python-sigfoxapi Documentation

Release 0.0.1

Markus Juenemann

Jul 11, 2017

Contents:

python-sigfoxapi is a Python wrapper for the Sigfox backend REST API.

At this stage only features that are accessible with *LIMITED_ADMIN* permissions have been implemented as I personally don't have full access tp the REST-API.

- Groups (info, list).
- Device types (list, edit, errors, warnings, messages, disengage).
- Callbacks (list, new, delete, enable, disable, errors, downlink).
- Devices (info, list, tokenstate, messages, locations, errors, warnings, networkstate, message metrics, consumptions).
- Coverage (redundancy, predictions).
- Users (list)

For more details about the Sigfox backend REST API navigate to the *Group* page in the Sigfox backend web interface, select a group, click on *REST-API* and then on the *API documentation* link. The documentation is generated automatically and tailored to the access permission of the logged-in user.

CHAPTER 1

Example

The example retrieves information about a device.

```
>>> from sigfoxapi import Sigfox
>>> s = Sigfox('mylogin', 'mypassword')
>>> s.device_info('002C')
{
      "id" : "002C",
      "name" : "Labege 4",
      "type" : "4d3091a05ee16b3cc86699ab",
      "last" : 1343321977,
      "averageSignal": 8.065601,
      "averageSnr": 8.065601,
      "averageRssi": -122.56,
      "state": 0,
      "lat" : 43.45,
      "lng" : 1.54,
      "computedLocation": {
          "lat" : 43.45,
          "lng" : 6.54,
          "radius": 500
      },
      "activationTime": 1404096340556,
      "pac": "545CB3B17AC98BA4",
      "tokenType": "CONTRACT",
      "contractId": "7896541254789654aedfba4c",
      "tokenEnd": 1449010800000,
      "preventRenewal": false
 }
```

It is also possible to have the `Sigfox()` methods return objects instead of dictionaries by setting `sigfoxapi.RETURN_OBJECTS` to `True`.

```
>>> sigfoxapi.RETURN_OBJECTS = True
>>> device = s.device_info('002C')
>>> device.averageRssi
```

```
-122.56
>>> device.computedLocation.lat
43.45
```

CHAPTER 2

Documentation

The full documentation can be found at http://python-sigfoxapi.readthedocs.io/en/master/#.

python-sigfox API

Note: Currently only the features that are accessible as *LIMITED_ADMIN* are implemented.

This page describes the API of the **python-sigfox** package. All code examples have been copied from the official documentation.

The sigfoxapi module

`sigfoxapi.`**`DEBUG`** **= False**
Set to `True` to enable debugging.

`sigfoxapi.`**`IGNORE_SSL_VALIDATION`** **= False**
Set to `True` to ignore SSL validation problems.

`sigfoxapi.`**`RETURN_OBJECTS`** **= False**
Change to `True` to return objects instead of dictionaries. Returning objects allows to access fields in the `object.field` syntax instead of `dict['field']` which some people may prefer.

```
>>> sigfoxapi.RETURN_OJECTS = False
>>> group = s.group_info('489b848ee4b0ca4786945614')
>>> group['name']
Group 1
>>> sigfoxapi.RETURN_OJECTS = True
>>> group = s.group_info('489b848ee4b0ca4786945614')
>>> group.name
Group 1
```

Exceptions

class sigfoxapi.**SigfoxApiError**
Base exception for all errors.

```
>>> try:
...     s.group_info('does_not_exist')
... except SigfoxApiNotFound:
...     print('Not found')
... except SigfoxApiError:
...     print('Other Sigfox error')
```

class sigfoxapi.**SigfoxApiBadRequest**
Exception for HTTP error 400 (Bad Request).

Important: This exception will also be raised if the *before* or *since* arguments to some methods are set in a way that no results would be returned. For example I didn't send any messages before 01-May-2017 so searching for any will raise *SigfoxApiBadRequest* instead of returning an empty list!

```
>>> t = 1493560800    # Unix timestamp 01-May-2017 00:00:00
>>> try:
...     s.device_messages(SIGFOX_DEVICE_ID, before=t)
... except SigfoxApiBadRequest:
...     print("No Sigfox messages found before 01-May-2017")
```

---

class sigfoxapi.**SigfoxApiAuthError**
Exception for HTTP error 401 (Authentication Error).

class sigfoxapi.**SigfoxApiAccessDenied**
Exception for HTTP error 403 (Access Denied).

class sigfoxapi.**SigfoxApiNotFound**
Exception for HTTP error 404 (Not Found).

```
>>> try:
...     s.group_info('123456789012345678901234')
... except SigfoxApiNotFound:
...     print('Not found')
```

class sigfoxapi.**SigfoxApiServerError**
Exception for HTTP error 500 (Internal Server Error).

The Sigfox class

class sigfoxapi.**Sigfox**(*login*, *password*)
Interact with the Sigfox backend API.

Parameters

- **login** – Login as shown on the *Group* - *REST API* pacge of the Sigfox backend web interface.
- **password** – Password as shown on the *Group* - *REST API* pacge of the Sigfox backend web interface.

```
>>> s = Sigfox('1234567890abcdef', 'fedcba09876543221')
```

Note: Response paging has not been implemented yet, i.e. currently at most 100 results (the backend default) will be returned.

Users

user_list

`Sigfox.`**`user_list`**(*groupid*, ***kwargs*)

Lists all users registered with a role associated to a specific group.

Parameters

- **`groupid`** – The group identifier.
- **`**kwargs`** – Optional keyword arguments as described in the official documentation (*limit*, *offset*).

Returns List of dictionaries with user details as returned in the `data` field of the REST-API response.

Note: This method may result in multiple HTTP request to automatically iterate through paged responses.

List all users.

```
>>> s.user_list(groupid)
[ {"firstName" : "Michel",
   "lastName" : "Dupont",
   "email" : "michel.dupont@sigfox.com",
   "timezone" : "Europe/Paris",
   "creationTime" : 1392812363644,
   "creationDate" : "Wed Feb 19 13:19:23 CET 2014",
   "lastLogin" : 1448351837467,
   "lastLoginDate" : "Tue Nov 24 08:57:17 CET 2015",
   "userRoles" : [ {
     "group" : {
       "id" : "babecafebabecafebabecafe",
       "name" : "Root",
       "nameCI" : "root",
       "description" : "Master Group",
       "path" : [ ],
       "billable" : false
       },
     "customRole" : {
       "id" : "51d19e7ce4b067e859e4c2c1",
       "name" : "SUPPORT_CORP"
       }
     } ]
   },
   { ... } ]
```

Only list users #10 to #20.

```
>>> s.user_list(groupid, offset=10, limit=10)
```

Groups

group_info

Sigfox.**group_info**(*groupid*)

Get the description of a particular group.

Parameters **groupid** – The group identifier.

Example

```
>>> s.group_info('489b848ee4b0ca4786945614')
{
    "id":"489b848ee4b0ca4786945614",
    "name":"Group 1",
    "nameCI":"group 1",
    "description":"Group 1 description text",
    "path":[
        "51f13454bc54518c7bae7d4d",
        "50f13484b846618c7bae77b7"
    ],
    "billable":true,
    "bssId": "bss-48631656321"
}
```

group_list

Sigfox.**group_list**(***kwargs*)

Lists all children groups of your group.

Parameters ****kwargs** – Optional keyword arguments as described in the official documentation (*limit*, *offset*, *parentId*).

```
>>> s.group_list()
[
    {
        "id":"510b848ee4b0ca47869752b5",
        "name":"Group 1",
        "nameCI":"group 1",
        "description":"Group 1 description text",
        "path":[
             "51f13454bc54518c7bae7d4d",
             "50f13484b846618c7bae77b7"
        ],
        "billable":true,
        "bssId": "bss-48631656321"
        },
        { ... }
]
```

Device types

devicetype_edit

Sigfox.**devicetype_edit**(*devicetypeid*, *changes*)
Edit a device type.

Parameters **params** – Dictionary of the format described in the official documentation

```
>>> changes = {
...     "name" : "dtname",
...     "description" : "the description",
...     "keepAlive" : 3000,
...     "alertEmail" : "alert@email.com",
...     "payloadType" : "None",
...     "downlinkMode" : 0,
...     "downlinkDataString" : "deadbeefcafebabe",
... }
>>> s.devicetype_edit(changes)
```

Note: The *changes* parameter may already contain the devicetype identifier (`id`) but it will be overridden by *devicetypeid*.

devicetype_list

Sigfox.**devicetype_list**()
Lists all device types available to your group.

..note:: `includeSubGroups` and `contractInfoId` are currently not supported.

```
>>> s.devicetype_list()
[
    {
        "id" : "4d3091a05ee16b3cc86699ab",
        "name" : "Sigfox test device",
        "group" : "4d39a4c9e03e6b3c430e2188",
        "description" : "Little things in the black boxes",
        "keepAlive" : 7200,
        "payloadType" : "None",
        "contract" : "523b1d10d777d3f5ae038a02"
    },
    { ... }
]
```

devicetype_errors

Sigfox.**devicetype_errors**(*devicetypeid*, ***kwargs*)
Get the communication down events for devices belonging to a device type.

Parameters

- **devicetypeid** – The device type identifier.
- ****kwargs** – Optional keyword arguments as described in the official documentation (*limit*, *offset*, *since* and *before*)

```
>>> s.devicetype_errors('5256c4d6c9a871b80f5a2e50')
[
    {
        "deviceId" : "0235",
        "time" : 1381410600026,
        "message" : "No message received since 2013-10-08 15:36:21",
        "severity" : "ERROR",
        "deviceTypeId" : "5256c4d6c9a871b80f5a2e50",
        "callbacks" : [
            {
                "url" : "http://host/path?id=0235&time=1381410600",
                "status" : 600,
                "info" : "Connection refused: host/path"
            },
            {
                "subject" : "some subject",
                "message" : "some messages",
                "status" : 200
            }
        ]
    },
    { ... }
]
```

devicetype_warnings

Sigfox.**devicetype_warnings**(*devicetypeid*, ***kwargs*)

Get the network issues events that were sent for devices belonging to a device type.

Parameters

- **devicetypeid** – The device type identifier.
- ****kwargs** – Optional keyword arguments as described in the official documentation (*limit*, *offset*, *since* and *before*)

See *Sigfox.devicetype_errors()* for example output.

devicetype_messages

Sigfox.**devicetype_messages**(*devicetypeid*, ***kwargs*)

Get the messages that were sent by all the devices of a device type.

Parameters

- **devicetypeid** – The device type identifier.
- ****kwargs** – Optional keyword arguments as described in the official documentation (*limit*, *offset*, *since* and *before*)

```
>>> s.devicetype_messages('5256c4d6c9a871b80f5a2e50')
[
    {
        "device" : "002C",
        "time" : 1343321977,
        "data" : "3235353843fc",
        "snr" : "38.2",
```

```
        "computedLocation": {
          "lat" : 43.45,
          "lng" : 6.54,
          "radius": 500
        },
        "linkQuality" : "GOOD",
        "downlinkAnswerStatus" : {
            "data" : "1511000a00007894"
         }
    },
    { ... }
]
```

Note: The `snr` field is a string and not a float. This is what the REST-API returns.

devicetype_disengage

`Sigfox.`**`devicetype_disengage`**(*devicetypeid*)

Disengage sequence number check for next message of each device of the device type.

Parameters **`devicetypeid`** – The device type identifier.

```
>>> s.devicetype_disengage('5256c4d6c9a871b80f5a2e50')
None
```

Devices

device_list

`Sigfox.`**`device_list`**(*devicetypeid*, ***kwargs*)

Lists the devices associated to a specific device type.

Parameters

- **`devicetypeid`** – The device type identifier.
- **`**kwargs`** – Optional keyword arguments as described in the official documentation (*snr*, *limit*, *offset*).

```
>>> s.device_list('4d3091a05ee16b3cc86699ab', srn=1)          # 1 = for SNR values␣
↪from 0 to 10 dB
[
    {
        "id" : "002C",
        "name" : "Labege 4",
        "type" : "4d3091a05ee16b3cc86699ab",
        "last" : 1343321977,
        "averageSignal": 8.065601,
        "averageSnr": 8.065601,
        "averageRssi": -122.56,
        "state": 0,
        "lat" : 43.45,
        "lng" : 1.54,
        "computedLocation": {
```

```
            "lat" : 43.45,
            "lng" : 6.54,
            "radius": 500
        },
        "activationTime": 1404096340556,
        "pac": "545CB3B17AC98BA4",
        "tokenType": "CONTRACT",
        "contractId": "7896541254789654aedfba4c",
        "tokenEnd": 1449010800000,
        "preventRenewal": false
    },
    { ... }
]
```

device_info

Sigfox.**device_info**(*deviceid*)
Get information about a device.

Parameters **deviceid** – The device identifier.

```
>>> s.device_info('002C')
{
    "id" : "002C",
    "name" : "Labege 4",
    "type" : "4d3091a05ee16b3cc86699ab",
    "last" : 1343321977,
    "averageSignal": 8.065601,
    "averageSnr": 8.065601,
    "averageRssi": -122.56,
    "state": 0,
    "lat" : 43.45,
    "lng" : 1.54,
    "computedLocation": {
        "lat" : 43.45,
        "lng" : 6.54,
        "radius": 500
    },
    "activationTime": 1404096340556,
    "pac": "545CB3B17AC98BA4",
    "tokenType": "CONTRACT",
    "contractId": "7896541254789654aedfba4c",
    "tokenEnd": 1449010800000,
    "preventRenewal": false
}
```

device_tokenstate

Sigfox.**device_tokenstate**(*deviceid*)
Get information about a device's token

Parameters **deviceid** – The device identifier.

```
>>> s.device_tokenstate('4d3091a05ee16b3cc86699ab')
{
```

```
    "code" : 1,
    "detailMessage" : "Off contract",
    "tokenType": "CONTRACT",
     "contractId": "7896541254789654aedfba4c",
     "tokenEnd": 1418673953200,
}
```

device_messages

Sigfox.**device_messages**(*deviceid*, ***kwargs*)

Get the messages that were sent by a device.

Parameters

- **deviceid** – The device identifier.
- ****kwargs** – Optional keyword arguments as described in the official documentation (*before*, *since*, *limit*, *offset*).

```
>>> s.device_messages('4d3091a05ee16b3cc86699ab', since=time.time()-60*60*24)
↪# Last 24 hours
[
    {
        "device" : "002C",
        "time" : 1343321977,
        "data" : "3235353843fc",
        "snr" : "38.2",
        "computedLocation": {
            "lat" : 43.45,
            "lng" : 6.54,
            "radius": 500
        },
        "linkQuality" : "GOOD",
        "downlinkAnswerStatus" : {
            "data" : "1511000a00007894"
        }
    },
    { ... }
]
```

Note: The `snr` field is a string and not a float. This is what the REST-API returns.

device_locations

Sigfox.**device_locations**(*deviceid*, ***kwargs*)

Get the messages location.

Parameters

- **deviceid** – The device identifier.
- ****kwargs** – Optional keyword arguments as described in the official documentation (*before*, *since*, *limit*, *offset*).

```
>>> s.device_locations('4d3091a05ee16b3cc86699ab', since=time.time()-60*60*24)
↪# Last 24 hours)
[
    {
        "time" : 1343321977000,
        "valid" : true,
        "lat" : 42.4631156,
        "lng" : 1.5652321,
        "radius" : 360,
    },
    { ... }
]
```

device_errors

Sigfox.**device_errors**(*deviceid*, ***kwargs*)

Get the communication down events for a device.

Parameters

- **deviceid** – The device identifier.
- ****kwargs** – Optional keyword arguments as described in the official documentation (*before*, *since*, *limit*, *offset*).

```
>>> s.device_errors('4830')
[
    {
        "deviceId" : "4830",
        "time" : 1381300600026,
        "message" : "No message received since 2013-10-08 15:36:21",
        "severity" : "ERROR",
        "deviceTypeId" : "5256c4d6c9a871b80f5a2e50",
        "callbacks" : [
            {
                "url" : "http://host/path?id=4830&time=1381300600",
                "status" : 200
            },
            {
                "subject" : "some subject",
                "message" : "some messages",
                "status" : 200
            }
        ]
    },
    { ... }
]
```

device_warnings

Sigfox.**device_warnings**(*deviceid*, ***kwargs*)

Get the network issues events that were sent for a device

Parameters

- **deviceid** – The device identifier.

- ****kwargs** – Optional keyword arguments as described in the official documentation (*before*, *since*, *limit*, *offset*).

```
>>> s.device_warnings('4830')
[
    {
        "deviceIds" : [ "0235", "023A", "4830" ],
        "time" : 1381410600026,
        "message" : "Sigfox network experiencing issues [SIC]",
        "severity" : "WARN",
        "deviceTypeId" : "5256c4d6c9a871b80f5a2e50",
        "callbacks" : [
            {
                "url" : "http://host/path?id=4830&time=1381410600",
                "status" : 600,
                "info" : "Connection refused: host/path"
            },
            {
                "subject" : "some subject",
                "message" : "some messages",
                "status" : 200
            }
        ]
    },
    { ... }
]
```

devices_networkstate

Sigfox.**device_networkstate**(*deviceid*)

Return the network status for a specific device.

Parameters **deviceid** – The device identifier.

```
>>> s.device_networkstate('4830')
{
    "networkStatus" : "NOK"
}
```

device_messagemetrics

Sigfox.**device_messagemetrics**(*deviceid*)

Returns the total number of device messages for one device, this day, this week and this month.

Parameters **deviceid** – The device identifier.

```
>>> s.device_messagemetrics('4830')
{
    "lastDay": 47,
    "lastWeek": 276,
    "lastMonth": 784
}
```

device_consumptions

Sigfox.**device_consumptions**(*deviceid*, *year*)
Get a Device's consumptions for a year.

Parameters

- **deviceid** – The device identifier.
- **year** – The year, e.g. 2017.

```
>>> s.device_consumptions('4830', 2017)
{
    "consumption": {
        "id" : "4830_2017",
        "consumptions": [
            {
                "frameCount": 12,
                "downlinkFrameCount": 3
            },
            { ... }
        ]
    }
}
```

Each entry in consumption is the data for one day, starting with the 1st of January.

Callbacks

callback_new

Sigfox.**callback_new**(*devicetypeid*, *callbacks*)
Create new callbacks.

Parameters **callbacks** – List of dictionaries as described in the official documentation.

Example

```
>>> new_callbacks = [
...     {
...         "channel" : "URL",
...         "callbackType" : 0,
...         "callbackSubtype" : 2,
...         "url" : "http://myserver.com/sigfox/callback",
...         "httpMethod" : "POST",
...         "enabled" : true,
...         "sendDuplicate" : false,
...         "sendSni": false,
...         "payloadConfig" : "var1::bool:1",
...         "bodyTemplate" : "device : {device} / {customData#var1}",
...         "headers" : {
...          "time" : "{time}"
...         },
...         "contentType" : "text/plain"
...     },
...     {
...         "channel" : "BATCH_URL",
...         "callbackType" : 0,
```

```
...         "callbackSubtype" : 2,
...         "url" : "http://myserver.com/sigfox/callback/batch",
...         "linePattern" : "{device};{data};",
...         "enabled" : true,
...         "sendDuplicate" : false,
...         "sendSni": false
...     }
... ]
>>> s.callback_new('5256c4d6c9a871b80f5a2e50', new_callbacks)
```

callback_list

Sigfox.**callback_list**(*devicetypeid*)
List the callbacks for a device type.

Parameters **devicetypeid** – The device type identifier.

```
>>> s.callback_list('5256c4d6c9a871b80f5a2e50')
[
    {
        "id" : "deadbeeffacecafebabecafe"
        "channel" : "URL",
        "callbackType" : 0,
        "payloadConfig" : "int1::uint:8 int2::uint:8",
        "callbackSubtype" : 0,
        "urlPattern" : "http://myserver.com/sigfox/callback",
        "httpMethod" : "POST",
        "headers" : { "key1" : "value1",
                      "key2" : "value2" },
        "enabled" : true,
        "sendDuplicate" : false,
        "dead":false,
        "downlinkHook":false
    },
    { ... }
]
```

callback_delete

Sigfox.**callback_delete**(*devicetypeid*, *callbackid*)
Delete a callback.

Parameters

- **devicetypeid** – The device type identifier.
- **callbackid** – The callback identifier.

```
>>> s.callback_delete('5256c4d6c9a871b80f5a2e50', 'deadbeeffacecafebabecafe')
```

callback_enable

Sigfox.**callback_enable**(*devicetypeid*, *callbackid*)
Enable a callback.

Parameters

- **devicetypeid** – The device type identifier.
- **callbackid** – The callback identifier.

```
>>> s.callback_enable('5256c4d6c9a871b80f5a2e50', 'deadbeeffacecafebabecafe')
```

callback_disable

Sigfox.**callback_disable**(*devicetypeid*, *callbackid*)

Disable a callback.

Parameters

- **devicetypeid** – The device type identifier.
- **callbackid** – The callback identifier.

```
>>> s.callback_disable('5256c4d6c9a871b80f5a2e50', 'deadbeeffacecafebabecafe')
```

callback_downlink

Sigfox.**callback_downlink**(*devicetypeid*, *callbackid*)

Select a downlink callback.

Parameters

- **devicetypeid** – The device type identifier.
- **callbackid** – The callback identifier.

```
>>> s.callback_downlink('5256c4d6c9a871b80f5a2e50', 'deadbeeffacecafebabecafe')
```

callback_errors

Sigfox.**callback_errors**(***kwargs*)

Returns device messages where at least one callback has failed.

Parameters ****kwargs** – Optional keyword arguments as described in the official documentation (*limit*, *offset*, *since*, *before*, *hexId*, *deviceTypeId*, *groupId*).

Coverage

coverage_redundancy

Sigfox.**coverage_redundancy**(*lat*, *lng*, *mode='INDOOR'*)

Get base station redundancy for a given latitude and longitude.

Parameters

- **lat** – The decimal latitude.
- **lng** – The decimal longitude.
- **mode** – Can be either `INDOOR` or `OUTDOOR`.

```
>>> s.coverage_redundancy(43.415, 1.9693, mode='OUTDOOR')
{
    "redundancy":3
}
```

coverage_predictions

Sigfox.**coverage_predictions**(*lat*, *lng*, *mode='INDOOR'*)

Get coverage levels for a given latitude and longitude.

Parameters

- **lat** – The decimal latitude.
- **lng** – The decimal longitude.
- **mode** – Can be either INDOOR, OUTDOOR or UNDERGROUND.

The return value contains the margins values (dB) for redundancy level 1, 2 and 3.

```
>>> s.coverage_preedictions(43.415, 1.9693)
{
    'margins': [48, 20, 7]
}
```

Indices and tables

- genindex
- modindex
- search

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index