temba Documentation

Nyaruka

Mar 20, 2019

Contents

1 Fetching Objects **3**

2 Rate Limiting **5**

3 Error Handling **7**

4 See Also **9**

4.1 temba_client.v2 package 9

4.1.1 Module contents 9

This is the official Python client library for RapidPro. The current stable API version is 2.

To create a `temba_client.v2.TembaClient` instance, you need to know the name of the host server, and your API token. If you don't know your API token then visit the API Explorer. For example:

```
from temba_client.v2 import TembaClient
client = TembaClient('rapidpro.io', <YOUR-API-TOKEN>)
```

Alternatively you can create a client from the complete URL of the API root. You can use this if you need to connect to a local RapidPro instance that doesn't use SSL. For example:

```
client = TembaClient('http://localhost:8000/api/v2', <YOUR-API-TOKEN>)
```

CHAPTER 1

Fetching Objects

For each type, the client provides a *get_* method for fetching a set of matching objects. If you want a single object then query using the unique identifier for that type. For most types this will be a UUID. However for broadcasts, messages and flow runs, this will be an integer id, and for fields this will be the key name.

You can then use the *.first()* to resolve that to a single object, or *None* if no such object exists. For example:

```
contact = client.get_contacts(uuid='cc276708-7752-4b52-a4ea-d3264847220e').first()
message = client.get_messages(id=12345).first()
field = client.get_fields(key='age').first()
```

You can also use *.all()* to fetch all of the matching objects. For example:

```
contacts = client.get_contacts(group='Reporters').all()
messages = client.get_messages(folder='Inbox').all()
fields = client.get_fields().all()
```

However such calls may require multiple API requests behind the scenes and take a long time to return depending on how many objects match the query. For this reason *.all()* should be used with caution. If you know that the number of objects is going to be high, it is better to use *.iterfetches()* to iterate over each individual request to the API. For example:

```
for contact_batch in client.get_contacts(group='Reporters').iterfetches():
  for contact in contact_batch:
    # do something with this contact...
```

CHAPTER 2

Rate Limiting

API v2 introduces rate limits which are currently 2500 requests per hour but may change in the future. If you exceed the number of requests for your organization, then the next client call you make will throw a *TembaRateExceededError* exception. The exception will have a *retry_after* attribute which is the number of seconds you should wait for before making another call.

For convenience the client can handle rate errors by sleeping and retrying after the specified period. For example:

```
contacts = client.get_contacts(group='Reporters').all(retry_on_rate_exceed=True):
```

CHAPTER 3

Error Handling

If an API request causes a validation error in RapidPro, the client call will raise a *TembaBadRequestError* which will contain the error messages for each field.

```
try:
    client.update_label('invalid-uuid', name="Test", parent=None)
except TembaBadRequestError as ex:
    for field, field_errors = ex.errors.iteritems():
        for field_error in field_errors:
            # TODO do something with each error message
```

Other exceptions which may be thrown are:

- *TembaConnectionError* if there was a problem connecting to the RapidPro instance.
- *TembaTokenError* if the API token used is invalid.
- *TembaNoSuchObjectError* if a request is made to update or delete a non-existent object.
- *TembaRateExceededError* if you have exceeded the allowed requests in a given time window.
- *TembaHttpError* if any other type of error code is returned

CHAPTER 4

See Also

4.1 temba_client.v2 package

This package contains the `temba_client.v2.TembaClient` client class. This is client for the v2 API which is current stable API version recommended for most users.

4.1.1 Module contents

- genindex
- search