

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 'appl' is already created
+ Add an application sensitive information as docker secret and distribute it to containers of the
 application appl (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":"appl"}' -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": "SecretID", "SecretName": "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":"appl"}' -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 'appl' (assuming
  that the service 'appl' 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](https://github.com/robotframework/QuickStartGuide/blob/master/QuickStart.rst#demo-application) 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 appl <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:

|                                  |    |
|----------------------------------|----|
| <a href="#">app</a>              | ?? |
| <a href="#">app.commonfunc</a>   | ?? |
| <a href="#">app.dksecrets</a>    | ?? |
| <a href="#">app.routes</a>       | ?? |
| <a href="#">app.vaultclient</a>  | ?? |
| <a href="#">app_linebr</a>       | ?? |
| <a href="#">CredStoreLibrary</a> | ?? |
| <a href="#">my_script</a>        | ?? |
| <a href="#">my_script_linebr</a> | ?? |



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

|                                   |    |
|-----------------------------------|----|
| object                            |    |
| CredStoreLibrary.CredStoreLibrary | ?? |
| Resource                          |    |
| app.dksecrets.Dksecret            | ?? |
| app.dksecrets.Dksecrets           | ?? |
| app.vaultclient.Secret            | ?? |
| app.vaultclient.Secrets           | ?? |
| app.vaultclient.Vaults            | ?? |



## Chapter 4

# Class Index

### 4.1 Class List

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

|                                                             |    |
|-------------------------------------------------------------|----|
| <a href="#">CredStoreLibrary.CredStoreLibrary</a> . . . . . | ?? |
| <a href="#">app.dksecrets.Dksecret</a> . . . . .            | ?? |
| <a href="#">app.dksecrets.Dksecrets</a>                     |    |
| END - INTERNAL FUNCTIONS . . . . .                          | ?? |
| <a href="#">app.vaultclient.Secret</a> . . . . .            | ?? |
| <a href="#">app.vaultclient.Secrets</a> . . . . .           | ?? |
| <a href="#">app.vaultclient.Vaults</a>                      |    |
| END - INTERNAL FUNCTIONS . . . . .                          | ?? |



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

|                                         |           |    |
|-----------------------------------------|-----------|----|
| <a href="#">my_script.py</a>            | . . . . . | ?? |
| <a href="#">app/__init__.py</a>         | . . . . . | ?? |
| <a href="#">app/commonfunc.py</a>       | . . . . . | ?? |
| <a href="#">app/dksecrets.py</a>        | . . . . . | ?? |
| <a href="#">app/routes.py</a>           | . . . . . | ?? |
| <a href="#">app/vaultclient.py</a>      | . . . . . | ?? |
| <a href="#">lib/CredStoreLibrary.py</a> | . . . . . | ?? |





## 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- %(levelname)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

```
11 def create_json_response(http_code, message_label, info_for_developer="",
12 additional_json = {}):
13 '''[summary]
14 Create a json object to respond a http request
15 [description]
16
17 Arguments:
18 http_code {[type]} -- [description]
19 message_label {[type]} -- [description]
20
21 Keyword Arguments:
22 info_for_developer {str} -- [description] (default: {}) Additional information in string format
23 additional_json {dict} -- [description] (default: {}) Additional information in json format
24
25 Returns:
26 Response [type] -- [description] http response
27 '''
28 data = {
29 'code' : http_code,
30 'message' : msg_dict[message_label] + info_for_developer
31 }
32 data.update(additional_json)
33 js = json.dumps(data)
34 resp = Response(js, status=http_code, mimetype='application/json')
35 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]
```

```
24 def create_secret(secretname, secretvalue):
25 '''[summary]
26 Create a docker secret
27 [description]
28 secretname -- name of secret
29 secretvalue -- value of secret
30 Returns:
31 [type] Integer -- [description] Id of the created secret
32
33 Raises:
34 e -- [description]
35 '''
36 client = docker.DockerClient(base_url='unix://var/run/docker.sock')
37 try:
38 secret=client.api.create_secret(name=secretname,data=secretvalue.encode('utf-8'))
39 secretid=secret.get('ID')
40 except Exception as e:
41 app.logger.error(e)
42 raise
43
44 return secretid
45
```

### 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
```

```

46 def delete_secret(secretname):
47 '''[summary]
48 Delete a docker secret
49 [description]
50
51 Arguments:
52 secretname {[type]} -- [description] Name of secret
53 '''
54 client = docker.DockerClient(base_url='unix://var/run/docker.sock')
55 try:
56 if DEBUG_MODE:
57 print('\nDelete docker secret')
58
59 secret_id = client.api.secrets(filters={"name" : secretname})[0]['ID']
60
61 if DEBUG_MODE:
62 print('\nSecret id:', secret_id)
63 client.api.remove_secret(secret_id)
64
65 except Exception as e:
66 app.logger.error(e)
67 raise
68

```

## 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]

82 def init_client():
83 """[summary]
84 Initialize the vault client
85 [description]
86
87 Returns:
88 [type] -- [description]
89 """
90 client = hvac.Client(url=VAULT_URL)
91 return client
92
```

### 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

48 def read_token():
49 """[summary]
50 Read the token from file 'vaulttoken'
51 [description]
52
53 Returns:
54 [type] string -- [description] the token
55 """
56 try:
57 f = open(VAULT_TOKEN_FILE, 'r')
58 root_token = f.read()
59 f.close()
60 return root_token
61 except Exception as e:
62 app.logger.error(e)
63 raise
64
```

### 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

```
65 def read_unseal_keys():
66 """[summary]
67 Read keys used to unseal the vault from file 'unsealkeys'
68 [description]
69
70 Returns:
71 [type] List -- [description] List of keys
72 """
73 try:
74 f = open(UNSEAL_KEYS_FILE, 'r')
75 unseal_keys = f.read().splitlines()
76 f.close()
77 return unseal_keys
78 except Exception as e:
79 app.logger.error(e)
80 raise
81
```

### 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

```
106 def seal_vault(client):
107 """[summary]
108 Seal the vault
109 [description]
110 This should be done to protect the vault while not using it
111 Arguments:
112 vault client {[type]} -- [description] vault client
113 """
114 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

93 def unseal_vault(client):
94 """[summary]
95 Unseal (open) the vault
96 [description]
97 This must be done prior to read contents from the vault.
98 Arguments:
99 vault client {[type]} -- [description] vault client
100 """
101 client.token = read_token()
102 # unseal the vault
103 unseal_keys = read_unseal_keys()
104 client.unseal_multi(unseal_keys)
105
```

### 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:

```
1 = {
2 'shares': fields.Integer,
3 'threshold' : fields.Integer
4 }
```

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

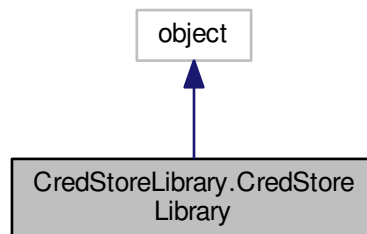
[summary]  
Main module.  
[description]  
The main module starts the web service

## Chapter 7

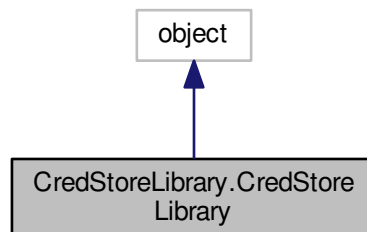
# Class Documentation

### 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)

14 def __init__(self):
15 #self._sut_path = os.path.join(os.path.dirname(__file__),
16 # '..', 'sut', 'login.py')
17 self._status = ''
18 self._data = ''
19
```

### 7.1.2 Member Function Documentation

#### 7.1.2.1 add\_a\_secret()

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

30 def add_a_secret(self, name, value):
31 url = 'http://127.0.0.1:5003/v1.0/secrets'
32 payload = {'name': name, 'value': value}
33 res = requests.post(url, json=payload)
34 json_data = json.loads(res.text)
35 self._status = json_data['code']
36
```

### 7.1.2.2 create\_app\_secret()

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

65 def create_app_secret(self, secretname, secretvalue, servicename):
66 url = 'http://127.0.0.1:5003/v1.0/appsecrets'
67 payload = {'name': secretname, 'value': secretvalue, 'service': servicename}
68 res = requests.post(url, json=payload)
69 json_data = json.loads(res.text)
70 self._status = json_data['code']
71
```

### 7.1.2.3 data\_should\_be()

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

25 def data_should_be(self, expected_data):
26 if expected_data != str(self._data):
27 raise AssertionError("Expected data to be '%s' but was '%s'."
28 % (expected_data, self._data))
29
```

### 7.1.2.4 delete\_a\_secret()

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

44 def delete_a_secret(self, secretname=None):
45 url = 'http://127.0.0.1:5003/v1.0/secrets/' + secretname
46 res = requests.delete(url)
47 json_data = json.loads(res.text)
48 self._status = json_data['code']
49
```

### 7.1.2.5 delete\_app\_secret()

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

72 def delete_app_secret(self, secretname, servicename):
73 url = 'http://127.0.0.1:5003/v1.0/appsecrets/' + secretname
74 payload = {'service': servicename}
75 res = requests.delete(url, json=payload)
76 json_data = json.loads(res.text)
77 self._status = json_data['code']
78
```

### 7.1.2.6 init\_a\_vault()

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

58 def init_a_vault(self, shares, threshold):
59 url = 'http://127.0.0.1:5003/v1.0/vaults'
60 payload = {'shares': shares, 'threshold': threshold}
61 res = requests.post(url, json=payload)
62 json_data = json.loads(res.text)
63 self._status = json_data['code']
64
```

### 7.1.2.7 read\_a\_secret()

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

50 def read_a_secret(self, secretname=None):
51 url = 'http://127.0.0.1:5003/v1.0/secrets/' + secretname
52 res = requests.get(url)
53 json_data = json.loads(res.text)
54 self._status = json_data['code']
55 if(self._status == http_code_ok):
56 self._data = json_data['data']['secret_value']
57
```

### 7.1.2.8 retrieve\_app\_secret\_id()

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

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

### 7.1.2.9 status\_should\_be()

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

20 def status_should_be(self, expected_status):
21 if expected_status != str(self._status):
22 raise AssertionError("Expected status to be '%s' but was '%s'."
23 % (expected_status, self._status))
24
```



## 7.1.2.10 update\_a\_secret()

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

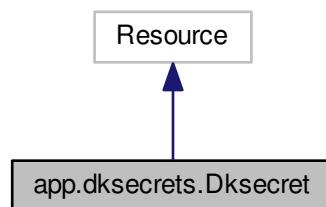
37 def update_a_secret(self, secretname, newvalue):
38 url = 'http://127.0.0.1:5003/v1.0/secrets/' + secretname
39 payload = {'value': newvalue}
40 res = requests.put(url, json=payload)
41 json_data = json.loads(res.text)
42 self._status = json_data['code']
43
```

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

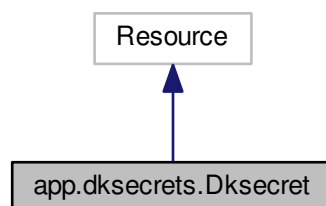
- [lib/CredStoreLibrary.py](#)

## 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 secret from docker swarm
Arguments:
 secret_name {[type]} -- [description] Name of secret

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

Raises:
 e -- [description]
```

```
142 def delete(self,secret_name):
143 '''[summary]
144 Remove secret from a service.
145 [description]
146 Remove secret from all container of a services. If there are no other services using the secret,
delete the secret from docker swarm
147 Arguments:
148 secret_name {[type]} -- [description] Name of secret
149
150 Returns:
151 Json [type] -- [description] Returned message
152
153 Raises:
154 e -- [description]
155 '''
156 json_body = request.json
157 service_name = json_body['service'] #Application/ service name
158
159 if DEBUG_MODE:
160 print('\nDELETE SERVICE SECRET')
161
162 try:
163 client = docker.APIClient(base_url='unix://var/run/docker.sock')
164 service = client.services(filters={"name" : service_name}).pop(0) # Get the service by name
165 except Exception as e:
166 app.logger.error(e)
167 resp = create_json_response(HTTP_CODE_BAD_REQUEST,'non_exist_service')
168 return resp
169
170 service_id = service['ID'] # get service_id
171 service_version = service['Version']['Index'] # get service_version
172
173 container_spec = service['Spec']['TaskTemplate']['ContainerSpec'] #get service container
specification
174 current_secrets = []
175 try:
176 current_secrets = container_spec['Secrets']
177 if DEBUG_MODE:
178 print('\nSecret list before removed: ', current_secrets)
179 print('1 item: ',current_secrets[0])
180 print('Type of current_secrets: ', type(current_secrets))
181 print('Type of item: ', type(current_secrets[0]))
```

```

182 print('Secret name of 1st item: ', current_secrets[0]['SecretName'])
183
184 update_current_secrets = [x for x in current_secrets if x['SecretName'] != secret_name]
185
186 if DEBUG_MODE:
187 print('\nSecret list after removed: ', update_current_secrets)
188
189 container_spec['Secrets'] = update_current_secrets
190 except Exception as e:
191 app.logger.error(e)
192 raise e
193
194 task_tmpl = docker.types.TaskTemplate(container_spec) # Create TaskTemplate from container
specification
195
196 if DEBUG_MODE:
197 print("\nTask template", task_tmpl.container_spec)
198
199 client.update_service(service=service_id,name=service_name,version=service_version,task_template=
task_tmpl) # Update service with new secret list
200
201 if DEBUG_MODE:
202 print("\nConfig of service after update:", client.inspect_service(service=service_name))
203
204 # Try to delete docker secret (if no other services are using it)
205 try:
206 delete_secret(secret_name)
207 except Exception as e:
208 app.logger.error(e)
209 pass
210
211 resp = create_json_response(HTTP_CODE_OK,'remove_dksecret_success')
212 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]

```

213 def get(self, secret_name):
214 '''[summary]
215 Retrieve id of a secret
216 [description]
217
218 Arguments:
219 secret_name {[type]} -- [description] Name of secret
220
221 Returns:
222 [type] -- [description]
223 '''
224 client = docker.DockerClient(base_url='unix://var/run/docker.sock')
225 if DEBUG_MODE:
226 print('\nRetrieve docker secret')
227
228 try:
229 secret_id = client.api.secrets(filters={"name" : secret_name})[0]['ID']
230
231 if DEBUG_MODE:
232 print('\nSecret id:', secret_id)
233 info = {'secret_id':secret_id}
234 resp = create_json_response(HTTP_CODE_OK,'secret_exists', additional_json =
info)

```

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

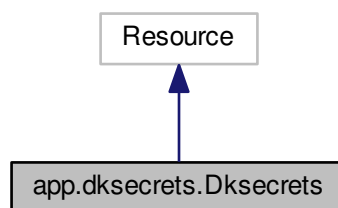
The documentation for this class was generated from the following file:

- [app/dksecrets.py](#)

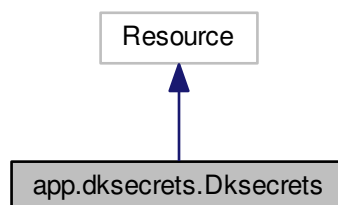
### 7.3 app.dksecrets.Dksecrets Class Reference

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 distribute the docker secret to the corresponding service

Returns:

```
 [type] Json object -- [description] Successful message or error message
```

```
73 def post(self):
74 '''[summary]
75 Creates a docker secret and distribute it to docker service
76 [description]
77 Input:
78 name -- secret's name
79 value -- secret's value
80 service -- service' name
81 Assuming that the service exists, this API creates a docker secret from the inputted name and
82 value, then distribute the docker secret to the corresponding service
83 Returns:
84 [type] Json object -- [description] Successful message or error message
85 '''
86 json_body = request.json
87
88 secret_name = json_body['name'] # Secret name
89 secret_value = json_body['value'] #Secret value
90 service_name = json_body['service'] #Application/ service name
91
92 # Verify query arguments
93 if(secret_name is None or secret_value is None or service_name is None or secret_name==' ' or
94 secret_value==' ' or service_name==' '):
95 resp = create_json_response(HTTP_CODE_BAD_REQUEST,'
96 bad_request_write_dksecret')
97 return resp
98
99 if DEBUG_MODE:
100 print('\nADD APPLICATION SECRET INTO DOCKER')
101
102 try:
103 client = docker.APIClient(base_url='unix://var/run/docker.sock')
104 service = client.services(filters={"name" : service_name}).pop(0) # Get the service by name
105 except Exception as e:
106 app.logger.error(e)
107 resp = create_json_response(HTTP_CODE_BAD_REQUEST,'non_exist_service')
108 return resp
109
110 # Create docker secret and add it to corresponding service
111 try:
112 secret_id = create_secret(secret_name, secret_value) # Create docker secret
113 except Exception as e:
114 app.logger.error(e)
```

```

113 resp = create_json_response(HTTP_CODE_BAD_REQUEST, 'existed_secret')
114 return resp
115
116 service_id = service['ID'] # get service_id
117 service_version = service['Version']['Index'] # get service_version
118
119 container_spec = service['Spec']['TaskTemplate']['ContainerSpec'] #get service container
specification
120 current_secrets = []
121 try:
122 current_secrets = container_spec['Secrets'] # get the service's current secret list (if
existed)
123 except:
124 pass
125
126 secret_list = [docker.types.SecretReference(secret_id, secret_name)] # Create list of
SecretReference
127 container_spec['Secrets']=current_secrets+secret_list # update list of secrets for the service
128 task_tmpl = docker.types.TaskTemplate(container_spec) # Create TaskTemplate from container
specification
129
130 if DEBUG_MODE:
131 print("Task template", task_tmpl.container_spec)
132
133 client.update_service(service=service_id, name=service_name, version=service_version, task_template=
task_tmpl) # Update service with new secret list
134
135 if DEBUG_MODE:
136 print("Config of service after update:", client.inspect_service(service=service_name))
137
138 resp = create_json_response(HTTP_CODE_CREATED, 'write_dksecret_success')
139 return resp
140

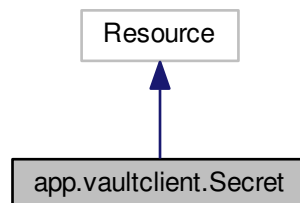
```

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

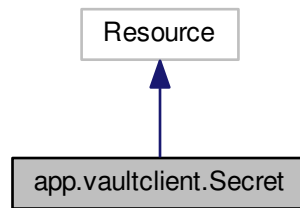
- [app/dksecrets.py](#)

## 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]

242 def delete(self, secret_name):
243 """[summary]
244 Remove a secret from the vault
245 [description]
246 """
247 if DEBUG_MODE:
248 print('\nDELETE SECRET FROM VAULT')
249
250 if(secret_name is None or secret_name==''): # verify parameters
251 resp = create_json_response(HTTP_CODE_BAD_REQUEST, '
bad_request_delete_secret') #'Lack of secret name'
252 return resp
253
254 # unseal the vault
255 client = init_client()
256 try:
257 unseal_vault(client)
258 except Exception as e:
259 app.logger.error(e)
260 resp = create_json_response(HTTP_CODE_SERVER_ERR, 'vault_not_initialized')
261 return resp
262
263 client.delete('secret/'+secret_name)
264 seal_vault(client)
265
266 resp = create_json_response(HTTP_CODE_OK, 'delete_secret_success') #'Delete
secret successfully
267 return resp
268

```

## 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

```
209 def get(self, secret_name):
210 """[summary]
211 Read a secret from the vault
212 [description]
213
214 Returns:
215 [type] json -- [description] a dictionary of all relevant information of the secret
216 """
217 if(secret_name is None or secret_name==''): # verify parameters
218 resp = create_json_response(HTTP_CODE_BAD_REQUEST, 'bad_request_read_secret'
219)
220 return resp
221
222 if DEBUG_MODE:
223 print('READ SECRET FROM VAULT')
224
225 # unseal the vault
226 client = init_client()
227 try:
228 unseal_vault(client)
229 except Exception as e:
230 app.logger.error(e)
231 resp = create_json_response(HTTP_CODE_SERVER_ERR, 'vault_not_initialized')
232 return resp
233
234 secret_values = client.read('secret/'+secret_name)
235 seal_vault(client)
236
237 if(secret_values is None): # If the required secret does not exist
238 resp = create_json_response(HTTP_CODE_NOT_FOUND, 'secret_not_exist')
239 return resp
240 else:
241 resp = create_json_response(HTTP_CODE_OK, 'read_secret_success',
242 additional_json =secret_values)
243 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]



```

269 def put(self, secret_name):
270 ''' [summary]
271 Update a secret in the vault
272 [description]
273
274 Arguments:
275 secret_name {[type]} -- [description] Name of secret
276
277 Returns:
278 [type] -- [description]
279 '''
280 if DEBUG_MODE:
281 print('\nUPDATE SECRET FROM VAULT')
282
283 json_body = request.json
284 secret_value = json_body['value']
285
286 if (secret_value is None or secret_value==''): # verify parameters
287 resp = create_json_response(HTTP_CODE_BAD_REQUEST, '
bad_request_update_secret') #'Lack of secret name'
288 return resp
289
290 # unseal the vault
291 client = init_client()
292 try:
293 unseal_vault(client)
294 except Exception as e:
295 app.logger.error(e)
296 resp = create_json_response(HTTP_CODE_SERVER_ERR, 'vault_not_initialized')
297 return resp
298
299 secret_values = client.read('secret/'+secret_name)
300
301 if (secret_values is None): # If the required secret does not exist
302 resp = create_json_response(HTTP_CODE_NOT_FOUND, 'secret_not_exist')
303 return resp
304 else:
305 client.write('secret/'+secret_name, secret_value=secret_value)#, lease='1h'
306 seal_vault(client)
307 resp = create_json_response(HTTP_CODE_OK, 'update_secret_success')
308 return resp

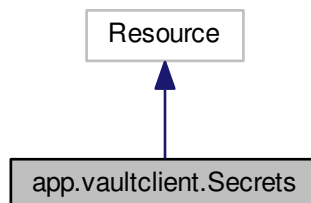
```

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

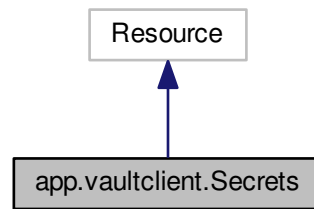
- [app/vaultclient.py](#)

## 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

```

173 def post(self):
174 """[summary]
175 Write or update a secret to the vault
176 [description]
177 Arguments:
178 name -- name of secret
179 value -- value of secret
180 """
181 json_body = request.json
182 secret_name = json_body['name']
183 secret_value = json_body['value']
184
185 if(secret_name is None or secret_value is None or secret_value==' ' or secret_name==' '): # verify
parameters
186 resp = create_json_response(HTTP_CODE_BAD_REQUEST,'bad_request_write_secret
')
187 return resp
188
189 if DEBUG_MODE:
190 print('\nADD SECRET INTO VAULT')
191 print('secret name:', secret_name)
192 print('secret_value:', secret_value)
193
194 client = init_client()
```

```
195 try:
196 unseal_vault(client)
197 except Exception as e:
198 app.logger.error(e)
199 resp = create_json_response(HTTP_CODE_SERVER_ERR, 'vault_not_initialized')
200 return resp
201
202 client.write('secret/'+secret_name, secret_value=secret_value)#, lease='1h'
203 seal_vault(client)
204
205 resp = create_json_response(HTTP_CODE_CREATED, 'write_secret_success')
206 return resp
207
```

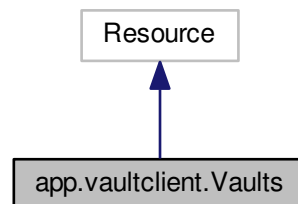
The documentation for this class was generated from the following file:

- [app/vaultclient.py](#)

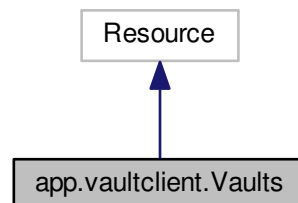
## 7.6 app.vaultclient.Vaults Class Reference

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 not

shares = The number of generated keys

threshold = The minimum number of generated keys needed to unseal the vault

```
119 def post(self):
120 """[summary]
121 Initialize a vault in the Vault Server (Credential Store)
122 [description]
123 A number of keys will be generated from the master key, then the master key is thrown away (The
124 Server will not store the key). The generated keys are kept by the Vault Client (Security Policy Manager)
125 shares = The number of generated keys
126 threshold = The minimum number of generated keys needed to unseal the vault
127 """
128 json_body = request.json
129 marshal_json = marshal(json_body, vault_init_params)
130
131 shares = marshal_json['shares']
132 threshold = marshal_json['threshold']
133
134 if DEBUG_MODE:
135 print('\nINIT VAULT:')
136 print('shares: ', shares)
137 print('threshold: ', threshold)
138
139 if(threshold>shares or (shares>=2 and threshold==1) or shares<=0 or threshold<=0):
140 resp = create_json_response(HTTP_CODE_BAD_REQUEST, '
init_vault_fail_due_to_parameter')
141 return resp
142
143 vault_exist = False
144 try:
145 client = init_client()
146 vault_exist = client.is_initialized()
147 except Exception as e:
148 app.logger.error(e)
149 resp = create_json_response(HTTP_CODE_SERVER_ERR, 'init_vault_fail') # Fail
150 to initialize vault
151 return resp
152
153 if(vault_exist): # if vault existed
154 resp = create_json_response(HTTP_CODE_CREATED, 'vault_existed')
155 return resp
156 else:
```

```
156 vault = client.initialize(shares,threshold)
157 root_token = vault['root_token']
158 unseal_keys = vault['keys']
159 # write root token into file
160 f = open(VAULT_TOKEN_FILE, 'w')
161 f.write(root_token)
162 f.close()
163
164 # write unseal_keys into file
165 f = open(UNSEAL_KEYS_FILE, 'w')
166 for key in unseal_keys:
167 f.write("%s\n" % key)
168 f.close()
169 resp = create_json_response(HTTP_CODE_CREATED, 'init_vault_success') #
Initialize vault successfully
170 return resp
171
```

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