redstone

Release 0.2.2

Mathew Odden

Feb 06, 2023

CONTENTS

1 Installation **3**

2 Modules **5**
- 2.1 `redstone` — Redstone Package . . . 5
- 2.2 `redstone.auth` — IBM Cloud IAM Authentication . . . 6
- 2.3 `redstone.client` — Redstone Service Clients . . . 7
- 2.4 `redstone.crypto` — Encrypting/Decrypting with KeyProtect . . . 13

3 Indices and tables **15**

Python Module Index **17**

Index **19**

Welcome to the redstone documentation.

CHAPTER

ONE

INSTALLATION

The redstone package is available for install from PyPA with pip:

```
$ pip install -U redstone
```

CHAPTER

TWO

MODULES

2.1 redstone — Redstone Package

The *redstone* module is used to interact with various IBM Cloud services.

Basic usage involves getting a *Session* object, which can be used to build a client for interacting with a specific service.

For example, to interact with Resource Controller, we can use the default session to get a client to talk to Resource Controller:

```
>>> import redstone
>>> session = redstone.get_default_session()
>>> rc = session.service("ResourceController")
```

The *service()* function at the top level can be used as a shortcut for accessing the default session to build clients. This is equivalent to the above:

```
>>> import redstone
>>> rc = redstone.service("ResourceController")
```

The default session is constructed lazily on first access and looks for the environment variable *IBMCLOUD_API_KEY* to use as a credential for interacting with services.

The default session can be overriden or created manually if desired. Any clients created using the default session will then use that session instead.

```
>>> import redstone
>>> redstone.DEFAULT_SESSION = redstone.Session(iam_api_key="...")
>>> rc = redstone.service("ResourceController")
```

2.1.1 Members

redstone.**DEFAULT_SESSION = None**

Holds the current default *Session* or None if no session has been built yet.

class redstone.**Session**(*region=None*, *iam_api_key=None*)

Session objects are used to create clients used to interact with various services. They hold region endpoint information and a credential object that is used by the clients for authentication. It's main purpose for sharing and caching credentials for use between multiple clients/services.

A Session can be created manually, but there is also a default session that can be accessed by using the *get_default_session()* function.

service(*service_name*, ***kwargs*)

redstone.**get_default_session**() → *Session*

Returns the current default session for building clients objects.

redstone.**service**(*service_name*, ***kwargs*)

Create and return a new client using the *DEFAULT_SESSION*.

2.2 redstone.auth — IBM Cloud IAM Authentication

This module holds functionality for authenticating with IBM Cloud services, which mainly involves getting a token from IAM and using that token while making requests to the various other services.

Most users will want to use the high level *TokenManager* which provides caching and automatic refresh for tokens.

There is also a low level *auth()* function that can be used to request a new token from IAM as needed.

2.2.1 Members

class redstone.auth.**TokenManager**(*api_key*, *iam_endpoint=None*, *use_refresh_token=False*)

TokenManager objects are a wrapper around an API key credential, which is used to request tokens when they are needed.

A TokenManager object caches tokens to minimize requests to IAM, and also will take care of requesting new tokens when the current cached one is expired.

Example usage:

```
tokman = TokenManager(api_key="my-cloud-api-key")

# get_token() will return a cached version or request one if needed
iam_token = tokman.get_token()
```

get_token() → str

Retrieve a valid, unexpired token from IAM if needed, or return a cached, unexpired token.

is_refresh_token_expired()

Use to check if the cached IAM Refresh token needs to be refreshed.

The Refresh token is a different token than the IAM token used to interact with services. A Refresh token is a longer lasting token that can be used instead of an API key or password credential to request a new IAM token. It is useful for some specific cases, where the API key or password needs to be dropped and the Refresh token can be used instead to generate IAM tokens.

Returns

bool, True if the Refresh token needs to be refreshed, False otherwise

is_token_expired()

Use to check if the cached IAM token needs to be refreshed.

Returns

bool, True if the token needs to be refreshed, False otherwise

redstone.auth.**auth**(*username=None*, *password=None*, *apikey=None*, *refresh_token=None*, *iam_endpoint=None*)

Makes a authentication request to the IAM API to retrieve an IAM token and IAM Refresh token.

Parameters

- **username** – Username
- **password** – Password
- **apikey** – IBMCloud/Bluemix API Key
- **refresh_token** – IBM IAM Refresh Token, if specified the refresh token is used to authenticate, instead of the API key
- **iam_endpoint** – base URL that can be specified to override the default IAM endpoint, if one, for example, wanted to test against their own IAM or an internal server

Returns

Response

redstone.auth.**find_space_and_org**(*bearer_token*, *org_name*, *space_name*)

redstone.auth.**get_orgs**(*bearer_token*)

redstone.auth.**get_spaces**(*bearer_token*, *spaces_path*)

redstone.auth.**inspect_token**(*token*)

redstone.auth.**main**()

2.3 redstone.client — Redstone Service Clients

This module holds the service specific client classes, as well as the BaseClient class that they extend from for shared business function.

If you wish to add or extend functionality to a client or service, this is where the concrete classes and logic are for those purposes.

2.3.1 Members

class redstone.client.**BaseClient**(*region=None*, *service_instance_id=None*, *iam_api_key=None*, *verify=True*, *endpoint_url=None*, *credentials=None*)

set_pre_send_fn(*fn*)

Sets the provided function as a hook to be called per request, just before the requests are sent.

This allows for modifying any aspects of the requests before they are sent, or for recording/logging/tracing purposes.

The function must handle its own exceptions, and must return a *requests.Request* object.

If *fn* resolves to false-y then it will not be called. This can be used to remove any hook currently set, by calling *set_pre_send_fn* with *None* type.

Example:

```
def add_header(req):
    req.headers["X-Trace-ID"] = "my-tracing-id"
    return req

client.set_pre_send_fn(add_header)
# X-Trace-ID will be set for all requests from this client object
for i in client.list_instances():
    print(i)

...
```

class redstone.client.**CIS**(**args*, ***kwargs*)

endpoint_for_region(*region*)

get_pool(*pool_id*)

names = ['cis']

pools()

update_pool(*pool*)

class redstone.client.**CISAuth**(*credentials*)

class redstone.client.**IKS**(**args*, ***kwargs*)

endpoint_for_region(*region*)

get_cluster_config(*cluster*)

Retrieve a KubeConfig that can be used with kubectl to interact with the given IKS cluster.

Returns

base64 encoded file data; can be decode and written to file, or used with the python kubernetes client to interact in the same process

get_clusters() → List[Dict]

List the current IKS clusters in a specific region.

Returns

A list of dict objects representing the cluster metadata.

get_kube_versions()

get_workers(*cluster*)

List the workers in an IKS cluster.

names = ['iks']

update_master(*cluster*, *version*)

Initate an update on the master nodes of a cluster.

update_worker(*cluster*, *worker*)

Initiate an update on a worker node.

The worker node will update to the latest revision that matches the master/API server version. i.e. If the master is at 1.16.x, the worker will update to the latest 1.16.x series.

class redstone.client.**KeyProtect**(**args*, ***kwargs*)

API Docs: https://cloud.ibm.com/apidocs/key-protect

exception KeyProtectError

static wrap(*http_error*)

cancel_dual_auth_delete(*key_id: str*)

Remove an authorization for a key with a dual authorization policy

API Docs: https://cloud.ibm.com/apidocs/key-protect#unsetkeyfordeletion

create(*name*, *payload=None*, *raw_payload=None*, *root=False*, *alias_list=None*)

create_import_token(*expiration: Optional[int] = None*, *max_allowed_retrievals: Optional[int] = None*)

Create an import token that can be used to import encrypted material as root keys.

expiration: The time in seconds from the creation of a import token that determines how long it remains valid.
The minimum value is 300 seconds (5 minutes), and the maximum value is 86400 (24 hours). The default value is 600 (10 minutes).

max_allowed_retrievals: The number of times that an import token can be retrieved within its expiration time
before it is no longer accessible. The default value is 1. The maximum value is 500.

API Docs: https://cloud.ibm.com/apidocs/key-protect#postimporttoken

create_key(*name*, *payload=None*, *raw_payload=None*, *root=False*, *alias_list=None*)

create_key_alias(*key_id: str*, *alias: str*)

Creates an alias name for a key. An alias is a user defined string that can be used in place of a normal UUID Key ID

API Docs: https://cloud.ibm.com/apidocs/key-protect#createkeyalias

create_key_ring(*key_ring_id: str*)

Create a key ring in the instance with the specified name.

API Docs: https://cloud.ibm.com/apidocs/key-protect#createkeyring

delete(*key_id*)

delete_key(*key_id*)

delete_key_alias(*key_id: str*, *alias: str*)

Deletes an alias name associated with a key

API Docs: https://cloud.ibm.com/apidocs/key-protect#deletekeyalias

delete_key_ring(*key_ring_id: str*)

Deletes a key ring from the associated instance

API Docs: https://cloud.ibm.com/apidocs/key-protect#deletekeyring

disable_key(*key_id: str*)

Disable a key.

The key will not be deleted, but it will not be active and key operations cannot be performed on a disabled key.

API Docs: https://cloud.ibm.com/apidocs/key-protect#disablekey

enable_key(*key_id: str*)

Enable a key.

Only disabled keys can be enabled. After calling this action, the key becomes active and key operations can be performed on it.

Note: This does not recover Deleted keys.

API Docs: https://cloud.ibm.com/apidocs/key-protect#enablekey

endpoint_for_region(*region*)

get(*key_id_or_alias*)

get_import_token()

Retrieves an import token associated with the current service instance. Token must be previously created by a create import token call.

API Docs: https://cloud.ibm.com/apidocs/key-protect#getimporttoken

get_instance_policies()

Retrieves a list of policies that are associated with a specified service instance

https://cloud.ibm.com/apidocs/key-protect#getinstancepolicy

get_key(*key_id_or_alias*)

get_key_policies(*key_id: str*)

Retrieves a list of policies that are associated with a specified key

API Docs: https://cloud.ibm.com/apidocs/key-protect#getpolicy

get_key_rings()

Get all key rings associated with specified instance

API Docs: https://cloud.ibm.com/apidocs/key-protect#listkeyrings

get_registrations(*key_id: Optional[str] = None*, *crn: Optional[str] = None*)

Retrieve a list of registrations

If *key_id* is None (the default) all registrations for the instance are returned, otherwise only the registrations associated with a specified root key are returned.

crn should be a str type that will be passed as the *urlEncodedResourceCRNQuery* parameter to the HTTP API. It is used to filter registration on a specific cloud resource. More information can be found in the API docs below.

API Docs:

- https://cloud.ibm.com/apidocs/key-protect#getregistrations
- https://cloud.ibm.com/apidocs/key-protect#getregistrationsallkeys

initiate_dual_auth_delete(*key_id: str*)

Authorize deletion for a key with a dual authorization policy

API Docs: https://cloud.ibm.com/apidocs/key-protect#setkeyfordeletion

keys()

list_keys()

names = ['kms']

purge_key(*key_id: str*)

Purge key method shreds all the metadata and registrations associated with a key that has been deleted. The purge operation is allowed to be performed on a key from 4 hours after its deletion

https://cloud.ibm.com/apidocs/key-protect#purgekey

restore_key(*key_id: str*)

Restore a key.

The RestoreKey method reverts a key's status from *Destroyed* to *Active*. This method cannot be used to restore a key that has been purged.

API Docs: https://cloud.ibm.com/apidocs/key-protect#restorekey

rotate(*key_id*, *payload=None*)

rotate_key(*key_id*, *payload=None*)

set_instance_allowed_ip_policy(*allowed_ip_enable: bool*, *allowed_ips: List[str]*)

Updates the allowed IP policy for the instance

API Docs: https://cloud.ibm.com/apidocs/key-protect#putinstancepolicy

set_instance_allowed_network_policy(*allowed_network_enable: bool*, *network_type: str*)

Updates the allowed network policy for the instance

network_type is a str type, and must be one of "public-and-private" or "private-only". The default is "public-and-private", but can be set to "private-only" to disable access to the instance from Internet client addresses.
API Docs: https://cloud.ibm.com/apidocs/key-protect#putinstancepolicy

set_instance_dual_auth_policy(*dual_auth_enable: bool*)

Updates the dual auth delete policy for the instance by passing the enable detail

API Docs: https://cloud.ibm.com/apidocs/key-protect#putinstancepolicy

set_instance_key_create_import_access_policy(*key_create_import_access_enable: bool*, *create_root_key: bool*, *create_standard_key: bool*, *import_root_key: bool*, *import_standard_key: bool*, *enforce_token: bool*)

Updates the key create import access policy details associated with an instance.

'key_create_import_access_enable' is boolean type, the instance policy is enabled if it's set to true

API Docs: https://cloud.ibm.com/apidocs/key-protect#putinstancepolicy

set_instance_metrics_policy(*metrics_enable: bool*)

Updates the metrics policy details associated with an instance

API Docs: https://cloud.ibm.com/apidocs/key-protect#putinstancepolicy

set_key_dual_auth_policy(*key_id: str*, *dual_auth_enable: bool*)

Updates the dual auth delete policy by passing the key ID and enable detail

API Docs: https://cloud.ibm.com/apidocs/key-protect#putpolicy

set_key_ring(*key_id: str*, *key_ring_id: str*, *new_key_ring_id: str*)

Transfers a key associated with one key ring to another key ring

https://cloud.ibm.com/apidocs/key-protect#patchkey

set_key_rotation_policy(*key_id: str*, *rotation_interval: int*)

Updates the rotation policy associated with a key by specifying key ID and rotation interval

API Docs: https://cloud.ibm.com/apidocs/key-protect#putpolicy

sync_associated_resources(*key_id: str*)

Executes the sync request which verifies and updates the resources associated with the key.

API Docs: https://cloud.ibm.com/apidocs/key-protect#syncassociatedresources

unwrap(*key_id*, *ciphertext*, *aad=None*)

wrap(*key_id*, *plaintext*, *aad=None*)

class redstone.client.**ResourceController**(**args*, ***kwargs*)

Client class for interacting with the Resource Controller service, which is used for managing service instances within the cloud account.

API Docs:

- https://console.bluemix.net/apidocs/resource-manager
- https://console.bluemix.net/apidocs/resource-controller

KEYPROTECT_PLAN_ID = 'eedd3585-90c6-4c8f-be3d-062069e99fc3'

create_instance(*name*, *plan_id*, *region=None*, *resource_group=None*)

Create/provision a service instance.

Returns
tuple of (service_GUID, service_CRN) if successful

Raises
Exception if there is an error –

delete_instance(*instance_crn*)

Delete/deprovision a service instance identified by the given CRN or UUID.

endpoint_for_region(*region*)

get_default_resource_group()

get_instance(*instance_id*)

list_instances()

Retrieve a list of all service and resource instances in the current account.

Note this will return an iterator that will handle the underlying pagination of large sets of instances returned.

Returns
a generator type that iterates over the collection of instances returned from the API request

names = ['rc']

resource_groups()

class redstone.client.**Session**

prepare_request(*request*)

Constructs a `PreparedRequest` for transmission and returns it. The `PreparedRequest` has settings merged from the `Request` instance and those of the *`Session`*.

Parameters
request – Request instance to prepare with this session's settings.

Return type
requests.PreparedRequest

class redstone.client.**TokenAuth**(*credentials*)

2.4 redstone.crypto — Encrypting/Decrypting with KeyProtect

2.4.1 Members

redstone.crypto.**decrypt**(*source: bytes*, *session: Optional[*Session*] = None*) → Tuple[bytes, MessageHeader]

Decrypt data previously encrypted with the encrypt function.

redstone.crypto.**encrypt**(*source: bytes*, *key_crns: List[str]*, *aad: Optional[str] = None*, *session: Optional[*Session*] = None*) → Tuple[bytes, MessageHeader]

Encrypt byte data using a given set of keys from KeyProtect.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

redstone, 5
redstone.auth, 6
redstone.client, 7
redstone.crypto, 13

INDEX

A

auth() (*in module redstone.auth*), 6

B

BaseClient (*class in redstone.client*), 7

C

cancel_dual_auth_delete() (*redstone.client.KeyProtect method*), 9
CIS (*class in redstone.client*), 8
CISAuth (*class in redstone.client*), 8
create() (*redstone.client.KeyProtect method*), 9
create_import_token() (*redstone.client.KeyProtect method*), 9
create_instance() (*redstone.client.ResourceController method*), 12
create_key() (*redstone.client.KeyProtect method*), 9
create_key_alias() (*redstone.client.KeyProtect method*), 9
create_key_ring() (*redstone.client.KeyProtect method*), 9

D

decrypt() (*in module redstone.crypto*), 13
DEFAULT_SESSION (*in module redstone*), 5
delete() (*redstone.client.KeyProtect method*), 9
delete_instance() (*redstone.client.ResourceController method*), 12
delete_key() (*redstone.client.KeyProtect method*), 9
delete_key_alias() (*redstone.client.KeyProtect method*), 9
delete_key_ring() (*redstone.client.KeyProtect method*), 9
disable_key() (*redstone.client.KeyProtect method*), 9

E

enable_key() (*redstone.client.KeyProtect method*), 9
encrypt() (*in module redstone.crypto*), 13
endpoint_for_region() (*redstone.client.CIS method*), 8
endpoint_for_region() (*redstone.client.IKS method*), 8
endpoint_for_region() (*redstone.client.KeyProtect method*), 10
endpoint_for_region() (*redstone.client.ResourceController method*), 12

F

find_space_and_org() (*in module redstone.auth*), 7

G

get() (*redstone.client.KeyProtect method*), 10
get_cluster_config() (*redstone.client.IKS method*), 8
get_clusters() (*redstone.client.IKS method*), 8
get_default_resource_group() (*redstone.client.ResourceController method*), 12
get_default_session() (*in module redstone*), 6
get_import_token() (*redstone.client.KeyProtect method*), 10
get_instance() (*redstone.client.ResourceController method*), 12
get_instance_policies() (*redstone.client.KeyProtect method*), 10
get_key() (*redstone.client.KeyProtect method*), 10
get_key_policies() (*redstone.client.KeyProtect method*), 10
get_key_rings() (*redstone.client.KeyProtect method*), 10
get_kube_versions() (*redstone.client.IKS method*), 8
get_orgs() (*in module redstone.auth*), 7
get_pool() (*redstone.client.CIS method*), 8
get_registrations() (*redstone.client.KeyProtect method*), 10
get_spaces() (*in module redstone.auth*), 7
get_token() (*redstone.auth.TokenManager method*), 6
get_workers() (*redstone.client.IKS method*), 8

I

IKS (*class in redstone.client*), 8

initiate_dual_auth_delete() (*redstone.client.KeyProtect method*), 10
inspect_token() (*in module redstone.auth*), 7
is_refresh_token_expired() (*redstone.auth.TokenManager method*), 6
is_token_expired() (*redstone.auth.TokenManager method*), 6

K

KeyProtect (*class in redstone.client*), 8
KeyProtect.KeyProtectError, 9
KEYPROTECT_PLAN_ID (*redstone.client.ResourceController attribute*), 12
keys() (*redstone.client.KeyProtect method*), 10

L

list_instances() (*redstone.client.ResourceController method*), 12
list_keys() (*redstone.client.KeyProtect method*), 10

M

main() (*in module redstone.auth*), 7
module
    redstone, 5
    redstone.auth, 6
    redstone.client, 7
    redstone.crypto, 13

N

names (*redstone.client.CIS attribute*), 8
names (*redstone.client.IKS attribute*), 8
names (*redstone.client.KeyProtect attribute*), 10
names (*redstone.client.ResourceController attribute*), 12

P

pools() (*redstone.client.CIS method*), 8
prepare_request() (*redstone.client.Session method*), 12
purge_key() (*redstone.client.KeyProtect method*), 10

R

redstone
    module, 5
redstone.auth
    module, 6
redstone.client
    module, 7
redstone.crypto
    module, 13
resource_groups() (*redstone.client.ResourceController method*), 12
ResourceController (*class in redstone.client*), 12
restore_key() (*redstone.client.KeyProtect method*), 11
rotate() (*redstone.client.KeyProtect method*), 11
rotate_key() (*redstone.client.KeyProtect method*), 11

S

service() (*in module redstone*), 6
service() (*redstone.Session method*), 5
Session (*class in redstone*), 5
Session (*class in redstone.client*), 12
set_instance_allowed_ip_policy() (*redstone.client.KeyProtect method*), 11
set_instance_allowed_network_policy() (*redstone.client.KeyProtect method*), 11
set_instance_dual_auth_policy() (*redstone.client.KeyProtect method*), 11
set_instance_key_create_import_access_policy() (*redstone.client.KeyProtect method*), 11
set_instance_metrics_policy() (*redstone.client.KeyProtect method*), 11
set_key_dual_auth_policy() (*redstone.client.KeyProtect method*), 11
set_key_ring() (*redstone.client.KeyProtect method*), 11
set_key_rotation_policy() (*redstone.client.KeyProtect method*), 11
set_pre_send_fn() (*redstone.client.BaseClient method*), 7
sync_associated_resources() (*redstone.client.KeyProtect method*), 12

T

TokenAuth (*class in redstone.client*), 13
TokenManager (*class in redstone.auth*), 6

U

unwrap() (*redstone.client.KeyProtect method*), 12
update_master() (*redstone.client.IKS method*), 8
update_pool() (*redstone.client.CIS method*), 8
update_worker() (*redstone.client.IKS method*), 8

W

wrap() (*redstone.client.KeyProtect method*), 12
wrap() (*redstone.client.KeyProtect.KeyProtectError static method*), 9