Python CrowdFlower API Documentation

Release 0.0

Ilja Everilä

April 05, 2016

Contents

CHAPTER 1

CrowdFlower

"CrowdFlower offers scalable solutions that deliver fast and accurate results for business data problems. Find one that's best for you, and harness the power of the world's largest workforce." [1]

"The CrowdFlower API gives developers the ability to build applications that interact with and use all the features of CrowdFlower in an automated fashion. Tasks can be generated, work can be ordered, and your application can be notified as data is processed and judged by the CrowdFlower platform. The methods and practices described in this documentation are subject to change as the CrowdFlower API matures." [2]

1.1 Python API

This python implementation is an unofficial project not related, endorsed or supported by CrowdFlower, Inc.

1.2 Installation

Install the latest version from GitHub:

```
git clone https://github.com/everilae/crowdflower.git
cd crowdflower
python setup.py develop
```

1.3 Examples

```
>>> import crowdflower.client
>>> client = crowdflower.client.Client('yourapikey')
>>> job = client.get_job(123123)
>>> job.title = 'New Title'
>>> job.instructions = """
... <h1>Better instructions</h1>
... <p>You should read them</p>
... """
>>> job.cml = """
... <cml:text label="Sample text field:" default="Enter text here" validates="required"/>
... """
```

[1] http://crowdflower.com/overview (Fri Jan 17 11:27:23 UTC 2014)
[2] http://success.crowdflower.com/customer/portal/articles/1288323-api-documentation (Wed Sep 3 14:27:43 UTC 2014)

```
>>> # Send changes to server
>>> job.update()
```

1.4 Status

Early alpha, things are going to change a lot still.

1.5 Contributors

Ilja Everilä <ilja.everila@liilak.com>

Contents

2.1 crowdflower.client

exception `crowdflower.client.`**`ApiError`**
API error class, wraps HTTP exceptions and such.

class `crowdflower.client.`**`Client`**(*key*)
CrowdFlower API client. Requires API `key` for authentication.

TODO: Trust data model types in order to provide general methods instead of specialized do_this and do_that methods.

Parameters **`key`** – CrowdFlower API key. Required for authentication.

`add_job_tag`(*job_id*, *tag*)
Add tag to job.

`call`(*path*, *data=None*, *headers={}*, *query={}*, *method='get'*, *files=None*, *as_json=True*)
Data may be str (unicode) or bytes. Unicode strings will be encoded to UTF-8 bytes.

Parameters

- **`data`** (*`str, bytes or dict`*) – Byte data for POST
- **`headers`** (*`dict`*) – Additional headers
- **`query`** (*`dict`*) – Additional query parameters
- **`method`** (*`str`*) – GET, POST, PUT or DELETE
- **`files`** (*`dict`*) – Files to upload
- **`as_json`** (*`bool`*) – Handle response as json, defaults to True

Returns JSON dictionary

Return type dict

`cancel_unit`(*job_id*, *unit_id*)
Cancel unit.

`convert_job_test_questions`(*job_id*)
Convert uploaded gold to test questions.

`copy_job`(*job_id*, *all_units*, *gold*)
Copy Job `job_id` to a new job.

Parameters

- **`all_units`** – If true, all of this job's units will be copied to the new job.
- **`gold`** – If true, only golden units will be copied to the new job.

Returns crowdflower.job.Job

`create_job`(*attrs*)

Create new job with `attrs`, where attributes of most interest are:

- title
- instructions
- cml
- js
- css

Other R/W attributes:

- auto_order
- auto_order_threshold
- auto_order_timeout
- fields
- confidence_fields
- custom_key
- excluded_countries
- gold_per_assignment
- included_countries
- judgments_per_unit
- language
- max_judgments_per_unit
- max_judgments_per_contributor
- min_unit_confidence
- options
- pages_per_assignment
- problem
- send_judgments_webhook
- state
- units_per_assignment
- webhook_uri

Parameters **`attrs`** (`dict`) – JSON dictionary of attributes for new job

Returns Newly created Job

Return type *crowdflower.job.Job*

debit_order (*job*, *units_count*, *channels*)
Create a debit `order` for `Job` with `units_count` at `channels`.

delete_job (*job_id*)
Delete job `job_id` from CrowdFlower.

get_job (*job_id*)
Get Job `job_id`

Parameters **job_id** (`int`) – Id of crowdflower job to get

Returns Crowdflower job

Return type *crowdflower.job.Job*

get_job_channels (*job_id*)
Get available and enabled channels for `job_id`.

```
# A response JSON dictionary
{
    "enabled_channels": [
        "amt"
    ],
    "available_channels": [
        "amt",
        "sama",
        "gambit",
        "mob",
        "iphone"
    ]
}
```

get_job_tags (*job_id*)
Get tags for `job`.

get_jobs ()
Get Jobs connected to this client and key.

Returns an iterator of CrowdFlower jobs

Return type iter of crowdflower.job.Job

get_judgment (*job*, *judgment_id*)
Get Judgment `judgment_id` for `job`.

get_judgmentaggregates (*job*)
Get JudgmentAggregates for `job`.

Note: Return value from judgments.json seems to be a dictionary, where the keys are Unit ids and values an aggregate of a sort. The aggregate lacks documentation at https://crowdflower.com/docs-api , so this code is very very likely to break in the future.

get_order (*job*, *order_id*)
Get `Order` by `order_id` for `Job` `job`.

get_report (*job*, *type_='json'*)
Download and uncompress reports. Returns a list of `Units`.

get_unit (*job*, *unit_id*)
Get `Unit` `unit_id` for `Job`.

get_units (*job*)
Get `unit promises` for `Job`.

paged_call (**args*, ***kwgs*)
Generate paged calls to API end points, wraps `_call()`. Provide `sentinel` in order to stop paging at desired point. If `sentinel` is a function, it should accept latest `response` as argument.

This can not yield items from response, since some responses are dictionaries, while others are lists.

Parameters

- **page** – Page to start at, defaults to 1.
- **limit** – Limit pages to `limit` items, defaults to 100.
- **sentinel** – Sentinel value for `iter()`.

set_job_channels (*job_id*, *channels*)
Enable `channels` for `job_id`.

Parameters

- **job_id** – Id of job to set channels for
- **channels** – a list of channels to enable

set_job_tags (*job_id*, *tags*)
Set tags for job.

unit_from_json (*data*)
Create a new Unit instance from JSON `data`.

update_job (*job_id*, *attrs*)
Update Job `job_id` with `attrs`

Parameters

- **job_id** (`int`) – Id of crowdflower job to update
- **attrs** (`dict`) – JSON dictionary of attributes to update

upload_job (*data*, *job_id=None*, *force=False*)
Upload given data as JSON.

Parameters

- **data** (`collections.abc.Iterable`) – Iterable of JSON serializable objects
- **job_id** (`int`) – Id of a crowdflower job to update (optional)
- **force** (`bool`) – If True force adding units even if the columns do not match existing data

Returns crowdflower.job.Job instance

Return type *crowdflower.job.Job*

upload_job_file (*file*, *type_=None*, *job_id=None*, *force=False*)
Upload a file like object or open a file for reading and upload.

Caller is responsible for handling context on file like objects. Type must be provided with data as there is no information to make a guess from. If file like object provides text (unicode) data, it will be encoded to UTF-8 bytes.

If explicit `type_` is not provided and the `file` is a string containing a filename to open, will make a guess with mimetypes. Returns a new Job instance related to the uploaded data.

If type information is not given and guessing did not work, will raise a ValueError.

Parameters

- **`file`** (*`str or file`*) – A file like object or a filename string, contains UTF-8 encoded data
- **`type`** (*`str`*) – Explicit type, required for file like objects
- **`job_id`** (*`int`*) – Id of a crowdflower job to update (optional)
- **`force`** (*`bool`*) – If True force adding units even if the columns do not match existing data

Returns crowdflower.job.Job instance

Return type *crowdflower.job.Job*

class `crowdflower.client.`**`PathFactory`**(*client*, *name=()*)
Magic attribute/item syntax for making calls.

2.2 crowdflower.job

class `crowdflower.job.`**`Job`**(*client=None*, ***data*)
CrowdFlower Job.

Documentation for attributes can be found at https://success.crowdflower.com/hc/en-us/articles/202703435-CrowdFlower-API-Jobs-Resource-Attributes.

Parameters

- **`data`** (*`dict`*) – Job JSON dictionary
- **`client`** (`crowdflower.client.Client`) – *`Client`* instance that created this job instance

`add_tag`(*tag*)
Add tag.

`cancel`()
Permanently `cancel` a Job, stopping any incoming judgments and refunding your account for unreceived judgments.

`channels`
List of enabled channels for this job.

`convert_test_questions`()
Convert uploaded golden units to test questions.

`copy`(*all_units=False*, *gold=False*)
Create a new job that is a copy of this job.

Parameters

- **`all_units`** – If true, all of this job's units will be copied to the new job.
- **`gold`** – If true, only golden units will be copied to the new job.

Returns crowdflower.job.Job

`delete`()
Delete this job, removing it from CrowdFlower. Calling *`Job`* instance will be invalid after deletion and must not be used anymore.

`get_judgment`(*judgment_id*)
Get single *`Judgment`* for this *`Job`*.

get_results_report()

Download and parse JSON report containing aggregates and individual judgments as a list of `Units`. :returns: list of crowdflower.unit.Unit

get_worker(*worker_id*)

Get *Worker* `worker_id` bound to this *Job*.

judgment_aggregates

List of *JudgmentAggregate* instances of this *Job*.

> **Warning:** Judgments are paged with a maximum of 100 items per page. If your job has a lot of judgments – thousands or more – this will take a very VERY long time to finish when accessed for the first time. This might produce some nasty surprises, if *Job* instances are inspected with `inspect.getmembers()` or some such.

launch(*units_count*, *channels=('on_demand'*, *)*)

Order job with `units_count` at `channels`.

legend()

Display generated keys submitted with the form.

pause()

Temporarily stop judgments from coming in. A paused Job may `resume`.

ping()

Check the status/progress of Job.

resume()

Resume a Job from `pause` at any time.

tags

List of tags.

units

List of *UnitPromise* instances of this *Job*.

update()

Send updates made to this instance to CrowdFlower. Note that `title`, `instructions` and `cml` attributes must exist (either in the update or in the job already) for any changes to really persist, and so this method raises a `RuntimeError`, if any of them is missing.

> "At minimum, your job must have a valid title, instructions, and one *required* CML form element to be saved successfully." [1]

> **Warning:** The API will happily return a "valid" response when sent only the 'instructions', but nothing will change on the server side without all three. The caller is responsible for providing valid CML.

Raises RuntimeError – if `title`, `instructions` or `cml` is missing

upload(*data*, *force=False*)

Upload given data as JSON.

Parameters

- **data** (*collections.abc.Iterable*) – Iterable of JSON serializable objects
- **force** (*bool*) – If True force adding units even if the columns do not match existing data

[1] https://success.crowdflower.com/hc/en-us/articles/202703435-CrowdFlower-API-Jobs-Resource-Attributes (Tue Jun 30 07:31:00 UTC 2015)

upload_file (*file*, *type_=None*, *force=False*)

Upload a file like object or open a file for reading and upload.

Caller is responsible for handling context on file like objects. Type must be provided with data as there is no information to make a guess from. If file like object provides text (unicode) data, it will be encoded to UTF-8 bytes.

If explicit `type_` is not provided and the `file` is a string containing a filename to open, will make a guess with mimetypes.

If type information is not given and guessing did not work, will raise a `ValueError`.

Valid types are `text/csv` and `application/json` for `.csv` and `.json` respectively.

Parameters

- **file** (*str or file*) – A file like object or a filename string, contains UTF-8 encoded data
- **type** (*str*) – Explicit type, required for file like objects
- **force** (*bool*) – If True force adding units even if the columns do not match existing data

Raises **ValueError** – if type information isn't provided and cannot guess

2.3 crowdflower.judgment

class `crowdflower.judgment.`**Judgment** (*job*, *client=None*, ***data*)

CrowdFlower Judgment.

Parameters

- **job** (`crowdflower.job.Job`) – *Job* instance that this *Judgment* belongs to
- **client** (`crowdflower.client.Client`) – *Client* instance
- **data** (*dict*) – Job JSON dictionary

class `crowdflower.judgment.`**JudgmentAggregate** (*job*, *client=None*, ***data*)

CrowdFlower Judgment aggregate.

Parameters

- **job** (`crowdflower.job.Job`) – *Job* instance that this *JudgmentAggregate* belongs to
- **client** (`crowdflower.client.Client`) – *Client* instance
- **data** (*dict*) – Job JSON dictionary

get_aggregate (*field*)

Aggregated result for `field`, chosen by CrowdFlower's aggregation logic.

get_fields ()

Get full aggregated field value dictionaries as a dictionary.

Returns dictionary of field, value items

Return type dict

get_results (*field*)

Full results for field `field`

Parameters **field** – Field name

Returns Results for field

judgments
List of *Judgment* instances for this aggregate.

2.4 crowdflower.unit

class `crowdflower.unit.`**`Unit`**(*job*, *client=None*, ***data*)
CrowdFlower Unit.

Documentation for attributes can be found at http://success.crowdflower.com/customer/portal/articles/1621707
.

Parameters

- **job** (`crowdflower.job.Job`) – *Job* instance owning this *Unit*
- **client** (`crowdflower.client.Client`) – *Client* instance
- **data** (*dict*) – Unit JSON dictionary

cancel()
Cancel unit.

get_aggregate(*key*, *default=None*)
Get aggregated result for `key`, or return `default`.

Parameters

- **key** (*str*) – Name of result value.
- **default** – Default value in case the given key is not found.

results
Get unit results, if available. RO attribute.

class `crowdflower.unit.`**`UnitPromise`**(*job*, *client=None*, ***data*)
A promise that a *Unit* will be available for querying attributes when needed.

2.5 crowdflower.worker

class `crowdflower.worker.`**`Worker`**(*job*, *client=None*, ***data*)
CrowdFlower Worker.

Parameters

- **job** (`crowdflower.job.Job`) – *Job* instance owning this *Worker*
- **client** (`crowdflower.client.Client`) – *Client* instance
- **data** (*dict*) – Attributes

bonus(*amount*, *reason=None*)
Pay *Worker* a bonus of `amount` cents. Optionally include a message stating the `reason` of the bonus.

Parameters

- **amount** (*int*) – Amount in cents
- **reason** (*str*) – Include a message with the bonus

deflag (*deflag*)
De-flags a *Worker* with the reason `deflag`.

Parameters **deflag** (*str*) – De-flag reason

flag (*flag*, *persist=False*)
Flags and prevents a *Worker* from completing the *Job* with the reason `flag`. Existing *judgments* will not be thrown away. If `persist` is se to `True`, then the Worker is flagged out from all Jobs.

Parameters

- **flag** (*str*) – Flag reason
- **persist** (*bool*) – If True, flag in all Jobs (default False)

notify (*message*)
Notify a *Worker* contributor with the `message`. The message appears in the workers dashboard.

Parameters **message** (*str*) – Message to Worker

reject ()
Prevents *Worker* from completing *jobs* and removes all *judgments*.

Care should be taken since a finalized *Job* cannot collect new *judgments* to replace the missing data.

This feature is only available to Pro and Enterprise users.

2.6 crowdflower.order

class `crowdflower.order.`**Order** (*job*, *client=None*, ***data*)
CrowdFlower Order.

Documentation for attributes can be found at http://success.crowdflower.com/customer/portal/articles/1288323-api-documentation#header_5

An Order must be placed for a *Job* to collect *Judgment*.

Parameters

- **job** (`crowdflower.job.Job`) – *Job* instance owning this `Unit`
- **client** (`crowdflower.client.Client`) – *Client* instance
- **data** (*dict*) – Unit JSON dictionary

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index