fortiosapi

Release 1.0.3

May 27, 2020

Contents:

You will find here the overall documentation of the opensource fortiosapi module for Fortigate/Fortios devices Available on pypi.

class fortiosapi.**FortiOSAPI**
Global class / example for FortiOSAPI

check_session()
Helper fonction to check if the session on the FortiOSAPI object is valid :return:

True or raise NotLogged or InvalidLicense

static debug(*status*)
Set the debug to on to have all the debug information from the library You should add logging.getLogger('fortiosapi') to your log handler

Parameters **status** – on to set the log level to DEBUG

Returns None

delete(*path*, *name*, *vdom=None*, *mkey=None*, *parameters=None*, *data=None*)
Delete a pointed object in the cmdb.

Parameters

- **path** – first part of the Fortios API URL like
- **name** – https://myfgt:8040/api/v2/cmdb/<path>/<name>
- **data** – json containing the param/values of the object to be set
- **mkey** – when the cmdb object have a subtable mkey represent the subobject. It is optionnal at creation the code will find the mkey name for you.
- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add on parameters understood by the API call can be "&select=" for example

Returns A formatted json with the last response from the API

download(*path*, *name*, *vdom=None*, *mkey=None*, *parameters=None*)
Use the download call on the monitoring part of the API. Can get the config, logs etc..

Parameters

- **path** – first part of the Fortios API URL like
- **name** – https://myfgt:8040/api/v2/cmdb/<path>/<name>
- **mkey** – when the cmdb object have a subtable mkey represent the subobject. It is optionnal at creation the code will find the mkey name for you.
- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add parameters understood by the API call in json. Must set "destination": "file" and scope

Returns The file is part of the returned json

execute(*path*, *name*, *data*, *vdom=None*, *mkey=None*, *parameters=None*)
Execute is an action done on a running fortigate it is actually doing a post to the monitor part of the API we choose this name for clarity

Parameters

- **path** – first part of the Fortios API URL like

- **name** – https://myfgt:8040/api/v2/cmdb/<path>/<name>
- **data** – json containing the param/values of the object to be set
- **mkey** – when the cmdb object have a subtable mkey represent the subobject. It is optionnal at creation the code will find the mkey name for you.
- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add on parameters understood by the API call can be "&select=" for example

Returns A formatted json with the last response from the API

get (*path*, *name*, *vdom=None*, *mkey=None*, *parameters=None*)
Execute a GET on the cmdb (i.e. configuration part) of the Fortios API

Parameters

- **path** – first part of the Fortios API URL like
- **name** – https://myfgt:8040/api/v2/cmdb/<path>/<name>
- **mkey** – when the cmdb object have a subtable mkey represent the subobject. It is optionnal at creation the code will find the mkey name for you.
- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add on parameters understood by the API call can be "&select=" for example

Returns A formatted json with the last response from the API, values are in return['results']

get_mkey (*path*, *name*, *data*, *vdom=None*)

Parameters

- **path** –
- **name** –
- **data** –
- **vdom** –

Returns

get_mkeyname (*path*, *name*, *vdom=None*)

Parameters

- **path** –
- **name** –
- **vdom** –

Returns

get_version ()

Returns

https (*status*)
Allow to use http or https (default). HTTP is necessary to use the API on unlicensed/trial Fortigates

Parameters **status** – 'on' to use https to connect to API, anything else will http

Returns

license(*vdom='root'*)

license check and update:

- GET /api/v2/monitor/license/status
- If pending (exec update-now) with FortiGuard if invalid POST api/v2/monitor/system/fortiguard/update and do the GET again

  Convinient when Fortigate starts and license validity takes time.

Parameters **vdom** – root by default, can be global to do a global check

Returns True if license is valid at the end of the process

login(*host*, *username*, *password*, *verify=True*, *cert=None*, *timeout=12*, *vdom='global'*)

Parameters

- **host** –
- **username** –
- **password** –
- **verify** –
- **cert** –
- **timeout** –
- **vdom** –

Returns

logout()

Returns

monitor(*path*, *name*, *vdom=None*, *mkey=None*, *parameters=None*)

Execute a GET on the montioring part of the Fortios API :param path: first part of the Fortios API URL like :param name: https://myfgt:8040/api/v2/cmdb/<path>/<name> :param mkey: when the cmdb object have a subtable mkey represent the subobject.

It is optionnal at creation the code will find the mkey name for you.

Parameters

- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add on parameters understood by the API call can be "&select=" for example

Returns A formatted json with the last response from the API, values are in return['results']

move(*path*, *name*, *vdom=None*, *mkey=None*, *where=None*, *reference_key=None*, *parameters={}*)

Move an object in a cmdb table (firewall/policies for example). Usefull for reordering too :param path: first part of the Fortios API URL like :param name: https://myfgt:8040/api/v2/cmdb/<path>/<name> :param data: json containing the param/values of the object to be set :param mkey: when the cmdb object have a subtable mkey represent the subobject.

It is optionnal at creation the code will find the mkey name for you.

Parameters

- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add on parameters understood by the API call can be "&select=" for example
- **where** – the destination mkey in the table
- **reference_key** – the origin mkey in the table

Returns A formatted json with the last response from the API

post (*path*, *name*, *data*, *vdom=None*, *mkey=None*, *parameters=None*)

Execute a REST POST on the API. It will fail if the targeted object already exist. When post to the upper name/path the mkey is in the data. So we can ensure the data set is correctly filled in case mkey is passed.

Parameters

- **path** – first part of the Fortios API URL like
- **name** – https://myfgt:8040/api/v2/cmdb/<path>/<name>
- **data** – json containing the param/values of the object to be set
- **mkey** – when the cmdb object have a subtable mkey represent the subobject. It is optionnal at creation the code will find the mkey name for you.
- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add on parameters understood by the API call can be "&select=" for example

Returns A formatted json with the last response from the API

put (*path*, *name*, *vdom=None*, *mkey=None*, *parameters=None*, *data=None*)
Execute a REST PUT on the specified object with parameters in the data field as a json formatted field

Parameters

- **path** – first part of the Fortios API URL like
- **name** – https://myfgt:8040/api/v2/cmdb/<path>/<name>
- **data** – json containing the param/values of the object to be set
- **mkey** – when the cmdb object have a subtable mkey represent the subobject. It is optionnal at creation the code will find the mkey name for you.
- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add on parameters understood by the API call can be "&select=" for example

Returns A formatted json with the last response from the API

set (*path*, *name*, *data*, *mkey=None*, *vdom=None*, *parameters=None*)
Fortios API definition is at https://fndn.fortinet.net Function targeting config management. You pass the data of the part of cmdb you want to be set and the function will try POST and PUT to ensure your modification go through.

Parameters

- **path** – first part of the Fortios API URL like
- **name** – https://myfgt:8040/api/v2/cmdb/<path>/<name>

- **data** – json containing the param/values of the object to be set
- **mkey** – when the cmdb object have a subtable mkey represent the subobject. It is optionnal at creation the code will find the mkey name for you.
- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add on parameters understood by the API call can be "&select=" for example

Returns A formatted json with the last response from the API

setoverlayconfig(*yamltree*, *vdom=None*)
take a yaml tree with name:

path: mkey:

structure and recursively set the values. create a copy to only keep the leaf as node (table firewall rules etc Split the tree in 2 yaml objects and iterates) Update the higher level, up to tables as those config parameters may influence which param are allowed in the level 3 table :param yamltree: a yaml formatted string of the differents part of CMDB to be changed :param vdom: (optionnal) default is root, can use vdom=global to swtich to global settings. :return:

static ssh(*cmds*, *host*, *user*, *password=None*, *port=22*)
DEPRECATED use paramiko directly. Send a multi line string via ssh to the fortigate

Parameters

- **cmds** – multi line string with the Fortigate config cli
- **host** – ip/hostname of the fortigate interface
- **user/password** – fortigate admin user and password
- **port** – port 22 if not set or a port on which fortigate listen for ssh commands.

Returns The output of the console commands and raise exception if failed

tokenlogin(*host*, *apitoken*, *verify=True*, *cert=None*, *timeout=12*, *vdom='global'*)
if using apitoken method then login/passwd will be disabled

Parameters

- **host** –
- **apitoken** –
- **verify** –
- **cert** –
- **timeout** –
- **vdom** –

Returns

upload(*path*, *name*, *vdom=None*, *mkey=None*, *parameters=None*, *data=None*, *files=None*)
Upload a file (refer to the monitoring part), used for license, config, certificates etc.. uploads.

Parameters

- **path** – first part of the Fortios API URL like
- **name** – https://myfgt:8040/api/v2/cmdb/<path>/<name>
- **data** – json containing the param/values of the object to be set

- **mkey** – when the cmdb object have a subtable mkey represent the subobject. It is optionnal at creation the code will find the mkey name for you.
- **vdom** – the vdom on which you want to apply config or global for global settings
- **parameters** – Add on parameters understood by the API call can be "&select=" for example
- **files** – the file to be uploaded

Returns A formatted json with the last response from the API

CHAPTER 1

fortiosAPI Overview

Opensource python library to configure Fortigate/Fortios devices (Fortigate REST API)

1.1 Ready for config management.

Compare to the REST API there a few add-ons: In addition to get,put,post,delete methods there is a set which will try to post and if failing will put and collect the mkey directly. The lib will also find the mkey for you

1.2 Examples

You can find and propose examples here: https://github.com/fortinet-solutions-cse/fortiosapi-examples Separated to avoid cluttering those who integrate the fortiosapi module.

New overlay configuration

You now have an overlayconfig call which can be pass a complex configuration change in yaml. Including multiple endpoints (name/path) as the simple example below shows:

```
antivirus:
  profile:
    apisettree:
      "scan-mode": "quick"
      'http': {"options": "scan avmonitor",}
      "emulator": "enable"
firewall:
  policy:
    67:
      'name': "Testfortiosapi"
      'action': "accept"
      'srcintf': [{"name": "port1"}]
      'dstintf': [{"name": "port2"}]
```

(continues on next page)

(continued from previous page)

```
        'srcaddr': [{"name": "all"}]
        'dstaddr': [{"name": "all"}]
        'schedule': "always"
        'service': [{"name": "HTTPS"}]
        "utm-status": "enable"
        "profile-type": "single"
        'av-profile': "apisettree"
        'profile-protocol-options': "default"
        'ssl-ssh-profile': "certificate-inspection"
        'logtraffic': "all"
```

The behaviour is to change the parameters at the higher level in the CMDB tree first then do a serie of set on the tables.

Will fail if one of the set fails.

Order in the yaml is preserved.

1.3 Login methods

User/password

Token (api key) documented in the Fortigate API Spec that you can find if having an account on http://fndn.fortinet.net/

1.4 Multi vdom

In multi vdom environment use vdom=global in the API call. As it is a reserved word the API will switch to use the global=1 and take care of the differences in the repsonses.

1.5 Schema

There is a get_schema call and an example to get the schema of the differents methods to ease writting them.

1.6 License (5.6)

A rest call to check and force license validation check starting with 5.6 See license. usage of schema and mkey for every call for 5.6

License validity is now checked at login

1.7 Versions

1.8 Test driven development

In tests folder you will find a tox based set of tests as examples. The test_fortiosapi_virsh need you to have virsh access, especially to the console. This allow to perform actions automatically from the CLI and check API calls actual results. Other tests are welcomed.

1.9 Files upload/download

You will find the calls to exchange files (config, logs, licenses) with Fortigate in this LIB

1.10 Known Usage

Fortiosapi library is used in Home-Assistant, Fortinet Ansible modules and in Cloudify plugins.

Maintained mainly by Fortinet employees.

CHAPTER 2

Hacking development tips

2.1 Push to pypi

Follow: https://packaging.python.org/tutorials/packaging-projects/#description

Quick:

```
rm -rf dist/
python3 setup.py sdist bdist_wheel --universal
python3 -m twine upload dist/* --verbose --cert /etc/ssl/certs/
```

2.2 git tags

```
git tag -a v1.0.1 -m "GA release with Verify of SSL on by default"
git push origin --tags
```

2.3 Run only 1 unit test

```
cd .tox/py27
. bin/activate
python -m unittest test_fortiosapi_virsh.TestFortinetRestAPI.test_00login test_
↪fortiosapi_virsh.TestFortinetRestAPI.test_central_management
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index