pypingdom Documentation

Release 0.1.1

Paolo Sechi <sekipaolo@gmail.com>

Oct 18, 2018

Contents:

Python library for interacting with Pingdom services (REST API and maintenance windows).

CHAPTER 1

Features

- Support for Multi-User Authentication
- Check management: create, delete, update, list
- Maintenance windows: create, delete, list
- Fetching outage summaries

Warning: Since the Pingdom REST API don't support maintenance windows, we interact with the Website for it. Therefore this feature is highly fragile and can *break* at any moment due to frontend changes on the Pingdom website.

CHAPTER 2

Requirements

- Pingdom account
- requests (0.10.8 or newer)

CHAPTER 3

Installation

```
pip install pypingdom
```

CHAPTER 4

Usage

The *client* object will allow you to interact both with the REST API and the GUI (for maintenance windows).

```
>>> import pypingdom
>>> client = pypingdom.Client(username="username@example.com",
                              password="your_password",
                              apikey="your_api_key",
                              email="your_email")
```

the *email* parameter is required for Multiuser Authentication.

CHAPTER 5

Checks

Since Pingdom does not treat the check name as identifier (as we probably want to do) the client object will retrieve the check list from the API and cache it as a dictionary (check_name => check_instance). You can access it through the *checks* attribute:

```
>>> client.checks["my awesome check"]
pingdom.Check <1895866>
  autoresolve: 0
  alert_policy: 2118909
  name: example_com
  created: 1448565930
  lasterrortime: 1489325292
  resolution: 1
  lastresponsetime: 558
  lasttesttime: 1489847772
  alert_policy_name: Production Systems
  paused: False
  host: hostname.example.com
  acktimeout: 0
  ipv6: False
  use_legacy_notifications: False
  type: http
  tags: []
```

a better way to retrieve a check would be:

```
>>> client.get_check("my awesome check")
```

that will return None if the check doesn't exists

List checks with *production* and *frontend* tags:

```
>>> client.get_checks(filters={"tags": ["production", "frontend"]})
```

Create a check:

```
>>> check_definition = {
        "name": "My awesome check",
        "paused": True,
        "alert_policy": 201745,
        "type": "http",
        "host": "www.google.com",
        "url": "/",
        "requestheaders": {
            'XCustom': 'my header value'
        },
        "tags": [{"name": "pypingdom-test"}, {"name": "custom-tag"}],
        "encryption": False
    }
>>> client.create_check(check_definition)
```

Refers to this page for the list of options.

When you create or modify a check some related entity need to be referenced by id:

Integrations

To enable/disable an integration plugins (like webhooks) use the field *integrationids* (array with integer ids to set or "null" tring to remove it)

Alert policies

To bind an alerting policy use the field *alert_policy* (numeric id to set it or string "null" to disable alerts)

Update a check:

```
>>> client.update_check(check, {"paused": True})
```

this will return True if an effective change was sent to the API and False otherwise (useful for idempotent usage, like ansible modules)

Delete a check:

```
>>> client.delete_check(check)
```

CHAPTER 6

Maintenance windows

Retreive maintenance windows for production websites in the last 7 days:

```
>>> import datetime
>>> checks = client.get_checks(filters={"tags": ["production", "frontend"]})
>>> start = datetime.datetime.now() - datetime.timedelta(days=7)
>>> client.get_maintenances(filters={"checks": checks, "after": start})
```

Create a 1 hour maintenance window for production websites:

```
>>> start = datetime.datetime.now() + datetime.timedelta(minutes=10)
>>> end = start + datetime.timedelta(hours=1)

>>> window = client.create_maintenance({"checks": checks, "name": "pypingdom test␣
↪maintenance", "start": start, "stop": end})
```

Delete future maintenance windows:

```
>>> windows = client.get_maintenances(filters={"checks": checks, "after": datetime.
↪datetime.now()}):
>>> for m in maintenances:
    client.delete_maintenance(m)
```

CHAPTER 7

Reporting/summary

Retrieve average response time and uptime summaries:

```
>>> checkid = client.get_check("my awesome check")._id
>>> start = int(time.time()) - 30*24*60*60 # 30 days back
>>> end = time.time()
>>> client.get_summary_average(checkid, start, end, include_uptime="true")
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search