fitbit_api

Release 1.0.0

Liz LeCrone

Feb 11, 2020

CONTENTS:

CHAPTER

ONE

QUICKSTART

1.1 Register a Fitbit app

If you haven't already done so, register your app at https://dev.fitbit.com/apps/new. This will give you your app's Client ID and Client Secret.

1.2 Install fitbit_api

fitbit_api can be installed using pip or by cloning the repository here.

```
pip install fitbit_api
```

1.3 Example

Once you have a Fitbit app registered, you need to initiate an OAuth2 dance to authorize pulling data. For this you'll need your Client ID from the Fitbit app and your Callback URI, which should be the address of whatever page you want to redirect to after authentication.

```
from fitbit_api import FitbitClient

CLIENT_ID = 'your Fitbit app Client ID'
CLIENT_SECRET = 'your Fitbit app Client Secret'
CALLBACK_URI = 'your Fitbit app Callback URI'

my_auth_url = FitbitClient.OAuth2_step_one(
    CLIENT_ID,
    CALLBACK_URI,
)
```

This will return an address that the user should go to to authorize your app to access their Fitbit data. Once they log in, the page will redirect to the callback URL you indicated when you registered your app, appended with a unique code. `FitbitClient.OAuth2_step_two` will take the entire URL as an argument and initialize a client. Note that each user will need a separate client.

```
callback_uri_with_unique_code = 'address the user was redirected to'

client = FitbitClient.OAuth2_step_two(
    CLIENT_ID,
```

(continues on next page)

(continued from previous page)

```
    CLIENT_SECRET,
    CALLBACK_URI,
    callback_uri_with_unique_code,
)
```

You now have a Fitbit client! You can now access that user's Fitbit data.

```
client.get_recent_activities()
```

Go to *Class API* for the full list of available methods. Check out *Authentication* for a deeper dive on authentication methods.

CHAPTER

TWO

AUTHENTICATION

There are two ways to initialize a client with fitbit_api:

1. *OAuth2 Dance*
2. *Create Using Token*

2.1 OAuth2 Dance

Initiate the OAuth2 dance by calling the class method `FitbitClient.OAuth2_step_one`.

```
from fitbit_api import FitbitClient

CLIENT_ID = 'your Fitbit app Client ID'
CLIENT_SECRET = 'your Fitbit app Client Secret'
CALLBACK_URI = 'your Fitbit app Callback URI'

authorization_uri = FitbitClient.OAuth2_step_one(
    client_id=CLIENT_ID,
    redirect_uri=CALLBACK_URI,
)
```

The user will need to navigate to `authorization_uri` to grant your app access to their data. Once the user logs into their Fitbit account and grants permission, they'll be redirected to a new URI with an appended code. You'll need this for `FitbitClient.OAuth2_step_two`.

```
redirect_uri = 'address the user was redirected to'

def new_token_callback(token):
    # save the token somewhere so the user doesn't have to log in again.

client = FitbitClient.OAuth2_step_two(
    client_id=CLIENT_ID,
    client_secret=CLIENT_SECRET,
    callback_uri=CALLBACK_URI,
    redirect_uri=redirect_uri,
    new_token_callback=new_token_callback,
)
```

Fitbit provides refresh tokens, so when this token expires, the client will automatically use the refresh token to request a new one.

If you want to save the session token, `FitbitClient.OAuth2_step_two` takes an optional argument called `new_token_callback`. The callback will also be called on subsequent tokens created whenever the session token

expires.

2.2 Create Using Token

If you have an existing token, you can initialize a client without the user needing to log in again.

```
from fitbit_api import FitbitClient

CLIENT_ID = 'your Fitbit app Client ID'
CALLBACK_URI = 'your Fitbit app Callback URI'

def new_token_callback(token):
    # save subsequent tokens somewhere so the user doesn't have to log in again.

token = {}  # some token retrieved from some place
client = FitbitClient.create_using_token(
    token=token,
    client_id=CLIENT_ID,
    callback_uri=CALLBACK_URI,
    new_token_callback=new_token_callback
)
```

Fitbit provides refresh tokens, so when this token expires, the client will automatically use the refresh token to request a new one.

If you want to save subsequent tokens, *`FitbitClient.create_using_token`* takes an optional argument called `new_token_callback`.

CHAPTER

THREE

EXCEPTIONS

exception fitbit_api.exceptions.**FitbitException**

All other exceptions subclass this FitbitException.

exception fitbit_api.exceptions.**InsufficientScope**

Thrown when your client was not initialized with the necessary scopes to make this request

exception fitbit_api.exceptions.**RateLimitException**(*retry_after*, **args*, ***kwargs*)

Thrown when you've exceeded the limit

Fitbit only allows 150 requests per hour per client.

```
try:
    client.get_foods_goal()
except RateLimitException as e:
    print('Seconds until we can try again:' e.retry_after)
```

CHAPTER

FOUR

CLASS API

`fitbit_api.ALL_SCOPES = ['activity', 'nutrition', 'heartrate', 'location', 'nutrition', 'p`

All available scopes for initializing Fitbit client

class `fitbit_api.FitbitClient`(*session*, *client_secret*)

A Fitbit API client

classmethod create_using_token(*token*, *client_id*, *client_secret*, *callback_uri*, *scope=None*, *new_token_callback=None*)

Initialize a client using an existing token.

Parameters

- **token** (*dict*) – A token provided by Fitbit. You can generate one with *OAuth2 Dance*.
- **client_id** (*str*) – Client ID provided when you register your app with Fitbit.
- **client_secret** (*str*) – Client Secret provided when you register your app with Fitbit.
- **callback_uri** (*str*) – Callback URI you provided when you registered your app with Fitbit.
- **scope** (*List[str], optional*) – List of permissions requested for this client. If none, uses *ALL_SCOPES*.
- **new_token_callback** (*func, optional*) – Callback function for saving subsequent tokens provided by Fitbit.

Returns A FitbitClient instance

classmethod OAuth2_step_one(*client_id*, *callback_uri*, *scope=None*)

Initiate the OAuth2 dance.

Parameters

- **client_id** (*str*) – Client ID provided when you register your app with Fitbit.
- **callback_uri** (*str*) – Callback URI you provided when you registered your app with Fitbit.
- **scope** (*List[str], optional*) – List of permissions requested for this client. If none, uses *ALL_SCOPES*.

Returns Authorization URI where the user will need to navigate to grant your app access to their data.

classmethod OAuth2_step_two(*client_id*, *client_secret*, *callback_uri*, *redirect_uri*, *scope=None*, *new_token_callback=None*)

Initialize a client by completing the OAuth2 dance.

Parameters

- **client_id** (*str*) – Client ID provided when you register your app with Fitbit.
- **client_secret** (*str*) – Client Secret provided when you register your app with Fitbit.
- **callback_uri** (*str*) – Callback URI you provided when you registered your app with Fitbit.
- **redirect_uri** (*str*) – URI the user was redirected to after they granted your app permission.
- **scope** (*List[str], optional*) – List of permissions requested for this client. If none, uses *ALL_SCOPES*.
- **new_token_callback** (*func, optional*) – Callback function for saving the initial and all subsequent refresh tokens provided by Fitbit.

Returns A FitbitClient instance

get_activities_by_date(*date*, ***kwargs*)

Get Activity Summary by Date

Retrieves a summary and list of a user's activities and activity log entries for a given day.

Parameters **date** (*datetime*) – A datetime object.

get_activities_resource_by_date_range(*resource_path*, *base_date*, *end_date*, ***kwargs*)

Get Activity Resource by Date Range

Returns activities time series data in the specified range for a given resource.

Parameters

- **resource_path** (*string*) – The resource-path; see options in the Resource Path Options section in the full documentation. Possible values: ['calories', 'caloriesBMR', 'steps', 'distance', 'floors', 'elevation', 'minutesSedentary', 'minutesLightlyActive', 'minutesFairlyActive', 'minutesVeryActive', 'activityCalories']
- **base_date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.

get_activities_tracker_resource_by_date_range(*resource_path*, *base_date*, *end_date*, ***kwargs*)

Get Activity Tracker Resource by Date Range Time Series

Returns time series data in the specified range for a given resource.

Parameters

- **resource_path** (*string*) – The resource-path; see options in the Resource Path Options section in the full documentation. Possible values: ['calories', 'caloriesBMR', 'steps', 'distance', 'floors', 'elevation', 'minutesSedentary', 'minutesLightlyActive', 'minutesFairlyActive', 'minutesVeryActive', 'activityCalories']
- **base_date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.

get_activities_resource_by_date_period(*resource_path*, *date*, *period*, ***kwargs*)

Get Activity Time Series

Returns time series data in the specified range for a given resource in the format requested using units in the unit system that corresponds to the Accept-Language header provided.

Parameters

- **resource_path** (*string*) – The resource-path; see options in the Resource Path Options section in the full documentation. Possible values: ['calories', 'caloriesBMR', 'steps', 'distance', 'floors', 'elevation', 'minutesSedentary', 'minutesLightlyActive', 'minutesFairlyActive', 'minutesVeryActive', 'activityCalories']
- **date** (*datetime*) – A datetime object.
- **period** (*string*) – The range for which data will be returned. Options are 1d, 7d, 30d, 1w, 1m, 3m, 6m, 1y, or max.

get_activities_tracker_resource_by_date_period(*resource_path*, *date*, *period*, ***kwargs*)

Get Activity Time Series

Returns time series data in the specified range for a given resource in the format requested using units in the unit system that corresponds to the Accept-Language header provided.

Parameters

- **resource_path** (*string*) – The resource-path; see options in the Resource Path Options section in the full documentation. Possible values: ['calories', 'caloriesBMR', 'steps', 'distance', 'floors', 'elevation', 'minutesSedentary', 'minutesLightlyActive', 'minutesFairlyActive', 'minutesVeryActive', 'activityCalories']
- **date** (*datetime*) – A datetime object.
- **period** (*string*) – The range for which data will be returned. Options are 1d, 7d, 30d, 1w, 1m, 3m, 6m, 1y, or max.

get_activities_resource_by_date_range_intraday(*resource_path*, *base_date*, *end_date*, *detail_level*, ***kwargs*)

Get Activity Intraday Time Series

Returns the Activity Intraday Time Series for a given resource in the format requested.

Parameters

- **resource_path** (*string*) – The resource-path; see options in the Resource Path Options section in the full documentation. Possible values: ['calories', 'steps', 'distance', 'floors', 'elevation']
- **base_date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.
- **detail_level** (*string*) – Number of data points to include. Either 1min or 15min. Optional.

get_activities_resource_by_date_intraday(*resource_path*, *date*, *detail_level*, ***kwargs*)

Get Intraday Time Series

Returns the Intraday Time Series for a given resource in the format requested.

Parameters

- **resource_path** (*string*) – The resource-path; see options in the Resource Path Options section in the full documentation. Possible values: ['calories', 'steps', 'distance', 'floors', 'elevation']
- **date** (*datetime*) – A datetime object.
- **detail_level** (*string*) – Number of data points to include. Either 1min or 15min. Optional.

get_activities_resource_by_date_range_time_series_intraday(*resource_path*, *date*, *end_date*, *detail_level*, *start_time*, *end_time*, ***kwargs*)

Get Activity Intraday Time Series

Returns the Intraday Time Series for a given resource in the format requested.

Parameters

- **resource_path** (*string*) – The resource-path; see options in the Resource Path Options section in the full documentation. Possible values: ['calories', 'steps', 'distance', 'floors', 'elevation']
- **date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.
- **detail_level** (*string*) – Number of data points to include. Either 1min or 15min.
- **start_time** (*string*) – The start of the period in the format HH:mm.
- **end_time** (*string*) – The end of the period in the format HH:mm.

get_activities_resource_by_date_time_series_intraday(*resource_path*, *date*, *detail_level*, *start_time*, *end_time*, ***kwargs*)

Get Intraday Time Series

Returns the Intraday Time Series for a given resource in the format requested.

Parameters

- **resource_path** (*string*) – The resource-path; see options in the Resource Path Options section in the full documentation. Possible values: ['calories', 'steps', 'distance', 'floors', 'elevation']
- **date** (*datetime*) – A datetime object.
- **detail_level** (*string*) – Number of data points to include. Either 1min or 15min.
- **start_time** (*string*) – The start of the period in the format HH:mm.
- **end_time** (*string*) – The end of the period in the format HH:mm.

add_activities_log(*activity_id*, *manual_calories*, *start_time*, *duration_millis*, *date*, *distance*, *activity_name=None*, *distance_unit=None*, ***kwargs*)

Log Activity

The Log Activity endpoint creates log entry for an activity or user's private custom activity using units in the unit system which corresponds to the Accept-Language header provided (or using optional custom distanceUnit) and get a response in the format requested.

Parameters

- **activity_id** (*int*) – The ID of the activity, directory activity or intensity level activity.
- **manual_calories** (*int*) – Calories burned that are manaully specified. Required with activityName must be provided.
- **start_time** (*string*) – Activity start time. Hours and minutes in the format HH:mm:ss.
- **duration_millis** (*int*) – Duration in milliseconds.

- **date** (*datetime*) – A datetime object.
- **distance** (*int*) – Distance is required for logging directory activity in the format X.XX and in the selected distanceUnit.
- **activity_name** (*string, optional*) – Custom activity name. Either activityId or activityName must be provided.
- **distance_unit** (*int, optional*) – Distance measurement unit. Steps units are available only for Walking (activityId=90013) and Running (activityId=90009) directory activities and their intensity levels.

get_activities_log (***kwargs*)

Get Lifetime Stats

Updates a user's daily activity goals and returns a response using units in the unit system which corresponds to the Accept-Language header provided.

delete_activities_log (*activity_log_id*, ***kwargs*)

Delete Activity Log

Deletes a user's activity log entry with the given ID.

Parameters **activity_log_id** (*int*) – The id of the activity log entry.

get_activities_log_list (*sort*, *offset*, *limit*, *before_date=None*, *after_date=None*, ***kwargs*)

Get Activity Log List

Retreives a list of user's activity log entries before or after a given day with offset and limit using units in the unit system which corresponds to the Accept-Language header provided.

Parameters

- **sort** (*string*) – The sort order of entries by date asc (ascending) or desc (descending).
- **offset** (*int*) – The offset number of entries.
- **limit** (*int*) – The maximum number of entries returned (maximum;100).
- **before_date** (*datetime, optional*) – A datetime object.
- **after_date** (*datetime, optional*) – A datetime object.

get_activities_t_c_x (*log_id*, *include_partial_t_c_x=None*, ***kwargs*)

Get Activity TCX

Retreives the details of a user's location and heart rate data during a logged exercise activity.

Parameters

- **log_id** (*string*) – The activity's log ID.
- **include_partial_t_c_x** (*bool, optional*) – Include TCX points regardless of GPS data being present

get_activities_types (***kwargs*)

Browse Activity Types

Retreives a tree of all valid Fitbit public activities from the activities catelog as well as private custom activities the user created in the format requested.

get_activities_type_detail (*activity_id*, ***kwargs*)

Get Activity Type

Returns the detail of a specific activity in the Fitbit activities database in the format requested. If activity has levels, it also returns a list of activity level details.

Parameters **activity_id** (*string*) – The activity ID.

get_frequent_activities (***kwargs*)
Get Frequent Activities

Retreives a list of a user's frequent activities in the format requested using units in the unit system which corresponds to the Accept-Language header provided.

get_recent_activities (***kwargs*)
Get Recent Activity Types

Retreives a list of a user's recent activities types logged with some details of the last activity log of that type using units in the unit system which corresponds to the Accept-Language header provided.

get_favorite_activities (***kwargs*)
Get Favorite Activities

Returns a list of a user's favorite activities.

delete_favorite_activities (*activity_id*, ***kwargs*)
Delete Favorite Activity

Removes the activity with the given ID from a user's list of favorite activities.

Parameters **activity_id** (*string*) – The ID of the activity to be removed.

add_favorite_activities (*activity_id*, ***kwargs*)
Add Favorite Activity

Adds the activity with the given ID to user's list of favorite activities.

Parameters **activity_id** (*string*) – The encoded ID of the activity.

get_activities_goals (*period*, ***kwargs*)
Get Activity Goals

Retreives a user's current daily or weekly activity goals using measurement units as defined in the unit system, which corresponds to the Accept-Language header provided.

Parameters **period** (*string*) – daily or weekly.

add_update_activities_goals (*period*, *type*, *value*, ***kwargs*)
Update Activity Goals

Updates a user's daily or weekly activity goals and returns a response using units in the unit system which corresponds to the Accept-Language header provided.

Parameters

- **period** (*string*) – daily or weekly.
- **type** (*string*) – goal type
- **value** (*string*) – goal value

get_body_fat_by_date (*date*, ***kwargs*)
Get Body Fat Logs

Retreives a list of all user's body fat log entries for a given day in the format requested.

Parameters **date** (*datetime*) – A datetime object.

get_body_fat_by_date_period (*date*, *period*, ***kwargs*)
Get Body Fat Logs

Retreives a list of all user's body fat log entries for a given day in the format requested.

Parameters

- **date** (*datetime*) – A datetime object.
- **period** (*string*) – The range for which data will be returned. Options are 1d, 7d, 30d, 1w, 1m, 3m, 6m, 1y, or max.

get_body_fat_by_date_range(*base_date*, *end_date*, ***kwargs*)

Get Body Fat Logs

Retreives a list of all user's body fat log entries for a given day in the format requested.

Parameters

- **base_date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.

add_body_fat_log(*fat*, *date*, *time*, ***kwargs*)

Log Body Fat

Creates a log entry for body fat and returns a response in the format requested.

Parameters

- **fat** (*int*) – Body fat in the format of X.XX in the unit system that corresponds to the Accept-Language header provided.
- **date** (*datetime*) – A datetime object.
- **time** (*string*) – Time of the measurement in hours and minutes in the format HH:mm:ss that is set to the last second of the day if not provided.

delete_body_fat_log(*body_fat_log_id*, ***kwargs*)

Delete Body Fat Log

Deletes a user's body fat log entry with the given ID.

Parameters **body_fat_log_id** (*int*) – The ID of the body fat log entry.

get_body_resource_by_date_period(*resource_path*, *date*, *period*, ***kwargs*)

Get Body Time Series

Returns time series data in the specified range for a given resource in the format requested using units in the unit system that corresponds to the Accept-Language header provided.

Parameters

- **resource_path** (*string*) – The resource path, which incudes the bmi, fat, or weight options. Possible values: ['bmi', 'fat', 'weight']
- **date** (*datetime*) – A datetime object.
- **period** (*string*) – The range for which data will be returned. Options are 1d, 7d, 30d, 1w, 1m, 3m, 6m, 1y, or max.

get_body_resource_by_date_range(*resource_path*, *base_date*, *end_date*, ***kwargs*)

Get Body Time Series

Returns time series data in the specified range for a given resource in the format requested using units in the unit system that corresponds to the Accept-Language header provided.

Parameters

- **resource_path** (*string*) – The resource path, which incudes the bmi, fat, or weight options. Possible values: ['bmi', 'fat', 'weight']

- **base_date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.

get_body_goals (*goal_type*, ***kwargs*)
Get Body Goals

Retreives a user's current body fat percentage or weight goal using units in the unit systems that corresponds to the Accept-Language header providedin the format requested.

Parameters **goal_type** (*string*) – weight or fat.

update_body_fat_goal (*fat*, ***kwargs*)
Update Body Fat Goal

Updates user's fat percentage goal.

Parameters **fat** (*string*) – Target body fat percentage; in the format X.XX.

update_weight_goal (*start_date*, *start_weight*, *weight=None*, ***kwargs*)
Update Weight Goal

Updates user's fat percentage goal.

Parameters

- **start_date** (*datetime*) – A datetime object.
- **start_weight** (*string*) – Weight goal start weight; in the format X.XX, in the unit systems that corresponds to the Accept-Language header provided.
- **weight** (*string, optional*) – Weight goal target weight; in the format X.XX, in the unit systems that corresponds to the Accept-Language header provided; required if user doesn't have an existing weight goal.

get_weight_by_date (*date*, ***kwargs*)
Get Weight Logs

Retreives a list of all user's body weight log entries for a given day using units in the unit systems which corresponds to the Accept-Language header provided.

Parameters **date** (*datetime*) – A datetime object.

get_weight_by_date_period (*date*, *period*, ***kwargs*)
Get Body Fat Logs

Retreives a list of all user's body weight log entries for a given day in the format requested.

Parameters

- **date** (*datetime*) – A datetime object.
- **period** (*string*) – The range for which data will be returned. Options are 1d, 7d, 30d, 1w, 1m, 3m, 6m, 1y, or max.

get_weight_by_date_range (*base_date*, *end_date*, ***kwargs*)
Get Body Fat Logs

Retreives a list of all user's body fat log entries for a given day in the format requested.

Parameters

- **base_date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.

add_weight_log(*weight*, *date*, *time=None*, ***kwargs*)
Log Weight

Creates log entry for a body weight using units in the unit systems that corresponds to the Accept-Language header provided and gets a response in the format requested.

Parameters

- **weight** (*int*) – Weight in the format of X.XX.
- **date** (*datetime*) – A datetime object.
- **time** (*string, optional*) – Time of the measurement; hours and minutes in the format of HH:mm:ss, which is set to the last second of the day if not provided.

delete_weight_log(*body_weight_log_id*, ***kwargs*)
Delete Weight Log

Deletes a user's body weight log entrywith the given ID.

Parameters **body_weight_log_id** (*int*) – The ID of the body weight log entry.

get_devices(***kwargs*)
Get Devices

Returns a list of the Fitbit devices connected to a user's account.

get_alarms(*tracker_id*, ***kwargs*)
Get Alarms

Returns alarms for a device

Parameters **tracker_id** (*int*) – The ID of the tracker for which data is returned. The tracker-id value is found via the Get Devices endpoint.

add_alarms(*tracker_id*, *time*, *enabled*, *recurring*, *week_days*, ***kwargs*)
Add Alarm

Adds the alarm settings to a given ID for a given device.

Parameters

- **tracker_id** (*int*) – The ID of the tracker for which data is returned. The tracker-id value is found via the Get Devices endpoint.
- **time** (*string*) – Time of day that the alarm vibrates with a UTC timezone offset, e.g. 07:15-08:00.
- **enabled** (*bool*) – true or false. If false, alarm does not vibrate until enabled is set to true.
- **recurring** (*string*) – true or false. If false, the alarm is a single event.
- **week_days** (*string*) – Comma separated list of days of the week on which the alarm vibrates, e.g. MONDAY, TUESDAY.

update_alarms(*tracker_id*, *alarm_id*, *time*, *enabled*, *recurring*, *week_days*, *snooze_length*, *snooze_count*, ***kwargs*)
Update Alarm

Updates the alarm entry with a given ID for a given device. It also gets a response in the format requested.

Parameters

- **tracker_id** (*int*) – The ID of the tracker for which data is returned. The tracker-id value is found via the Get Devices endpoint.

- **alarm_id** (*int*) – The ID of the alarm to be updated. The alarm-id value is found in the response of the Get Activity endpoint.
- **time** (*string*) – Time of day that the alarm vibrates with a UTC timezone offset, e.g. 07:15-08:00.
- **enabled** (*bool*) – true or false. If false, the alarm does not vibrate until enabled is set to true.
- **recurring** (*string*) – true or false. If false, the alarm is a single event.
- **week_days** (*string*) – Comma seperated list of days of the week on which the alarm vibrates, e.g. MONDAY, TUESDAY.
- **snooze_length** (*int*) – Minutes between alarms.
- **snooze_count** (*int*) – Maximum snooze count.

delete_alarms (*tracker_id*, *alarm_id*, ***kwargs*)

Delete Alarm

Deletes the user's device alarm entry with the given ID for a given device.

Parameters

- **tracker_id** (*int*) – The ID of the tracker whose alarms is managed. The tracker-id value is found via the Get Devices endpoint.
- **alarm_id** (*int*) – The ID of the alarm to be updated. The alarm-id value is found via the Get Alarms endpoint.

get_foods_locales (***kwargs*)

Get Food Locales

Returns the food locales that the user may choose to search, log, and create food in.

get_foods_goal (***kwargs*)

Get Food Goals

Returns a user's current daily calorie consumption goal and/or foodPlan value in the format requested.

add_update_foods_goal (*calories*, *intensity=None*, *personalized=None*, ***kwargs*)

Update Food Goal

Updates a user's daily calories consumption goal or food plan and returns a response in the format requested.

Parameters

- **calories** (*int*) – Manual calorie consumption goal in either calories or intensity must be provided.
- **intensity** (*string, optional*) – Food plan intensity (MAINTENANCE, EASIER, MEDIUM, KINDAHARD, or HARDER). Either calories or intensity must be provided.
- **personalized** (*string, optional*) – Food plan type; true or false.

get_foods_by_date (*date*, ***kwargs*)

Get Food Logs

Retreives a summary and list of a user's food log entries for a given day in the format requested.

Parameters **date** (*datetime*) – A datetime object.

get_water_by_date(*date*, ***kwargs*)
Get Water Logs

Retreives a summary and list of a user's water log entries for a given day in the requested using units in the unit system that corresponds to the Accept-Language header provided.

Parameters **date** (*datetime*) – A datetime object.

get_water_goal(***kwargs*)
Get Water Goal

Retreives a summary and list of a user's water goal entries for a given day in the requested using units in the unit system that corresponds to the Accept-Language header provided.

add_update_water_goal(*target*, ***kwargs*)
Update Water Goal

Updates a user's daily calories consumption goal or food plan and returns a response in the format requested.

Parameters **target** (*int*) – The target water goal in the format X.X is set in unit based on locale.

get_foods_by_date_range(*resource_path*, *base_date*, *end_date*, ***kwargs*)
Get Food or Water Time Series

Updates a user's daily activity goals and returns a response using units in the unit system which corresponds to the Accept-Language header provided.

Parameters

- **resource_path** (*string*) – The resouce path. See options in the Resouce Path Options section in the full documentation. Possible values: ['caloriesIn', 'water']
- **base_date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.

get_foods_resource_by_date_period(*resource_path*, *date*, *period*, ***kwargs*)
Get Food or Water Time Series

Updates a user's daily activity goals and returns a response using units in the unit system which corresponds to the Accept-Language header provided.

Parameters

- **resource_path** (*string*) – The resouce path. See options in the Resouce Path Options section in the full documentation. Possible values: ['caloriesIn', 'water']
- **date** (*datetime*) – A datetime object.
- **period** (*string*) – The range for which data will be returned. Options are 1d, 7d, 30d, 1w, 3m, 6m, 1y, or max.

add_foods_log(*food_id*, *meal_type_id*, *unit_id*, *amount*, *date*, *food_name=None*, *favorite=None*, *brand_name=None*, *calories=None*, ***kwargs*)
Log Food

Creates food log entries for users with or without foodId value.

Parameters

- **food_id** (*string*) – The ID of the food to be logged. Either foodId or foodName must be provided.

- **meal_type_id** (*string*) – Meal types. 1=Breakfast; 2=Morning Snack; 3=Lunch; 4=Afternoon Snack; 5=Dinner; 7=Anytime.
- **unit_id** (*string*) – The ID of units used. Typically retrieved via a previous call to Get Food Logs, Search Foods, or Get Food Units.
- **amount** (*string*) – The amount consumed in the format X.XX in the specified unitId.
- **date** (*datetime*) – A datetime object.
- **food_name** (*string, optional*) – Food entry name. Either foodId or foodName must be provided.
- **favorite** (*bool, optional*) – The *true* value will add the food to the user's favorites after creating the log entry; while the *false* value will not. Valid only with foodId value.
- **brand_name** (*string, optional*) – Brand name of food. Valid only with foodName parameters.
- **calories** (*int, optional*) – Calories for this serving size. This is allowed with foodName parameter (default to zero); otherwise it is ignored.

delete_foods_log(*food_log_id*, ***kwargs*)

Delete Food Log

Deletes a user's food log entry with the given ID.

Parameters **food_log_id** (*string*) – The ID of the food log entry to be deleted.

add_water_log(*date*, *amount*, *unit=None*, ***kwargs*)

Log Water

Creates a log entry for water using units in the unit systems that corresponds to the Accept-Language header provided.

Parameters

- **date** (*datetime*) – A datetime object.
- **amount** (*int*) – The amount consumption in the format X.XX and in the specified waterUnit or in the unit system that corresponds to the Accept-Language header provided.
- **unit** (*string, optional*) – Water measurement unit; *ml*, *fl oz*, or *cup*.

delete_water_log(*water_log_id*, ***kwargs*)

Delete Water Log

Deletes a user's water log entry with the given ID.

Parameters **water_log_id** (*string*) – The ID of the waterUnit log entry to be deleted.

update_water_log(*water_log_id*, *amount*, *unit=None*, ***kwargs*)

Update Water Log

Updates a user's water log entry with the given ID.

Parameters

- **water_log_id** (*string*) – The ID of the waterUnit log entry to be deleted.
- **amount** (*string*) – Amount consumed; in the format X.X and in the specified waterUnit or in the unit system that corresponds to the Accept-Language header provided.
- **unit** (*string, optional*) – Water measurement unit. 'ml', 'fl oz', or 'cup'.

get_favorite_foods(***kwargs*)
Get Favorite Foods

Returns a list of a user's favorite foods in the format requested. A favorite food in the list provides a quick way to log the food via the Log Food endpoint.

get_frequent_foods(***kwargs*)
Get Frequent Foods

Returns a list of a user's frequent foods in the format requested. A frequent food in the list provides a quick way to log the food via the Log Food endpoint.

add_favorite_food(*food_id*, ***kwargs*)
Add Favorite Food

Updates a user's daily activity goals and returns a response using units in the unit system which corresponds to the Accept-Language header provided.

Parameters **food_id** (*string*) – The ID of the food to be added to user's favorites.

delete_favorite_food(*food_id*, ***kwargs*)
Delete Favorite Food

Deletes a food with the given ID to the user's list of favorite foods.

Parameters **food_id** (*string*) – The ID of the food to be deleted from user's favorites.

get_meals(***kwargs*)
Get Meals

Returns a list of meals created by user in the user's food log in the format requested. User creates and manages meals on the Food Log tab on the website.

add_meal(*name*, *description*, *food_id*, *unit_id*, *amount*, ***kwargs*)
Create Meal

Creates a meal with the given food contained in the post body.

Parameters

- **name** (*string*) – Name of the meal.
- **description** (*string*) – Short description of the meal.
- **food_id** (*string*) – ID of the food to be included in the meal.
- **unit_id** (*string*) – ID of units used. Typically retrieved via a previous call to Get Food Logs, Search Foods, or Get Food Units.
- **amount** (*string*) – Amount consumed; in the format X.XX, in the specified unitId.

update_meal(*meal_id*, *name*, *description*, *food_id*, *unit_id*, *amount*, ***kwargs*)
Edit Meal

Replaces an existing meal with the contents of the request. The response contains the updated meal.

Parameters

- **meal_id** (*string*) – Id of the meal to edit.
- **name** (*string*) – Name of the meal.
- **description** (*string*) – Short description of the meal.
- **food_id** (*string*) – ID of the food to be included in the meal.

- **unit_id** (*string*) – ID of units used. Typically retrieved via a previous call to Get Food Logs, Search Foods, or Get Food Units.
- **amount** (*string*) – Amount consumed; in the format X.XX, in the specified unitId.

delete_meal (*meal_id*, ***kwargs*)

Delete Meal

Deletes a user's meal with the given meal id.

Parameters **meal_id** (*string*) – Id of the meal to delete.

get_recent_foods (***kwargs*)

Get Recent Foods

Returns a list of a user's frequent foods in the format requested. A frequent food in the list provides a quick way to log the food via the Log Food endpoint.

add_foods (*name*, *default_food_measurement_unit_id*, *default_serving_size*, *calories*, *form_type=None*, *description=None*, ***kwargs*)

Create Food

Creates a new private food for a user and returns a response in the format requested. The created food is found via the Search Foods call.

Parameters

- **name** (*string*) – The food name.
- **default_food_measurement_unit_id** (*string*) – The ID of the default measurement unit. Full list of units can be retrieved via the Get Food Units endpoint.
- **default_serving_size** (*string*) – The size of the default serving. Nutrition values should be provided for this serving size.
- **calories** (*string*) – The calories in the default serving size.
- **form_type** (*string, optional*) – Form type; LIQUID or DRY.
- **description** (*string, optional*) – The description of the food.

delete_foods (*food_id*, ***kwargs*)

Delete Custom Food

Deletes custom food for a user and returns a response in the format requested.

Parameters **food_id** (*string*) – The ID of the food to be deleted.

get_foods_info (*food_id*, ***kwargs*)

Get Food

Returns the details of a specific food in the Fitbit food databases or a private food that an authorized user has entered in the format requested.

Parameters **food_id** (*string*) – The ID of the food.

get_foods_units (***kwargs*)

Get Food Units

Returns a list of all valid Fitbit food units in the format requested.

get_foods_list (*query*, ***kwargs*)

Search Foods

Returns a list of public foods from the Fitbit food database and private food the user created in the format requested.

Parameters **query** (*string*) – The URL-encoded search query.

get_friends (***kwargs*)
Get Friends

Returns data of a user's friends in the format requested using units in the unit system which corresponds to the Accept-Language header provided.

get_friends_leaderboard (***kwargs*)
Get Friends Leaderboard

Returns data of a user's friends in the format requested using units in the unit system which corresponds to the Accept-Language header provided.

get_friends_invitations (***kwargs*)
Get Friend Invitations

Returns a list of invitations to become friends with a user in the format requested.

create_friends_invitations (*invited_user_email=None*, *invited_user_id=None*, ***kwargs*)
Invite Friends

Creates an invitation to become friends with the authorized user. Either invitedUserEmail or invitedUserId needs to be provided.

Parameters

- **invited_user_email** (*string, optional*) – Email of the user to invite.
- **invited_user_id** (*string, optional*) – Encoded ID of the user to invite.

respond_friends_invitation (*from_user_id*, *accept*, ***kwargs*)
Respond to Friend Invitation

Accepts or rejects an invitation to become friends wit inviting user.

Parameters

- **from_user_id** (*string*) – The encoded ID of a user from which to accept or reject invitation.
- **accept** (*string*) – Accept or reject invitation; true or false.

get_heart_by_date_period (*date*, *period*, ***kwargs*)
Get Heart Rate Time Series

Returns the time series data in the specified range for a given resource in the format requested using units in the unit systems that corresponds to the Accept-Language header provided.

Parameters

- **date** (*datetime*) – A datetime object.
- **period** (*string*) – The range of which data will be returned. Options are 1d, 7d, 30d, 1w, and 1m.

get_heart_by_date_range (*base_date*, *end_date*, ***kwargs*)
Get Heart Rate Time Series

Returns the time series data in the specified range for a given resource in the format requested using units in the unit systems that corresponds to the Accept-Language header provided.

Parameters

- **base_date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.

get_heart_by_date_range_intraday(*date*, *end_date*, *detail_level*, ***kwargs*)

Get Heart Rate Intraday Time Series

Returns the intraday time series for a given resource in the format requested. If your application has the appropriate access, your calls to a time series endpoint for a specific day (by using start and end dates on the same day or a period of 1d), the response will include extended intraday values with a one-minute detail level for that day. Unlike other time series calls that allow fetching data of other users, intraday data is available only for and to the authorized user.

Parameters

- **date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.
- **detail_level** (*string*) – The number of data points to include either 1sec or 1min.

get_heart_by_date_range_timestamp_intraday(*date*, *end_date*, *detail_level*, *start_time*, *end_time*, ***kwargs*)

Get Heart Rate Intraday Time Series

Returns the intraday time series for a given resource in the format requested. If your application has the appropriate access, your calls to a time series endpoint for a specific day (by using start and end dates on the same day or a period of 1d), the response will include extended intraday values with a one-minute detail level for that day. Unlike other time series calls that allow fetching data of other users, intraday data is available only for and to the authorized user.

Parameters

- **date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.
- **detail_level** (*string*) – The number of data points to include either 1sec or 1min.
- **start_time** (*string*) – The start of the period in the format of HH:mm.
- **end_time** (*string*) – The end time of the period in the format of HH:mm.

get_heart_by_date_intraday(*date*, *detail_level*, ***kwargs*)

Get Heart Rate Intraday Time Series

Returns the intraday time series for a given resource in the format requested. If your application has the appropriate access, your calls to a time series endpoint for a specific day (by using start and end dates on the same day or a period of 1d), the response will include extended intraday values with a one-minute detail level for that day. Unlike other time series calls that allow fetching data of other users, intraday data is available only for and to the authorized user.

Parameters

- **date** (*datetime*) – A datetime object.
- **detail_level** (*string*) – The number of data points to include either 1sec or 1min.

get_heart_by_date_timestamp_intraday(*date*, *detail_level*, *start_time*, *end_time*, ***kwargs*)

Get Heart Rate Intraday Time Series

Returns the intraday time series for a given resource in the format requested. If your application has the appropriate access, your calls to a time series endpoint for a specific day (by using start and end dates on the same day or a period of 1d), the response will include extended intraday values with a one-minute detail level for that day. Unlike other time series calls that allow fetching data of other users, intraday data is available only for and to the authorized user.

Parameters

- **date** (*datetime*) – A datetime object.
- **detail_level** (*string*) – The number of data points to include either 1sec or 1min.
- **start_time** (*string*) – The start of the period in the format of HH:mm.
- **end_time** (*string*) – The end time of the period in the format of HH:mm.

delete_sleep (*log_id*, ***kwargs*)
Delete Sleep Log

Deletes a user's sleep log entry with the given ID.

Parameters **log_id** (*string*) – The ID of the sleep log to be deleted.

get_sleep_by_date (*date*, ***kwargs*)
Get Sleep Log

The Get Sleep Logs by Date endpoint returns a summary and list of a user's sleep log entries (including naps) as well as detailed sleep entry data for a given day.

Parameters **date** (*datetime*) – A datetime object.

get_sleep_by_date_range (*base_date*, *end_date*, ***kwargs*)
Get Sleep Logs by Date Range

The Get Sleep Logs by Date Range endpoint returns a list of a user's sleep log entries (including naps) as well as detailed sleep entry data for a given date range (inclusive of start and end dates).

Parameters

- **base_date** (*datetime*) – A datetime object.
- **end_date** (*datetime*) – A datetime object.

get_sleep_list (*sort*, *offset*, *limit*, *before_date=None*, *after_date=None*, ***kwargs*)
Get Sleep Logs List

The Get Sleep Logs List endpoint returns a list of a user's sleep logs (including naps) before or after a given day with offset, limit, and sort order.

Parameters

- **sort** (*string*) – The sort order of entries by date asc (ascending) or desc (descending).
- **offset** (*int*) – The offset number of entries.
- **limit** (*int*) – The maximum number of entries returned (maximum;100).
- **before_date** (*datetime, optional*) – A datetime object.
- **after_date** (*datetime, optional*) – A datetime object.

get_sleep_goal (***kwargs*)
Get Sleep Goal

Returns the user's sleep goal.

update_sleep_goal (*min_duration*, ***kwargs*)
Update Sleep Goal

Create or update the user's sleep goal and get a response in the JSON format.

Parameters **min_duration** (*string*) – Duration of sleep goal.

add_sleep (*start_time*, *duration*, *date*, ***kwargs*)
Log Sleep

Creates a log entry for a sleep event and returns a response in the format requested.

Parameters

- **start_time** (*string*) – Start time includes hours and minutes in the format HH:mm.
- **duration** (*int*) – Duration in milliseconds.
- **date** (*datetime*) – A datetime object.

get_subscriptions_list (*collection_path*, ***kwargs*)

Get a List of Subscriptions

Retreives a list of a user's subscriptions for your application in the format requested. You can either fetch subscriptions for a specific collection or the entire list of subscriptions for the user. For best practice, make sure that your application maintains this list on your side and use this endpoint only to periodically ensure data consistency.

Parameters **collection_path** (*string*) – This is the resource of the collection to receive notifications from (foods, activities, sleep, or body). If not present, subscription will be created for all collections. If you have both all and specific collection subscriptions, you will get duplicate notifications on that collections' updates. Each subscriber can have only one subscription for a specific user's collection.

add_subscriptions (*collection_path*, *subscription_id*, ***kwargs*)

Add a Subscription

Adds a subscription in your application so that users can get notifications and return a response in the format requested. The subscription-id value provides a way to associate an update with a particular user stream in your application.

Parameters

- **collection_path** (*string*) – This is the resource of the collection to receive notifications from (foods, activities, sleep, or body). If not present, subscription will be created for all collections. If you have both all and specific collection subscriptions, you will get duplicate notifications on that collections' updates. Each subscriber can have only one subscription for a specific user's collection.
- **subscription_id** (*string*) – This is the unique ID of the subscription created by the API client application. Each ID must be unique across the entire set of subscribers and collections. The Fitbit servers will pass this ID back along with any notifications about the user indicated by the user parameter in the URL path.

delete_subscriptions (*collection_path*, *subscription_id*, ***kwargs*)

Delete a Subscription

Deletes a subscription for a user..

Parameters

- **collection_path** (*string*) – This is the resource of the collection to receive notifications from (foods, activities, sleep, or body). If not present, subscription will be created for all collections. If you have both all and specific collection subscriptions, you will get duplicate notifications on that collections' updates. Each subscriber can have only one subscription for a specific user's collection.
- **subscription_id** (*string*) – This is the resource of the collection to receive notifications from (foods, activities, sleep, or body). If not present, subscription will be created for all collections. If you have both all and specific collection subscriptions, you will get duplicate notifications on that collections' updates. Each subscriber can have only one subscription for a specific user's collection.

get_badges(***kwargs*)

Get Badges

Retrieves the user's badges in the format requested. Response includes all badges for the user as seen on the Fitbit website badge locker (both activity and weight related.) The endpoint returns weight and distance badges based on the user's unit profile preference as on the website.

get_profile(***kwargs*)

Get Profile

Returns a user's profile. The authenticated owner receives all values. However, the authenticated user's access to other users' data is subject to those users' privacy settings. Numerical values are returned in the unit system specified in the Accept-Language header.

update_profile(*gender=None*, *birthday=None*, *height=None*, *about_me=None*, *fullname=None*, *country=None*, *state=None*, *city=None*, *stride_length_walking=None*, *stride_length_running=None*, *weight_unit=None*, *height_unit=None*, *water_unit=None*, *glucose_unit=None*, *timezone=None*, *foods_locale=None*, *locale=None*, *locale_lang=None*, *locale_country=None*, *start_day_of_week=None*, ***kwargs*)

Update Profile

Updates a user's profile using a form.

Parameters

- **gender** (*string, optional*) – The sex of the user; (MALE/FEMALE/NA).
- **birthday** (*datetime, optional*) – A datetime object.
- **height** (*string, optional*) – The height in the format X.XX in the unit system that corresponds to the Accept-Language header provided.
- **about_me** (*string, optional*) – This is a short description of user.
- **fullname** (*string, optional*) – The fullname of the user.
- **country** (*string, optional*) – The country of the user's residence. This is a two-character code.
- **state** (*string, optional*) – The US state of the user's residence. This is a two-character code if the country was or is set to US.
- **city** (*string, optional*) – The US city of the user's residence.
- **stride_length_walking** (*string, optional*) – Walking stride length in the format X.XX, where it is in the unit system that corresponds to the Accept-Language header provided.
- **stride_length_running** (*string, optional*) – Running stride length in the format X.XX, where it is in the unit system that corresponds to the Accept-Language header provided.
- **weight_unit** (*string, optional*) – Default weight unit on website (which doesn't affect API); one of (en_US, en_GB, 'any' for METRIC).
- **height_unit** (*string, optional*) – Default height/distance unit on website (which doesn't affect API); one of (en_US, en_GB, 'any' for METRIC).
- **water_unit** (*string, optional*) – Default water unit on website (which doesn't affect API); one of (en_US, en_GB, 'any' for METRIC).
- **glucose_unit** (*string, optional*) – Default glucose unit on website (which doesn't affect API); one of (en_US, en_GB, 'any' for METRIC).

- **timezone** (*string, optional*) – The timezone in the format 'America/Los_Angeles'.
- **foods_locale** (*string, optional*) – The food database locale in the format of xx.XX.
- **locale** (*string, optional*) – The locale of the website (country/language); one of the locales, currently – (en_US, fr_FR, de_DE, es_ES, en_GB, en_AU, en_NZ, ja_JP).
- **locale_lang** (*string, optional*) – The Language in the format 'xx'. You should specify either locale or both - localeLang and localeCountry (locale is higher priority).
- **locale_country** (*string, optional*) – The Country in the format 'xx'. You should specify either locale or both - localeLang and localeCountry (locale is higher priority).
- **start_day_of_week** (*string, optional*) – The Start day of the week, meaning what day the week should start on. Either Sunday or Monday.

PYTHON MODULE INDEX

f

INDEX

I

O

R

U