COLA

Generated by Doxygen 1.8.15

Contents

Chapter 1

Component-security-policy-manager

v1.0:

Overview:

This module provides APIs to manage sensitive information, including application sensitive information and infrastructure sensitive information.

Infrastructure sensitive information or infrastructure secret:

- Initialize a vault to store secrets for the infrastructure
- Create or update a secret
- Read a secret
- Delete a secret

Application sensitive information or application secret:

- Add application sensitive information as docker secret
- Provision the secret to application services in worker nodes

How to use the API:

Infrastructure sensitive information or infrastructure secret

- Initialize a vault to store secrets

~~~
(Shares must be equal or larger than threshold. If shares > 1, threshold must be larger than 1)

+ Add a secret 'secret1' into the initialized vault. If the secret exists, it will be overwritten.

```curl -H "Content-Type: application/json" -d '{"name":"secret1","value":"123"}' -X POST
        spm:5003/v1.0/secret
~~~

- Update the secret named 'secret1' with a new value

~~~
+ Read a secret named 'secret1' from the vault

```curl -X GET spm:5003/v1.0/secrets/secret
~~~

- Delete a secret named 'secret1' from the vault

~~~
### Application sensitive information or application secret

+ Assuming that a service named 'app1' is already created

+ Add an applicaton sensitive information as docker secret and distribute it to containers of the
       application app1 (If the application service has existing secrets, this function add one more while keeping the other
       secrets intact):

```curl -H "Content-Type: application/json" -d '{"name":"secret1","value":"123","service":"app1"}' -X POST
       spm:5003/v1.0/appsecret
~~~

- Verify if the secret is added to the service or not by calling the below command line in the master node

```
You shall see something like this:
```

Spec": { "TaskTemplate": { "ContainerSpec": { "Secrets": [{ "File": { "Name": "secret1", }, "SecretID":, "Secret↩
Name": "secret1" }, ...

```
+ Retrieve docker secret id of 'secret1'

curl -X GET spm:5003/v1.0/appsecrets/secret1

+ Remove a secret named 'secret1' from the service

curl -H "Content-Type: application/json" -d '{"service":"app1"}' -X DELETE spm:5003/v1.0/appsecrets/secret1
```

~~~
## How to use command line in the master node

### Infrastructure sensitive information or infrastructure secret
+ Initialize a vault to store secrets with shares = 2, threshold = 2

```micadoctl.sh initvault 2
~~~

- Add a secret 'secret1' with value 123 into the initialized vault. If the secret exists, it will be overwritten.

~~~
 {micadoctl.sh}

+ Read a secret named 'secret1' from the vault

```micadoctl.sh readsecret secret
~~~

- Delete a secret named 'secret1' from the vault

~~~
 {micadoctl.sh}

### Application sensitive information or application secret

+ Add a secret 'secret1' with value 123 in docker secrets and provision it to the service 'app1' (assuming
       that the service 'app1' is deployed already)

```micadoctl.sh addappsecret secret1 123 app
~~~

- Check if 'secret1' is provisioned to 'app1' or not

```
## How to use the automatic test script for managing secrets infrastructure sensitive information:

Assuming that you installed Robot framework successfully (Please follow this link if you has not installed
      the Robot framework yet: https://
      github.com/robotframework/QuickStartGuide/blob/master/QuickStart.rst#demo-application)

1. Launch the vault server in localhost:

  * Download the vault server from https://www.vaultproject.io/downloads.html

  * Create a config file named vault.hcl with the below content:
```

storage "file" { path = "datafile" } listener "tcp" { address = "127.0.0.1:8200" tls_disable = 1 }

~~~
(all secrets will be written in the file 'datafile' which resides in the same directory with the executable
       file 'vault')

  * Launch the vault server by command line

```./vault server -config=vault.hc
~~~

1. Edit the file app/vaultclient.py to change VAULT_URL into

~~~
3. Run the source code by command line

```python my_script.p
~~~

1. Run the test script by command line

~~~
## How to use the automatic test script for managing secrets application sensitive information:

1. Run the python code using sudo role

``` sudo python my_script.p
~~~

Please notice that the python code must be run with sudo

1. Create a docker service

~~~
``` docker service create --name app1 <docker_image
~~~

3.Run the test script by command line

```
robot test_script.rst
```

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 app Namespace Reference

Namespaces

- commonfunc
- dksecrets
- routes
- vaultclient

Variables

- app = Flask(__name__)
- logHandler = RotatingFileHandler('error.log', maxBytes=1000, backupCount=1)
- formatter = logging.Formatter('%(asctime)s - %(name)s - %(module)s - %(funcName)s - %(lineno)d- %(level-name)s - %(message)s')

6.1.1 Variable Documentation

6.1.1.1 app

```
app.app = Flask(__name__)
```

6.1.1.2 formatter

```
app.formatter = logging.Formatter('%(asctime)s - %(name)s - %(module)s - %(funcName)s - %(lineno)d-
%(levelname)s - %(message)s')
```

6.1.1.3 logHandler

```
app.logHandler = RotatingFileHandler('error.log', maxBytes=1000, backupCount=1)
```

6.2 app.commonfunc Namespace Reference

Functions

- def create_json_response (http_code, message_label, info_for_developer="", additional_json={})

Variables

- reader = csv.DictReader(open('resource.csv', 'r'))
- dictionary msg_dict = {}

6.2.1 Function Documentation

6.2.1.1 create_json_response()

```
def app.commonfunc.create_json_response (
              http_code,
              message_label,
              info_for_developer = "",
              additional_json = {} )
```

```
[summary]
Create a json object to respond a http request
[description]

Arguments:
    http_code {[type]} -- [description]
    message_label {[type]} -- [description]

Keyword Arguments:
    info_for_developer {str} -- [description] (default: {""}) Additional information in string format
    additional_json {dict} -- [description] (default: {{}}) Additional information in json format

Returns:
    Response [type] -- [description] http response
```

```
def create_json_response(http_code, message_label, info_for_developer="",
     additional_json = {}):
    '''[summary]
    Create a json object to respond a http request
    [description]

    Arguments:
        http_code {[type]} -- [description]
        message_label {[type]} -- [description]

    Keyword Arguments:
        info_for_developer {str} -- [description] (default: {""}) Additional information in string format
        additional_json {dict} -- [description] (default: {{}}) Additional information in json format

    Returns:
        Response [type] -- [description] http response
    '''
    data = {
        'code' : http_code,
        'message'  : msg_dict[message_label] + info_for_developer
    }
    data.update(additional_json)
    js = json.dumps(data)
    resp = Response(js, status=http_code, mimetype='application/json')
    return resp
```

6.2.2 Variable Documentation

6.2.2.1 msg_dict

```
dictionary app.commonfunc.msg_dict = {}
```

6.2.2.2 reader

```
app.commonfunc.reader = csv.DictReader(open('resource.csv', 'r'))
```

6.3 app.dksecrets Namespace Reference

Classes

- class Dksecret
- class Dksecrets

    END - INTERNAL FUNCTIONS.

Functions

- def create_secret (secretname, secretvalue)

    END - CONSTANT VALUES.
- def delete_secret (secretname)

Variables

- int HTTP_CODE_OK = 200

    CONSTANT VALUES.
- int HTTP_CODE_CREATED = 201
- int HTTP_CODE_BAD_REQUEST = 400
- bool DEBUG_MODE = False

6.3.1 Function Documentation

6.3.1.1 create_secret()

```
def app.dksecrets.create_secret (
              secretname,
              secretvalue )
```

END - CONSTANT VALUES.

INTERNAL FUNCTIONS

```
[summary]
Create a docker secret
[description]
    secretname -- name of secret
    secretvalue -- value of secret
Returns:
    [type] Integer -- [description] Id of the created secret

Raises:
    e -- [description]
```

```
def create_secret(secretname, secretvalue):
    '''[summary]
    Create a docker secret
    [description]
        secretname -- name of secret
        secretvalue -- value of secret
    Returns:
        [type] Integer -- [description] Id of the created secret

    Raises:
        e -- [description]
    '''
    client = docker.DockerClient(base_url='unix://var/run/docker.sock')
    try:
        secret=client.api.create_secret(name=secretname,data=secretvalue.encode('utf-8'))
        secretid=secret.get('ID')
    except Exception as e:
        app.logger.error(e)
        raise

    return secretid

```

6.3.1.2 delete_secret()

```
def app.dksecrets.delete_secret (
              secretname )
```

```
[summary]
Delete a docker secret
[description]

Arguments:
    secretname {[type]} -- [description] Name of secret
```

```
def delete_secret(secretname):
    '''[summary]
    Delete a docker secret
    [description]

    Arguments:
        secretname {[type]} -- [description] Name of secret
    '''
    client = docker.DockerClient(base_url='unix://var/run/docker.sock')
    try:
        if DEBUG_MODE:
            print('\nDelete docker secret')

        secret_id = client.api.secrets(filters={"name" : secretname})[0]['ID']

        if DEBUG_MODE:
            print('\nSecret id:', secret_id)
        client.api.remove_secret(secret_id)

    except Exception as e:
        app.logger.error(e)
        raise

```

6.3.2 Variable Documentation

6.3.2.1 DEBUG_MODE

`bool app.dksecrets.DEBUG_MODE = False`

6.3.2.2 HTTP_CODE_BAD_REQUEST

`int app.dksecrets.HTTP_CODE_BAD_REQUEST = 400`

6.3.2.3 HTTP_CODE_CREATED

`int app.dksecrets.HTTP_CODE_CREATED = 201`

6.3.2.4 HTTP_CODE_OK

`int app.dksecrets.HTTP_CODE_OK = 200`

CONSTANT VALUES.

6.4 app.routes Namespace Reference

Variables

- api = Api(app)

6.4.1 Variable Documentation

6.4.1.1 api

```
app.routes.api = Api(app)
```

6.5 app.vaultclient Namespace Reference

Classes

- class Secret
- class Secrets
- class Vaults

    END - INTERNAL FUNCTIONS.

Functions

- def read_token ()

    END - MODELS.
- def read_unseal_keys ()
- def init_client ()
- def unseal_vault (client)
- def seal_vault (client)

Variables

- string VAULT_URL = "http://credstore:8200"

    CONSTANT VALUES "http://127.0.0.1:8200" for localhost test, "http://credstore:8200" for docker environment.
- int HTTP_CODE_OK = 200
- int HTTP_CODE_CREATED = 201
- int HTTP_CODE_BAD_REQUEST = 400
- int HTTP_CODE_NOT_FOUND = 404
- int HTTP_CODE_SERVER_ERR = 500
- int DEFAULT_SHARES = 1
- int DEFAULT_THRESHOLD = 1
- string VAULT_TOKEN_FILE = 'vaulttoken'
- string UNSEAL_KEYS_FILE = 'unsealkeys'
- bool DEBUG_MODE = False
- dictionary vault_init_params

    END - CONSTANT VALUES.

6.5.1 Function Documentation

6.5.1.1 init_client()

```
def app.vaultclient.init_client ( )
```

```
[summary]
Initialize the vault client
[description]

Returns:
  [type] -- [description]
```

```
def init_client():
    """[summary]
    Initialize the vault client
    [description]

    Returns:
      [type] -- [description]
    """
    client = hvac.Client(url=VAULT_URL)
    return client

```

6.5.1.2 read_token()

```
def app.vaultclient.read_token ( )
```

END - MODELS.

INTERNAL FUNCTIONS

```
[summary]
Read the token from file 'vaulttoken'
[description]

Returns:
  [type] string -- [description] the token
```

```
def read_token():
    """[summary]
    Read the token from file 'vaulttoken'
    [description]

    Returns:
      [type] string -- [description] the token
    """
    try:
        f = open(VAULT_TOKEN_FILE, 'r')
        root_token = f.read()
        f.close()
        return root_token
    except Exception as e:
        app.logger.error(e)
        raise

```

6.5.1.3 read_unseal_keys()

```
def app.vaultclient.read_unseal_keys ( )
```

```
[summary]
Read keys used to unseal the vault from file ’unsealkeys’
[description]

Returns:
  [type] List -- [description] List of keys
```

```
def read_unseal_keys():
    """[summary]
    Read keys used to unseal the vault from file ’unsealkeys’
    [description]

    Returns:
      [type] List -- [description] List of keys
    """
    try:
        f = open(UNSEAL_KEYS_FILE, ’r’)
        unseal_keys = f.read().splitlines()
        f.close()
        return unseal_keys
    except Exception as e:
        app.logger.error(e)
        raise
```

6.5.1.4 seal_vault()

```
def app.vaultclient.seal_vault (
                client )
```

```
[summary]
Seal the vault
[description]
This should be done to protect the vault while not using it
Arguments:
  vault client {[type]} -- [description] vault client
```

```
def seal_vault(client):
    """[summary]
    Seal the vault
    [description]
    This should be done to protect the vault while not using it
    Arguments:
      vault client {[type]} -- [description] vault client
    """
    client.seal()
```

6.5.1.5 unseal_vault()

```
def app.vaultclient.unseal_vault (
              client )
```

```
[summary]
Unseal (open) the vault
[description]
This must be done prior to read contents from the vault.
Arguments:
  vault client {[type]} -- [description] vault client
```

```
def unseal_vault(client):
    """[summary]
    Unseal (open) the vault
    [description]
    This must be done prior to read contents from the vault.
    Arguments:
      vault client {[type]} -- [description] vault client
     """
     client.token = read_token()
     # unseal the vault
     unseal_keys = read_unseal_keys()
     client.unseal_multi(unseal_keys)

```

6.5.2 Variable Documentation

6.5.2.1 DEBUG_MODE

```
bool app.vaultclient.DEBUG_MODE = False
```

6.5.2.2 DEFAULT_SHARES

```
int app.vaultclient.DEFAULT_SHARES = 1
```

6.5.2.3 DEFAULT_THRESHOLD

```
int app.vaultclient.DEFAULT_THRESHOLD = 1
```

6.5.2.4 HTTP_CODE_BAD_REQUEST

```
int app.vaultclient.HTTP_CODE_BAD_REQUEST = 400
```

6.5.2.5 HTTP_CODE_CREATED

```
int app.vaultclient.HTTP_CODE_CREATED = 201
```

6.5.2.6 HTTP_CODE_NOT_FOUND

```
int app.vaultclient.HTTP_CODE_NOT_FOUND = 404
```

6.5.2.7 HTTP_CODE_OK

```
int app.vaultclient.HTTP_CODE_OK = 200
```

6.5.2.8 HTTP_CODE_SERVER_ERR

```
int app.vaultclient.HTTP_CODE_SERVER_ERR = 500
```

6.5.2.9 UNSEAL_KEYS_FILE

```
string app.vaultclient.UNSEAL_KEYS_FILE = 'unsealkeys'
```

6.5.2.10 vault_init_params

```
dictionary app.vaultclient.vault_init_params
```

Initial value:

```
=  {
    'shares': fields.Integer,
    'threshold' : fields.Integer
}
```

END - CONSTANT VALUES.

MODELS

6.5.2.11 VAULT_TOKEN_FILE

```
string app.vaultclient.VAULT_TOKEN_FILE = 'vaulttoken'
```

6.5.2.12 VAULT_URL

```
string app.vaultclient.VAULT_URL = "http://credstore:8200"
```

CONSTANT VALUES "http://127.0.0.1:8200" for localhost test, "http://credstore:8200" for docker environment.

6.6 app_linebr Namespace Reference

6.6.1 Detailed Description

```
[summary]
Init module
[description]
The init module creates Flask object, databases, and logging handler
```

6.7 CredStoreLibrary Namespace Reference

Classes

- class CredStoreLibrary

Variables

- int http_code_ok = 200

6.7.1 Variable Documentation

6.7.1.1 http_code_ok

```
int CredStoreLibrary.http_code_ok = 200
```

6.8 my_script Namespace Reference

Variables

- host
- port

6.8.1 Variable Documentation

6.8.1.1 host

```
my_script.host
```

6.8.1.2 port

```
my_script.port
```

6.9 my_script_linebr Namespace Reference

6.9.1 Detailed Description

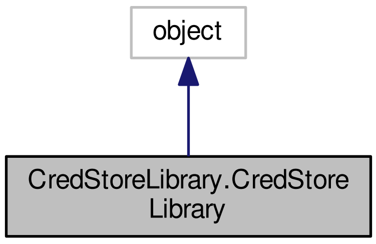
```
[summary]
Main module.
[description]
The main module starts the web service
```

Chapter 7

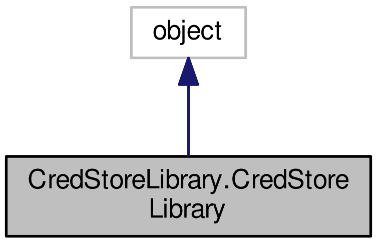
Class Documentation

7.1 CredStoreLibrary.CredStoreLibrary Class Reference

Inheritance diagram for CredStoreLibrary.CredStoreLibrary:



Collaboration diagram for CredStoreLibrary.CredStoreLibrary:



Public Member Functions

- def __init__ (self)
- def status_should_be (self, expected_status)
- def data_should_be (self, expected_data)
- def add_a_secret (self, name, value)
- def update_a_secret (self, secretname, newvalue)
- def delete_a_secret (self, secretname=None)
- def read_a_secret (self, secretname=None)
- def init_a_vault (self, shares, threshold)
- def create_app_secret (self, secretname, secretvalue, servicename)
- def delete_app_secret (self, secretname, servicename)
- def retrieve_app_secret_id (self, secretname)

7.1.1 Constructor & Destructor Documentation

7.1.1.1 __init__()

```
def CredStoreLibrary.CredStoreLibrary.__init__ (
               self )
```

```
    def __init__(self):
        #self._sut_path = os.path.join(os.path.dirname(__file__),
         #                             '..', 'sut', 'login.py')
        self._status = ''
        self._data = ''

```

7.1.2 Member Function Documentation

7.1.2.1 add_a_secret()

```
def CredStoreLibrary.CredStoreLibrary.add_a_secret (
               self,
               name,
               value )
```

```
    def add_a_secret(self, name, value):
        url     = 'http://127.0.0.1:5003/v1.0/secrets'
        payload = {'name': name, 'value': value}
        res = requests.post(url, json=payload)
        json_data = json.loads(res.text)
        self._status = json_data['code']

```

7.1.2.2 create_app_secret()

```
def CredStoreLibrary.CredStoreLibrary.create_app_secret (
                self,
                secretname,
                secretvalue,
                servicename )
```

```
    def create_app_secret(self, secretname, secretvalue, servicename):
        url     = 'http://127.0.0.1:5003/v1.0/appsecrets'
        payload = {'name': secretname, 'value': secretvalue, 'service': servicename}
        res = requests.post(url, json=payload)
        json_data = json.loads(res.text)
        self._status = json_data['code']

```

7.1.2.3 data_should_be()

```
def CredStoreLibrary.CredStoreLibrary.data_should_be (
                self,
                expected_data )
```

```
    def data_should_be(self, expected_data):
        if expected_data != str(self._data):
            raise AssertionError("Expected data to be '%s' but was '%s'."
                                 % (expected_data, self._data))

```

7.1.2.4 delete_a_secret()

```
def CredStoreLibrary.CredStoreLibrary.delete_a_secret (
                self,
                secretname = None )
```

```
    def delete_a_secret(self, secretname=None,):
        url     = 'http://127.0.0.1:5003/v1.0/secrets/' + secretname
        res = requests.delete(url)
        json_data = json.loads(res.text)
        self._status = json_data['code']

```

7.1.2.5 delete_app_secret()

```
def CredStoreLibrary.CredStoreLibrary.delete_app_secret (
                self,
                secretname,
                servicename )
```

```
    def delete_app_secret(self, secretname, servicename):
        url     = 'http://127.0.0.1:5003/v1.0/appsecrets/' + secretname
        payload = {'service': servicename}
        res = requests.delete(url, json=payload)
        json_data = json.loads(res.text)
        self._status = json_data['code']

```

7.1.2.6 init_a_vault()

```
def CredStoreLibrary.CredStoreLibrary.init_a_vault (
              self,
              shares,
              threshold )
```

```
    def init_a_vault(self, shares, threshold):
        url     = 'http://127.0.0.1:5003/v1.0/vaults'
        payload = {'shares': shares, 'threshold': threshold}
        res = requests.post(url, json=payload)
        json_data = json.loads(res.text)
        self._status = json_data['code']

```

7.1.2.7 read_a_secret()

```
def CredStoreLibrary.CredStoreLibrary.read_a_secret (
              self,
              secretname = None )
```

```
    def read_a_secret(self, secretname=None):
        url     = 'http://127.0.0.1:5003/v1.0/secrets/' + secretname
        res = requests.get(url)
        json_data = json.loads(res.text)
        self._status = json_data['code']
        if(self._status == http_code_ok):
            self._data = json_data['data']['secret_value']

```

7.1.2.8 retrieve_app_secret_id()

```
def CredStoreLibrary.CredStoreLibrary.retrieve_app_secret_id (
              self,
              secretname )
```

```
    def retrieve_app_secret_id(self, secretname):
        url     = 'http://127.0.0.1:5003/v1.0/appsecrets/' + secretname
        res = requests.get(url)
        json_data = json.loads(res.text)
        self._status = json_data['code']
```

7.1.2.9 status_should_be()

```
def CredStoreLibrary.CredStoreLibrary.status_should_be (
              self,
              expected_status )
```

```
    def status_should_be(self, expected_status):
        if expected_status != str(self._status):
            raise AssertionError("Expected status to be '%s' but was '%s'."
                                 % (expected_status, self._status))

```

7.1.2.10 update_a_secret()

```
def CredStoreLibrary.CredStoreLibrary.update_a_secret (
              self,
              secretname,
              newvalue )
```

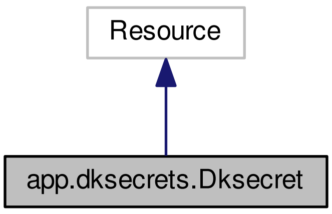
```
    def update_a_secret(self,secretname,newvalue):
        url     = 'http://127.0.0.1:5003/v1.0/secrets/' + secretname
        payload = {'value': newvalue}
        res = requests.put(url, json=payload)
        json_data = json.loads(res.text)
        self._status = json_data['code']

```

The documentation for this class was generated from the following file:

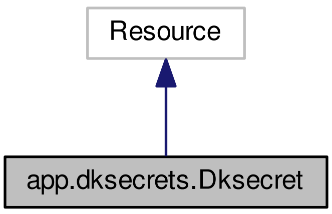
- lib/CredStoreLibrary.py

7.2 app.dksecrets.Dksecret Class Reference

Inheritance diagram for app.dksecrets.Dksecret:



Collaboration diagram for app.dksecrets.Dksecret:



Public Member Functions

- def delete (self, secret_name)
- def get (self, secret_name)

7.2.1 Member Function Documentation

7.2.1.1 delete()

```
def app.dksecrets.Dksecret.delete (
              self,
              secret_name )
```

```
[summary]
Remove secret from a service.
[description]
Remove secret from all container of a services. If there are no other services using the secret, delete the se
Arguments:
    secret_name {[type]} -- [description] Name of secret

Returns:
    Json [type] -- [description] Returned message

Raises:
    e -- [description]
```

```
 def delete(self,secret_name):
     '''[summary]
     Remove secret from a service.
     [description]
     Remove secret from all container of a services. If there are no other services using the secret,
 delete the secret from docker swarm
     Arguments:
         secret_name {[type]} -- [description] Name of secret

     Returns:
         Json [type] -- [description] Returned message

     Raises:
         e -- [description]
     '''
     json_body = request.json
     service_name = json_body['service'] #Application/ service name

     if DEBUG_MODE:
         print('\nDELETE SERVICE SECRET')

     try:
         client = docker.APIClient(base_url='unix://var/run/docker.sock')
         service = client.services(filters={"name" : service_name}).pop(0) # Get the service by name
     except Exception as e:
         app.logger.error(e)
         resp = create_json_response(HTTP_CODE_BAD_REQUEST,'non_exist_service')
         return resp

     service_id = service['ID'] # get service_id
     service_version = service['Version']['Index'] # get service_version

     container_spec = service['Spec']['TaskTemplate']['ContainerSpec'] #get service container
 specification
     current_secrets = []
     try:
         current_secrets = container_spec['Secrets']
         if DEBUG_MODE:
             print('\nSecret list before removed: ', current_secrets)
             print('1 item: ',current_secrets[0])
             print('Type of current_secrets: ', type(current_secrets))
             print('Type of item: ', type(current_secrets[0]))
```

```
                print('Secret name of 1st item: ', current_secrets[0]['SecretName'])

            update_current_secrets = [x for x in current_secrets if x['SecretName'] != secret_name]

            if DEBUG_MODE:
                print('\nSecret list after removed: ', update_current_secrets)

            container_spec['Secrets'] = update_current_secrets
        except Exception as e:
            app.logger.error(e)
            raise e

        task_tmpl = docker.types.TaskTemplate(container_spec) # Create TaskTemplate from container
 specification

        if DEBUG_MODE:
            print("\nTask template", task_tmpl.container_spec)

        client.update_service(service=service_id,name=service_name,version=service_version,task_template=
task_tmpl) # Update service with new secret list

        if DEBUG_MODE:
            print("\nConfig of service after update:", client.inspect_service(service=service_name))

        # Try to delete docker secret (if no other services are using it)
        try:
            delete_secret(secret_name)
        except Exception as e:
            app.logger.error(e)
            pass

        resp = create_json_response(HTTP_CODE_OK,'remove_dksecret_success')
        return resp
```

7.2.1.2 get()

```
def app.dksecrets.Dksecret.get (
                self,
                secret_name )
```

```
[summary]
Retrieve id of a secret
[description]

Arguments:
    secret_name {[type]} -- [description] Name of secret

Returns:
    [type] -- [description]
```

```
    def get(self, secret_name):
        '''[summary]
        Retrieve id of a secret
        [description]

        Arguments:
            secret_name {[type]} -- [description] Name of secret

        Returns:
            [type] -- [description]
        '''
        client = docker.DockerClient(base_url='unix://var/run/docker.sock')
        if DEBUG_MODE:
            print('\nRetrieve docker secret')

        try:
            secret_id = client.api.secrets(filters={"name" : secret_name})[0]['ID']

            if DEBUG_MODE:
                print('\nSecret id:', secret_id)
            info = {'secret_id':secret_id}
            resp = create_json_response(HTTP_CODE_OK,'secret_exists', additional_json =
info)
```

```
            return resp
        except Exception as e:
            app.logger.error(e)
            resp = create_json_response(HTTP_CODE_BAD_REQUEST,'non_exist_secret')
            return resp
```

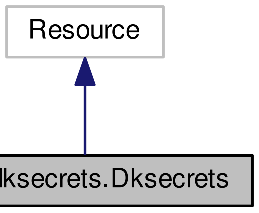
The documentation for this class was generated from the following file:

- app/dksecrets.py

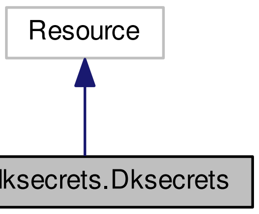
7.3 app.dksecrets.Dksecrets Class Reference

END - INTERNAL FUNCTIONS.

Inheritance diagram for app.dksecrets.Dksecrets:



Collaboration diagram for app.dksecrets.Dksecrets:



Public Member Functions

- def post (self)

7.3.1 Detailed Description

END - INTERNAL FUNCTIONS.

RESOURCES

7.3.2 Member Function Documentation

7.3.2.1 post()

```
def app.dksecrets.Dksecrets.post (
                self )
```

```
[summary]
Creates a docker secret and distribute it to docker service
[description]
Input:
    name -- secret's name
    value -- secret's value
    service -- service' name
Assuming that the service exists, this API creates a docker secret from the inputted name and value, then dist
Returns:
    [type] Json object -- [description] Successful message or error message
```

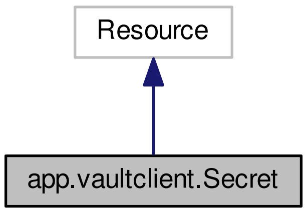
```
    def post(self):
        '''[summary]
        Creates a docker secret and distribute it to docker service
        [description]
        Input:
            name -- secret's name
            value -- secret's value
            service -- service' name
        Assuming that the service exists, this API creates a docker secret from the inputted name and
    value, then distribute the docker secret to the corresponding service
        Returns:
            [type] Json object -- [description] Successful message or error message
        '''

        json_body = request.json

        secret_name = json_body['name'] # Secret name
        secret_value = json_body['value'] #Secret value
        service_name = json_body['service'] #Application/ service name

        # Verify query arguments
        if(secret_name is None or secret_value  is None or service_name is None or secret_name=='' or
  secret_value=='' or service_name==''):
            resp = create_json_response(HTTP_CODE_BAD_REQUEST,'
  bad_request_write_dksecret')
            return resp

        if DEBUG_MODE:
            print('\nADD APPLICATION SECRET INTO DOCKER')

        try:
            client = docker.APIClient(base_url='unix://var/run/docker.sock')
            service = client.services(filters={"name" : service_name}).pop(0) # Get the service by name
        except Exception as e:
            app.logger.error(e)
            resp = create_json_response(HTTP_CODE_BAD_REQUEST,'non_exist_service')
            return resp

        # Create docker secret and add it to corresponding service
        try:
            secret_id = create_secret(secret_name, secret_value) # Create docker secret
        except Exception as e:
            #app.logger.error(e)
            resp = create_json_response(HTTP_CODE_BAD_REQUEST,'existed_secret')
            return resp

        service_id = service['ID'] # get service_id
        service_version = service['Version']['Index'] # get service_version

        container_spec = service['Spec']['TaskTemplate']['ContainerSpec'] #get service container
 specification
        current_secrets = []
        try:
            current_secrets = container_spec['Secrets'] # get the service's current secret list (if
 existed)
        except:
            pass

        secret_list = [docker.types.SecretReference(secret_id,secret_name)] # Create list of
 SecretReference
        container_spec['Secrets']=current_secrets+secret_list # update list of secrets for the service
        task_tmpl = docker.types.TaskTemplate(container_spec) # Create TaskTemplate from container
 specification

        if DEBUG_MODE:
            print("Task template", task_tmpl.container_spec)

        client.update_service(service=service_id,name=service_name,version=service_version,task_template=
task_tmpl) # Update service with new secret list

        if DEBUG_MODE:
            print("Config of service after update:", client.inspect_service(service=service_name))

        resp = create_json_response(HTTP_CODE_CREATED,'write_dksecret_success')
        return resp
```

The documentation for this class was generated from the following file:

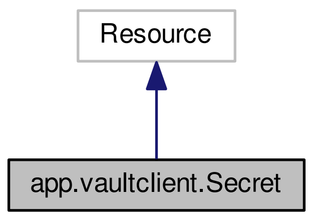
- app/dksecrets.py

7.4 app.vaultclient.Secret Class Reference

Inheritance diagram for app.vaultclient.Secret:



Collaboration diagram for app.vaultclient.Secret:



Public Member Functions

- def get (self, secret_name)
- def delete (self, secret_name)
- def put (self, secret_name)

7.4.1 Member Function Documentation

7.4.1.1 delete()

```
def app.vaultclient.Secret.delete (
              self,
              secret_name )
```

```
[summary]
Remove a secret from the vault
[description]
```

```
    def delete(self, secret_name):
        """[summary]
        Remove a secret from the vault
        [description]
        """
        if DEBUG_MODE:
            print('\nDELETE SECRET FROM VAULT')

        if(secret_name is None or  secret_name==''): # verify parameters
            resp = create_json_response(HTTP_CODE_BAD_REQUEST,'
      bad_request_delete_secret') #'Lack of secret name'
            return resp

        # unseal the vault
        client = init_client()
        try:
            unseal_vault(client)
        except Exception as e:
            app.logger.error(e)
            resp = create_json_response(HTTP_CODE_SERVER_ERR,'vault_not_initialized')
            return resp

        client.delete('secret/'+secret_name)
        seal_vault(client)

        resp = create_json_response(HTTP_CODE_OK,'delete_secret_success') #'Delete
       secret successfully
        return resp

```

7.4.1.2 get()

```
def app.vaultclient.Secret.get (
              self,
              secret_name )
```

```
[summary]
Read a secret from the vault
[description]

Returns:
  [type] json -- [description] a dictionary of all relevant information of the secret
```

```
    def get(self,secret_name):
        """[summary]
        Read a secret from the vault
        [description]

        Returns:
          [type] json -- [description] a dictionary of all relevant information of the secret
        """
        if(secret_name is None or  secret_name==''): # verify parameters
            resp = create_json_response(HTTP_CODE_BAD_REQUEST,'bad_request_read_secret'
      )
            return resp

        if DEBUG_MODE:
            print('READ SECRET FROM VAULT')

        # unseal the vault
        client = init_client()
        try:
            unseal_vault(client)
        except Exception as e:
            app.logger.error(e)
            resp = create_json_response(HTTP_CODE_SERVER_ERR,'vault_not_initialized')
            return resp

        secret_values = client.read('secret/'+secret_name)
        seal_vault(client)

        if(secret_values is None): # If the required secret does not exist
            resp = create_json_response(HTTP_CODE_NOT_FOUND,'secret_not_exist')
            return resp
        else:
            resp = create_json_response(HTTP_CODE_OK,'read_secret_success',
      additional_json =secret_values)
            return resp
```

7.4.1.3 put()

```
def app.vaultclient.Secret.put (
              self,
              secret_name )
```

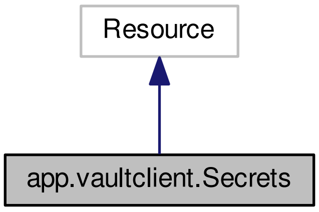
```
[summary]
Update a secret in the vault
[description]

Arguments:
    secret_name {[type]} -- [description] Name of secret

Returns:
    [type] -- [description]
    def put(self, secret_name):
        '''[summary]
        Update a secret in the vault
        [description]

        Arguments:
            secret_name {[type]} -- [description] Name of secret

        Returns:
            [type] -- [description]
        '''
        if DEBUG_MODE:
            print('\nUPDATE SECRET FROM VAULT')

        json_body = request.json
        secret_value = json_body['value']

        if(secret_value is None or  secret_value==''): # verify parameters
            resp = create_json_response(HTTP_CODE_BAD_REQUEST,'
      bad_request_update_secret') #'Lack of secret name'
            return resp

         # unseal the vault
        client = init_client()
        try:
            unseal_vault(client)
        except Exception as e:
            app.logger.error(e)
            resp = create_json_response(HTTP_CODE_SERVER_ERR,'vault_not_initialized')
            return resp

        secret_values = client.read('secret/'+secret_name)

        if(secret_values is None): # If the required secret does not exist
            resp = create_json_response(HTTP_CODE_NOT_FOUND,'secret_not_exist')
            return resp
        else:
            client.write('secret/'+secret_name, secret_value=secret_value)#, lease='1h'
            seal_vault(client)
            resp = create_json_response(HTTP_CODE_OK,'update_secret_success')
            return resp
```

The documentation for this class was generated from the following file:

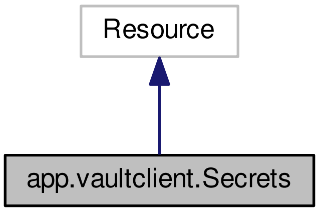
- app/vaultclient.py

7.5 app.vaultclient.Secrets Class Reference

Inheritance diagram for app.vaultclient.Secrets:



Collaboration diagram for app.vaultclient.Secrets:



Public Member Functions

- def post (self)

7.5.1 Member Function Documentation

7.5.1.1 post()

```
def app.vaultclient.Secrets.post (
                self )
```

```
[summary]
 Write or update a secret to the vault
[description]
Arguments:
    name -- name of secret
    value -- value of secret
```

```
    def post(self):
        '''[summary]
         Write or update a secret to the vault
        [description]
        Arguments:
            name -- name of secret
            value -- value of secret
        '''
        json_body = request.json
        secret_name = json_body['name']
        secret_value = json_body['value']

        if(secret_name is None or secret_value  is None or secret_value=='' or  secret_name==''): # verify
       parameters
            resp = create_json_response(HTTP_CODE_BAD_REQUEST,'bad_request_write_secret
      ')
            return resp

        if DEBUG_MODE:
            print('\nADD SECRET INTO VAULT')
            print('secret name:', secret_name)
            print('secret_value:', secret_value)

        client = init_client()
        try:
            unseal_vault(client)
        except Exception as e:
            app.logger.error(e)
            resp = create_json_response(HTTP_CODE_SERVER_ERR,'vault_not_initialized')
            return resp

        client.write('secret/'+secret_name, secret_value=secret_value)#, lease='1h'
        seal_vault(client)

        resp = create_json_response(HTTP_CODE_CREATED,'write_secret_success')
        return resp

```

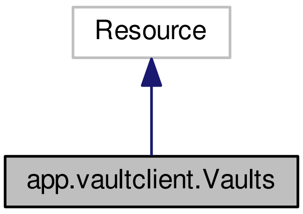
The documentation for this class was generated from the following file:

- app/vaultclient.py

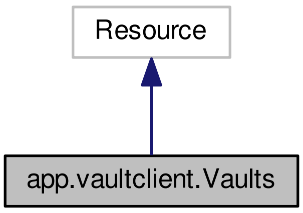
7.6 app.vaultclient.Vaults Class Reference

END - INTERNAL FUNCTIONS.

Inheritance diagram for app.vaultclient.Vaults:



Collaboration diagram for app.vaultclient.Vaults:



Public Member Functions

- def post (self)

7.6.1 Detailed Description

END - INTERNAL FUNCTIONS.

RESOURCES

7.6.2 Member Function Documentation

7.6.2.1 post()

```
def app.vaultclient.Vaults.post (
               self )
```

```
[summary]
Initialize a vault in the Vault Server (Credential Store)
[description]
A number of keys will be generated from the master key, then the master key is thrown away (The Server will nc
shares = The number of generated keys
threshold = The minimum number of generated keys needed to unseal the vault
```

```
    def post(self):
        """[summary]
        Initialize a vault in the Vault Server (Credential Store)
        [description]
        A number of keys will be generated from the master key, then the master key is thrown away (The
       Server will not store the key). The generated keys are kept by the Vault Client (Security Policy Manager)
        shares = The number of generated keys
        threshold = The minimum number of generated keys needed to unseal the vault
        """
        json_body = request.json

        marshal_json = marshal(json_body, vault_init_params)

        shares = marshal_json['shares']
        threshold = marshal_json['threshold']

        if DEBUG_MODE:
            print('\nINIT VAULT:')
            print('shares: ', shares)
            print('threshold: ', threshold)

        if(threshold>shares or (shares>=2 and threshold==1) or shares<=0 or threshold<=0):
            resp = create_json_response(HTTP_CODE_BAD_REQUEST,'
      init_vault_fail_due_to_parameter')
            return resp

        vault_exist = False
        try:
            client = init_client()
            vault_exist = client.is_initialized()
        except Exception as e:
            app.logger.error(e)
            resp = create_json_response(HTTP_CODE_SERVER_ERR,'init_vault_fail') # Fail
       to initialize vault
            return resp

        if(vault_exist): # if vault existed
            resp = create_json_response(HTTP_CODE_CREATED,'vault_existed')
            return resp
        else:
```

```
        vault = client.initialize(shares,threshold)
        root_token = vault['root_token']
        unseal_keys = vault['keys']
        # write root token into file
        f = open(VAULT_TOKEN_FILE, 'w')
        f.write(root_token)
        f.close()

        # write unseal_keys into file
        f = open(UNSEAL_KEYS_FILE,'w')
        for key in unseal_keys:
            f.write("%s\n" % key)
        f.close()
        resp = create_json_response(HTTP_CODE_CREATED,'init_vault_success') # Initialize vault successfully
        return resp

```

The documentation for this class was generated from the following file:

- app/vaultclient.py

Chapter 8

File Documentation

8.1 app/__init__.py File Reference

Namespaces

- app
- app_linebr

Variables

- app.app = Flask(__name__)
- app.logHandler = RotatingFileHandler('error.log', maxBytes=1000, backupCount=1)
- app.formatter = logging.Formatter('%(asctime)s - %(name)s - %(module)s - %(funcName)s - %(lineno)d- %(levelname)s - %(message)s')

8.2 app/commonfunc.py File Reference

Namespaces

- app.commonfunc

Functions

- def app.commonfunc.create_json_response (http_code, message_label, info_for_developer="", additional↩ _json={})

Variables

- app.commonfunc.reader = csv.DictReader(open('resource.csv', 'r'))
- dictionary app.commonfunc.msg_dict = {}

8.3 app/dksecrets.py File Reference

Classes

- class app.dksecrets.Dksecrets

  END - INTERNAL FUNCTIONS.
- class app.dksecrets.Dksecret

Namespaces

- app.dksecrets

Functions

- def app.dksecrets.create_secret (secretname, secretvalue)

  END - CONSTANT VALUES.
- def app.dksecrets.delete_secret (secretname)

Variables

- int app.dksecrets.HTTP_CODE_OK = 200

  CONSTANT VALUES.
- int app.dksecrets.HTTP_CODE_CREATED = 201
- int app.dksecrets.HTTP_CODE_BAD_REQUEST = 400
- bool app.dksecrets.DEBUG_MODE = False

8.4 app/routes.py File Reference

Namespaces

- app.routes

Variables

- app.routes.api = Api(app)

8.5 app/vaultclient.py File Reference

Classes

- class app.vaultclient.Vaults

  END - INTERNAL FUNCTIONS.
- class app.vaultclient.Secrets
- class app.vaultclient.Secret

Namespaces

- app.vaultclient

Functions

- def app.vaultclient.read_token ()

  END - MODELS.
- def app.vaultclient.read_unseal_keys ()
- def app.vaultclient.init_client ()
- def app.vaultclient.unseal_vault (client)
- def app.vaultclient.seal_vault (client)

Variables

- string app.vaultclient.VAULT_URL = "http://credstore:8200"

  CONSTANT VALUES "http://127.0.0.1:8200" for localhost test, "http://credstore:8200" for docker environment.
- int app.vaultclient.HTTP_CODE_OK = 200
- int app.vaultclient.HTTP_CODE_CREATED = 201
- int app.vaultclient.HTTP_CODE_BAD_REQUEST = 400
- int app.vaultclient.HTTP_CODE_NOT_FOUND = 404
- int app.vaultclient.HTTP_CODE_SERVER_ERR = 500
- int app.vaultclient.DEFAULT_SHARES = 1
- int app.vaultclient.DEFAULT_THRESHOLD = 1
- string app.vaultclient.VAULT_TOKEN_FILE = 'vaulttoken'
- string app.vaultclient.UNSEAL_KEYS_FILE = 'unsealkeys'
- bool app.vaultclient.DEBUG_MODE = False
- dictionary app.vaultclient.vault_init_params

  END - CONSTANT VALUES.

8.6 lib/CredStoreLibrary.py File Reference

Classes

- class CredStoreLibrary.CredStoreLibrary

Namespaces

- CredStoreLibrary

Variables

- int CredStoreLibrary.http_code_ok = 200

8.7 my_script.py File Reference

Namespaces

- my_script
- my_script_linebr

Variables

- my_script.host
- my_script.port

8.8 README.md File Reference