pyopnsense Documentation

Matthew Treinish

Apr 03, 2022

Contents

Contents:

CHAPTER 1

pyopnsense

You can see the full rendered docs at: http://pyopnsense.readthedocs.io/en/latest/

A python API client for the OPNsense API. This provides a python interface for interacting with the OPNsense API.

1.1 Installation

pyopnsense is available via pypi so all you need to do is run:

```
pip install -U pyopnsense
```

to get the latest pyopnsense release on your system. If you need to use a development version of pyopnsense you can clone the repo and install it locally with:

```
git clone https://github.com/mtreinish/pyopnsense && pip install -e pyopnsense
```

which will install pyopnsense in your python environment in editable mode for development.

1.2 Usage

To use pyopnsense you need a couple pieces of information, the API key and the API secret. Both can be created/found from the OPNsense web UI by navigating to: *System->Access->Users* under *API keys*.

More information on this can be found in the OPNsense documentation: https://docs.opnsense.org/development/how-tos/api.html

Once you have the API key and API secret you can use pyopnsense to interact with your OPNsense installation. You can do this by passing your credentials to a client class. For example:

```
from pyopnsense import diagnostics
```

(continues on next page)

(continued from previous page)

```
api_key = 'XXXXXX'
api_secret = 'XXXXXXXXXXXXXXX'
opnsense_url = 'http://192.168.1.1/api'

netinsight_client = diagnostics.NetworkInsightClient(
    api_key, api_secret, opnsense_url)

print(netinsight_client.get_interfaces())
```

which will print a dictionary mapping physical devices to their interface label.

This same formula can be used to access each individual API endpoint you need to access. The basic structure of the library is setup to roughly mirror the endpoint tree of the OPNsense API. Each client module maps to the base endpoint and then there is a client class in those modules for the next level up off that.

You can find more detail on how to use the clients in the API reference documentation found here:

http://pyopnsense.readthedocs.io/en/latest/api.html

CHAPTER 2

API Reference

This document attempts to document the API provided by the pyopnsense library. It is a combination of autogenerated api documentation and usage explanations.

2.1 Instantiating Clients

All the firmware client classes are based off the base OPNClient class and are instantiated the same way. They require the same 3 mandatory arguments the `api_key`, the `api_secret`, and the `base_url`. With these 3 pieces of information you can instantiate any of the client classes. The *Usage* section of the README contains details on how to get the `api_key` and `api_secret` values. The `base_url` is the base api endpoint for your OPNsense installtion and is normally just http://\protect\T1\textdollarOPNsenseAddress/api where `$OPNsenseAddress` is the hostname or IP address of your OPNsense installation.

2.1.1 SSL Certificate Verification

By default the SSL certificate verification is disabled. This is to enable a working client out of the box. (since by default OPNsense is it's own CA, so it likely won't be in your system's CA bundle) The tradeoff here is obviously security. It's **strongly** recommended that you enable SSL verification once you start using the client for anything beyond basic testing. To do this the `verify` kwarg is used. This value gets passed directly to requests verify kwarg on the HTTP methods. You can set this to either `True` which will enable it and use your default system installed CA bundles, or the path to a CA certificate or bundle directory. More details can be found in the requests documentation here: http://docs.python-requests.org/en/master/user/advanced/#ssl-cert-verification

CHAPTER 3

Client Classes

3.1 Firmware API

class pyopnsense.firmware.**FirmwareClient**(*api_key*, *api_secret*, *base_url*, *verify_cert=False*)

Bases: pyopnsense.client.OPNClient

A client for interacting with the core/firmware endpoint.

Parameters

- **api_key** (*str*) – The API key to use for requests
- **api_secret** (*str*) – The API secret to use for requests
- **base_url** (*str*) – The base API endpoint for the OPNsense deployment

status()

Return the current firmware update status.

Returns A dict representing the current upgrade status for the OPNsense firmware.

Return type dict

upgrade(*upgrade_list=None*)

Issue an upgrade request.

Parameters **upgrade_list** (*list*) – The list of packages to upgrade. If none are specified it will issue a request to upgrade all packages.

3.2 Diagnostics API

class pyopnsense.diagnostics.**InterfaceClient**(*api_key*, *api_secret*, *base_url*, *verify_cert=False*)

Bases: pyopnsense.client.OPNClient

A client for interacting with the diagnostics/interface endpoint

Parameters

- **api_key** (*str*) – The API key to use for requests
- **api_secret** (*str*) – The API secret to use for requests
- **base_url** (*str*) – The base API endpoint for the OPNsense deployment

get_arp()
Get ARP table for router.

get_ndp()
Get NDP table for router.

class pyopnsense.diagnostics.**NetFlowClient**(*api_key*, *api_secret*, *base_url*, *verify_cert=False*)

Bases: pyopnsense.client.OPNClient

A client for interacting with the diagnostics/netflow endpoint.

Parameters

- **api_key** (*str*) – The API key to use for requests
- **api_secret** (*str*) – The API secret to use for requests
- **base_url** (*str*) – The base API endpoint for the OPNsense deployment

status()
Return the current netflow status.

Returns A dict representing the current status of netflow

Return type dict

class pyopnsense.diagnostics.**NetworkInsightClient**(*api_key*, *api_secret*, *base_url*, *verify_cert=False*)

Bases: pyopnsense.client.OPNClient

A client for interacting with the diagnostics/networkinsight endpoint.

Parameters

- **api_key** (*str*) – The API key to use for requests
- **api_secret** (*str*) – The API secret to use for requests
- **base_url** (*str*) – The base API endpoint for the OPNsense deployment

get_interfaces()
Return the available interfaces.

get_protocols()
Return the protocols.

get_services()
Return the available services.

get_timeserie()
Return the time serie.

class pyopnsense.diagnostics.**SystemHealthClient**(*api_key*, *api_secret*, *base_url*, *verify_cert=False*)

Bases: pyopnsense.client.OPNClient

A client for interacting with the diagnostics/systemhealth endpoint.

Parameters

- **api_key** (*str*) – The API key to use for requests
- **api_secret** (*str*) – The API secret to use for requests
- **base_url** (*str*) – The base API endpoint for the OPNsense deployment

get_health_data(*metric*, *start=0*, *stop=0*, *maxitems=1024*, *inverse=False*, *details=False*)
Return the health data.

get_health_list()
Return the health list.

3.3 Routes API

class pyopnsense.routes.**GatewayClient**(*api_key*, *api_secret*, *base_url*, *verify_cert=False*)
Bases: pyopnsense.client.OPNClient

A client for interacting with the routes/gateway endpoint.

Parameters

- **api_key** (*str*) – The API key to use for requests
- **api_secret** (*str*) – The API secret to use for requests
- **base_url** (*str*) – The base API endpoint for the OPNsense deployment

status()
Return the current gateways status.

Returns A dict representing the current status of gateways

Return type dict

Python Module Index

p

Index