```
DELETE /{version}/{container_id}/consumers/{consumer_id}
```

Deleted: ref

This operation deletes the specified consumer for the specified container.

Deleted: D

The following table shows the possible response codes for this operation.

Code	Description
204	Successful request
401	Invalid X-Auth-Token or the token doesn’t have permissions to this resource
404	Not Found

Comment [KH47]: See my preceding comments about the response code table.

Request

The following table shows the URI parameters for the request.

Deleted: :

Parameter name	Type	Description	Default
container_id	string	The UUID for the container	None
consumer_id	string	The UUID for the consumer	None

Comment [KH48]: Delete this column, unless you are adding “Default” columns to the other parameter tables.

Deleted: containerID

Deleted: consumerID

The following table shows the body parameters for the request.

Deleted: :

Parameter name	Type	Description	Default
name	string	The name of the consumer set by the user. The name must match the name that was used when the consumer was created.	None
URL	string	The URL for the user or service using the container. The URL must match the URL that was used when the consumer was created.	None

Comment [KH49]: Delete.

Deleted: url

Example: Delete a consumer, cURL request

```
curl -X DELETE -H 'X-Auth-Token: $AUTH-TOKEN' \
    -d '{"name": "consumername", "URL": "consumerURL"}' \
$ENDPOINT/v1/containers/{container_id}/consumers/{consumer_id}
```

Deleted: containerID

Deleted: consumerID

Deleted: where name and URL must match the name and URL that were used when the consumer was created.

Response

The operation returns an HTTP 204 Accepted response code, if successful. It does not return a response body.

Quota API operations

Deleted: Operations

Quotas specify resource limits for a Rackspace Cloud Keep project. Rackspace Cloud Keep provides a way to configure default quota values in the Rackspace Cloud Keep configuration file (`barbican.conf`). The default values are used unless a custom value is configured.

Deleted: to

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Use the Quotas API to retrieve a list of quotas for your project.

Note: These examples use the `$ENDPOINT` and `$AUTH_TOKEN` environment variables to specify the API endpoint and authentication token values for accessing the service. Before you run the code samples, be sure to configure your environment variables.

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

Deleted: Make sure you

Deleted: C

Deleted: before running the code samples

Retrieve project quotas

Deleted: Get

```
GET /{version}/quotas
```

This operation lists the effective quotas for the project of the requester. The project ID of the requester is derived from the authentication token provided in the `X-Auth-Token` header.

Comment [KH50]: tenant?

Deleted: Get

Deleted: id

Comment [KH51]: tenant?

Formatted: code_body Char, Font: (Default) Times New Roman, Font color: Auto

The following table shows the possible response codes for this operation.

Deleted: :

Code	Description
200	Successful Request
401	Invalid X-Auth-Token or the token doesn't have permissions to this resource

Comment [KH52]: See my earlier comments about this table.

Request

This request does not accept URI or body parameters.

Example: Retrieve project quotas, cURL request

Deleted: Get

```
curl -H 'Accept: application/json' -H 'X-Auth-Token:$AUTH_TOKEN'\
    $ENDPOINT/v1/quotas
```

Response

The following table shows the response attributes.

Deleted: for this request

Name	Type	Description
quotas	dict	Returns a dictionary with quota information.
quotas.secrets	integer	Returns the effective quota value of the current project for the secret resource.
quotas.cas	integer	Returns the effective quota value of the current project for the CAs resource.
quotas.orders	integer	Returns the effective quota value of the current project for the orders resource.
quotas.containers	integer	Returns the effective quota value of the current project for the containers resource.
quotas.consumers	integer	Returns the effective quota value of the current project for the consumers resource.

Comment [KH53]: This is not defined anywhere in the guide. What is the CAs resource?

Effective quota values are interpreted as follows:

Value	Description

Value	Description
-1	A negative value indicates that the resource is unconstrained by a quota.
0	A zero value indicates that the resource is disabled.
integer	A positive value indicates the maximum number of that resource that can be created for the current project.

Example: Retrieve project quotas, JSON response

Deleted: Get

Deleted: HTTP header and

```
{
 "quotas": {
  "secrets": 1000,
  "cas": 0,
  "orders": 0,
  "containers": 1000,
  "consumers": 1000
 }
}
```