
Kiefer Documentation

Release 0.2

Andy Goldschmidt

May 30, 2015

1	Installation	3
1.1	Installing from PyPI	3
1.2	Installing from source	3
2	Authentication	5
2.1	Getting the access token	5
2.2	Refreshing the access token	6
2.3	Existing access token	6
2.4	Storing tokens	6
3	Client	7
3.1	Usage	7
3.2	Why do I get an authorization_error?	8
4	API Reference	9
4.1	Authentication	9
4.2	Client	10
	Python Module Index	15

Contents:

Installation

Installing *kiefer* is pretty simple. You can either install it from PyPI or from source.

1.1 Installing from PyPI

```
pip install kiefer
```

1.2 Installing from source

```
git clone git@github.com:andygoldschmidt/kiefer.git
cd kiefer/
python setup.py install
```

Authentication

The UP API offers OAuth 2.0 authentication, which requires user interaction to allow access of third-party services. *kiefer* aims to make this process as easy as possible.

Before you start, go to the [Jawbone developer page](#) and create an account and an app, if you haven't done yet. Be sure to use an *HTTPS Redirect URI*.

2.1 Getting the access token

Before we can start with the authentication process, we need to create a JSON config file. This config needs values for the following keys:

- `client_id`
- `client_secret`
- `redirect_uri`
- `scope`

You can copy the `client_id` and `client_secret` from your app page. For a list of available scopes have a look at [Jawbone's authentication docs](#).

Note: You can find a config file template [here](#).

After creating the config we can start authentication:

```
from kiefer.auth import KieferAuth

auth = KieferAuth('PATH_TO_CONFIG_FILE')
access_token, refresh_token = auth.get_access_token()
```

When calling the `get_access_token()` method, you will be prompted to visit an authorization url. After permitting access copy-paste the **full redirect URL** into the command line prompt and you're done.

Your access token and a refresh token are returned. The refresh token is needed to issue a new access token to you, when it expires.

2.2 Refreshing the access token

Currently, UP API access tokens are valid for 1 year. After that time, the token expires and you need to get a new one using your refresh token.

If you have a valid access token and forgot your refresh token, you can retrieve it using:

```
refresh_token = auth.get_refresh_token()
```

After your access token is expired you can issue a new one like that:

```
access_token, refresh_token = auth.refresh_access_token()
```

Note: The `KieferAuth` object needs to have a valid refresh token set. You can either put the refresh token in your config (see *Storing tokens*) or you can set it using the `set_refresh_token()` method.

2.3 Existing access token

If you already have a valid access token, you can skip the whole authentication process and use the [Client](#) directly.

2.4 Storing tokens

Although there is no option to save the access token and refresh token yet, you can put them in your config file and `KieferAuth` will recognize them during initialization:

```
from kiefer.auth import KieferAuth

auth = KieferAuth('PATH_TO_CONFIG_FILE')
access_token = auth.access_token
refresh_token = auth.refresh_token
```

Client

Before you start using the client, make sure you have a valid access token. If you don't have one yet, see [Authentication](#) for details.

The first step is to initialize the client with your access token:

```
from kiefer.client import KieferClient

client = KieferClient('YOUR_ACCESS_TOKEN')
```

The client provides endpoints for these event types:

- band events
- body events
- heart rate
- custom events
- goals
- meals
- moods
- moves
- settings
- sleeps
- time zone
- user information
- workouts

3.1 Usage

kiefer supports most of the endpoints provided by the [Jawbone UP API](#). The usage is straight forward:

```
# Get information about authorized user
client.get_user_information()

# Add a new body event (weight)
client.add_body_event(title='New body event', weight=85.0, body_fat=22.5, share=True)
```

```
# Delete a sleep event
client.delete_sleep('sleep_id')

# Some endpoints support updates as well, e.g. workouts:
client.update_workout('workout_id', calories=500)
```

For a full list of supported endpoints, please refer to the [API Reference](#).

3.2 Why do I get an `authorization_error`?

You only can use endpoints that are covered by the scope of your access token. If you need more rights, change your scopes accordingly and request a new access token.

API Reference

4.1 Authentication

class `kiefer.auth.KieferAuth` (*config_path*)

Takes care of authentication with the Jawbone UP API. The provided config file needs to include these 4 values:

- `client_id`
- `client_secret`
- `redirect_uri`
- `scope`

Additionally, `access_token` and `refresh_token` will be recognized during initialization, if provided. This is helpful to retrieve the refresh token or to refresh your access token.

Parameters `config_path` – `str`, path to config file.

get_access_token ()

Use this method to retrieve your access token.

Returns `access_token`

get_refresh_token ()

Get refresh token.

Returns `refresh token`

refresh_access_token ()

Refresh your (expired) access token.

The *KieferAuth* instance needs a valid access token and refresh token. Use `set_access_token()` and `set_refresh_token()` for that. Alternatively, you can add `access_token` and `refresh_token` as keys to your config file.

Returns `access token`

set_access_token (*token*)

Set access token.

Parameters `token` – `str`

set_refresh_token (*token*)

Set refresh token.

Parameters `token` – `str`

4.2 Client

`class kiefer.client.KieferClient (access_token)`

Client class for the Jawbone UP API.

Parameters `access_token` – Your access token for the UP API.

`add_body_event (**kwargs)`

Add body event (weight).

Values for `**kwargs`:

- `title`: str
- `weight`: float (required)
- `body_fat`: float
- `lean_mass`: float
- `bmi`: float
- `note`: str
- `time_created`: int
- `tz`: str
- `share`: bool

`add_meal (**kwargs)`

Add a new meal.

Possible values for `kwargs` are:

- `note`: str
- `sub_type`: int (1 = Breakfast, 2 = Lunch, 3 = Dinner)
- `place_lat`: float
- `place_lon`: float
- `place_acc`: float
- `place_name`: str
- `time_created`: int
- `tz`: str
- `items`: dict (For details refer to: https://jawbone.com/up/developer/endpoints/meals#post_meal)

`add_mood (**kwargs)`

Add a new mood.

Possible values for `kwargs` are:

- `title`: str
- `sub_type`: int (1 = Amazing, 2 = Pumped UP, 3 = Energized, 8 = Good, 4 = Meh, 5 = Dragging, 6 = Exhausted, 7 = Totally Done)
- `time_created`: int
- `tz`: str
- `share`: bool

add_sleep (**kwargs)

Add a new sleep.

Possible values for kwargs are:

- time_created: int
- time_completed: int
- tz: str
- share: bool

add_workout (**kwargs)

Add a new workout.

Possible values for kwargs are:

- sub_type: int (refer to https://jawbone.com/up/developer/endpoints/workouts#post_workout for list of values)
- time_created: int
- time_completed: int
- place_lat: float
- place_lon: float
- place_acc: float
- place_name: str
- tz: str
- share: bool
- calories: int
- distance: int
- image_url: str
- intensity: int (1 = easy, 2 = moderate, 3 = intermediate, 4 = difficult, 5 = hard)

delete_body_event (xid)

Delete a body event.

Parameters **xid** – str, id of body event

delete_meal (xid)

Delete a meal.

Parameters **xid** – str, meal id

delete_mood (xid)

Delete a mood.

Parameters **xid** – str, mood id

delete_sleep (xid)

Delete a sleep.

Parameters **xid** – str, sleep id

delete_workout (xid)

Delete a workout.

Parameters **xid** – “str, workout id

get_band_events ()
Get list of band hardware events.

get_body_event (*xid*)
Get a single body event.

Parameters **xid** – str, id of body event

get_body_events (***kwargs*)
Get list of body events.

get_custom_events (***kwargs*)
Get list of custom/generic events.

get_goals ()
Get list of goals.

get_heart_rates (***kwargs*)
Get list of heart rates.

get_meal (*xid*)
Get a single meal.

Parameters **xid** – str, meal id

get_meals (***kwargs*)
Get list of meals.

get_mood (*xid*)
Get a single mood.

Parameters **xid** – str, id of mood

get_moods (***kwargs*)
Get list of moods.

get_move (*xid*)
Get a single move.

Parameters **xid** – str, move id

get_move_graph (*xid*)
Get graph of a single move.

Parameters **xid** – str, move id

get_move_ticks (*xid*)
Get ticks of a single move.

Parameters **xid** – str, move id

get_moves (***kwargs*)
Get list of moves.

get_settings ()
Retrieve user settings.

get_sleep (*xid*)
Get a single sleep.

Parameters **xid** – str, id of sleep

get_sleep_graph (*xid*)
Get graph of a single sleep.

Parameters **xid** – str, sleep id

get_sleep_phases (*xid*)
Get sleep phases of a single sleep.

Parameters **xid** – str, sleep id

get_sleeps (***kwargs*)
Get list of sleeps.

get_timezone (***kwargs*)
Get user time zone.

get_trends (***kwargs*)
Get trends.

get_user_friends ()
Get list of the user’s friends.

get_user_information ()
Get basic information of the user.

get_workout (*xid*)
Get a single workout.

Parameters **xid** – str, id of workout

get_workout_graph (*xid*)
Get graph for a single workout.

Parameters **xid** – str, workout id

get_workout_ticks (*xid*)
Get ticks for a single workout.

Parameters **xid** – str, workout id

get_workouts (***kwargs*)
Get list of workouts.

update_goal (***kwargs*)
Creates or updates an user’s goal(s).
Possible values for kwargs are:

- move_steps: int
- sleep_total: int
- body_weight: float
- body_weight_intent: int (0 = lose, 1 = maintain, 2 = gain)

update_meal (*xid*, ***kwargs*)
Updates an existing meal.
Refer to [add_meal\(\)](#) for a list of keyword arguments.

Parameters **xid** – str, id of meal

update_workout (*xid*, ***kwargs*)
Updates an existing workout.
Refer to [add_workout\(\)](#) for a list of keyword arguments.

Parameters **xid** – str, workout id

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

k

`kiefer.auth`, 9

`kiefer.client`, 10

A

`add_body_event()` (kiefier.client.KieferClient method), 10
`add_meal()` (kiefier.client.KieferClient method), 10
`add_mood()` (kiefier.client.KieferClient method), 10
`add_sleep()` (kiefier.client.KieferClient method), 10
`add_workout()` (kiefier.client.KieferClient method), 11

D

`delete_body_event()` (kiefier.client.KieferClient method), 11
`delete_meal()` (kiefier.client.KieferClient method), 11
`delete_mood()` (kiefier.client.KieferClient method), 11
`delete_sleep()` (kiefier.client.KieferClient method), 11
`delete_workout()` (kiefier.client.KieferClient method), 11

G

`get_access_token()` (kiefier.auth.KieferAuth method), 9
`get_band_events()` (kiefier.client.KieferClient method), 11
`get_body_event()` (kiefier.client.KieferClient method), 12
`get_body_events()` (kiefier.client.KieferClient method), 12
`get_custom_events()` (kiefier.client.KieferClient method), 12
`get_goals()` (kiefier.client.KieferClient method), 12
`get_heart_rates()` (kiefier.client.KieferClient method), 12
`get_meal()` (kiefier.client.KieferClient method), 12
`get_meals()` (kiefier.client.KieferClient method), 12
`get_mood()` (kiefier.client.KieferClient method), 12
`get_moods()` (kiefier.client.KieferClient method), 12
`get_move()` (kiefier.client.KieferClient method), 12
`get_move_graph()` (kiefier.client.KieferClient method), 12
`get_move_ticks()` (kiefier.client.KieferClient method), 12
`get_moves()` (kiefier.client.KieferClient method), 12
`get_refresh_token()` (kiefier.auth.KieferAuth method), 9
`get_settings()` (kiefier.client.KieferClient method), 12
`get_sleep()` (kiefier.client.KieferClient method), 12
`get_sleep_graph()` (kiefier.client.KieferClient method), 12
`get_sleep_phases()` (kiefier.client.KieferClient method), 12
`get_sleeps()` (kiefier.client.KieferClient method), 13
`get_timezone()` (kiefier.client.KieferClient method), 13

`get_trends()` (kiefier.client.KieferClient method), 13
`get_user_friends()` (kiefier.client.KieferClient method), 13
`get_user_information()` (kiefier.client.KieferClient method), 13
`get_workout()` (kiefier.client.KieferClient method), 13
`get_workout_graph()` (kiefier.client.KieferClient method), 13
`get_workout_ticks()` (kiefier.client.KieferClient method), 13
`get_workouts()` (kiefier.client.KieferClient method), 13

K

`kiefier.auth` (module), 9
`kiefier.client` (module), 10
`KieferAuth` (class in `kiefier.auth`), 9
`KieferClient` (class in `kiefier.client`), 10

R

`refresh_access_token()` (kiefier.auth.KieferAuth method), 9

S

`set_access_token()` (kiefier.auth.KieferAuth method), 9
`set_refresh_token()` (kiefier.auth.KieferAuth method), 9

U

`update_goal()` (kiefier.client.KieferClient method), 13
`update_meal()` (kiefier.client.KieferClient method), 13
`update_workout()` (kiefier.client.KieferClient method), 13