

---

# **python-sigfoxapi Documentation**

***Release 0.0.1***

**Markus Juenemann**

**Jul 11, 2017**



---

## Contents:

---

<b>1</b>	<b>Example</b>	<b>3</b>
<b>2</b>	<b>Documentation</b>	<b>5</b>
2.1	python-sigfox API . . . . .	5
2.1.1	The sigfoxapi module . . . . .	5
2.1.2	Exceptions . . . . .	6
2.1.3	The Sigfox class . . . . .	6
2.1.4	Users . . . . .	6
2.1.5	Groups . . . . .	7
2.1.6	Device types . . . . .	8
2.1.7	Devices . . . . .	11
2.1.8	Callbacks . . . . .	16
2.1.9	Coverage . . . . .	18
2.2	Indices and tables . . . . .	19
<b>3</b>	<b>Indices and tables</b>	<b>21</b>



**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 to 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
```



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_OJETS = False
>>> group = s.group_info('489b848ee4b0ca4786945614')
>>> group['name']
Group 1
>>> sigfoxapi.RETURN_OJETS = 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.**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* page of the Sigfox backend web interface.
- **password** – Password as shown on the *Group - REST API* page 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" : "deadbeefcafebab",
... }
>>> 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', snr=1)           # 1 = for SNR values
↳ from 0 to 10 dB
[
  {
    "id" : "002C",
    "name" : "Labège 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" : "Labège 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",
    "deviceId" : "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",
    "deviceId" : "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, deviceId, 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_predictions(43.415, 1.9693)
{
  'margins': [48, 20, 7]
}
```

## Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)





## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



## C

callback\_delete() (sigfoxapi.Sigfox method), 17  
callback\_disable() (sigfoxapi.Sigfox method), 17  
callback\_downlink() (sigfoxapi.Sigfox method), 18  
callback\_enable() (sigfoxapi.Sigfox method), 17  
callback\_errors() (sigfoxapi.Sigfox method), 18  
callback\_list() (sigfoxapi.Sigfox method), 16  
callback\_new() (sigfoxapi.Sigfox method), 16  
coverage\_predictions() (sigfoxapi.Sigfox method), 18  
coverage\_redundancy() (sigfoxapi.Sigfox method), 18

## D

DEBUG (in module sigfoxapi), 5  
device\_consumptions() (sigfoxapi.Sigfox method), 15  
device\_errors() (sigfoxapi.Sigfox method), 13  
device\_info() (sigfoxapi.Sigfox method), 12  
device\_list() (sigfoxapi.Sigfox method), 11  
device\_locations() (sigfoxapi.Sigfox method), 13  
device\_messagemetrics() (sigfoxapi.Sigfox method), 15  
device\_messages() (sigfoxapi.Sigfox method), 12  
device\_networkstate() (sigfoxapi.Sigfox method), 15  
device\_tokenstate() (sigfoxapi.Sigfox method), 12  
device\_warnings() (sigfoxapi.Sigfox method), 14  
devicetype\_disengage() (sigfoxapi.Sigfox method), 11  
devicetype\_edit() (sigfoxapi.Sigfox method), 8  
devicetype\_errors() (sigfoxapi.Sigfox method), 9  
devicetype\_list() (sigfoxapi.Sigfox method), 9  
devicetype\_messages() (sigfoxapi.Sigfox method), 10  
devicetype\_warnings() (sigfoxapi.Sigfox method), 10

## G

group\_info() (sigfoxapi.Sigfox method), 7  
group\_list() (sigfoxapi.Sigfox method), 8

## I

IGNORE\_SSL\_VALIDATION (in module sigfoxapi), 5

## R

RETURN\_OBJECTS (in module sigfoxapi), 5

## S

Sigfox (class in sigfoxapi), 6  
SigfoxApiAccessDenied (class in sigfoxapi), 6  
SigfoxApiAuthError (class in sigfoxapi), 6  
SigfoxApiError (class in sigfoxapi), 6  
SigfoxApiNotFound (class in sigfoxapi), 6  
SigfoxApiServerError (class in sigfoxapi), 6

## U

user\_list() (sigfoxapi.Sigfox method), 6