FCM Client Documentation

Release 0.2.2

Sardar Yumatov

Jun 13, 2018

Contents

Python client for Firebase Cloud Messaging (FCM).

The GCM library was originally written by Sardar Yumatov. It seems to have been abandoned around 2015 or 2016. When google announced the move to Firebase, there was a need for a updated version of this software.

CHAPTER 1

Requirements

- requests - HTTP request, handles proxies etc.

CHAPTER 2

Alternatives

Th only alternative library known at the time of writing was pyfcm. This library differs in the following design decisions:

- *Predictable execution time*. Do not automatically retry request on failure. According to Google's recommendations, each retry has to wait exponential back-off delay. We use a job queue backend like celery, where the best way to retry after some delay will be scheduling the task with `countdown=delay`. Sleeping while in Celery worker hurts your concurrency.
- *Do not forget results if you need to retry*. This sounds obvious, but `python-fcm` drops important results, such as canonical ID mapping if request needs to be (partially) retried.
- *Clean pythonic API*. No need to borrow all Java like exceptions etc.
- *Do not hard-code validation, let FCM fail*. This decision makes library a little bit more future proof.

CHAPTER 3

Support

FCM client is maintained by John Loehrer, contact me if you find any bugs or need help. You can view outstanding issues on the FCM Github page.

Contents:

3.1 Getting Started

Follow the instructions here for how to set up firebase cloud messaging.

3.1.1 Usage

Usage is straightforward:

```
from fcmclient import *

# Pass 'proxies' keyword argument, as described in 'requests' library if you
# use proxies. Check other options too.
fcm = FCM(API_KEY)

# Construct (key => scalar) payload. do not use nested structures.
data = {'str': 'string', 'int': 10}

multicast = JSONMessage(["registration_id_1", "registration_id_2"], data, collapse_
↪key='my.key', dry_run=True)

try:
    # attempt send
    res = fcm.send(multicast)

    # nothing to do on success
    for reg_id, msg_id in res.success.items():
```

(continues on next page)

(continued from previous page)

```
        print "Successfully sent %s as %s" % (reg_id, msg_id)

    # update your registration ID's
    for reg_id, new_reg_id in res.canonical.items():
        print "Replacing %s with %s in database" % (reg_id, new_reg_id)

    # probably app was uninstalled
    for reg_id in res.not_registered:
        print "Removing %s from database" % reg_id

    # unrecoverably failed, these ID's will not be retried
    # consult FCM manual for all error codes
    for reg_id, err_code in res.failed.items():
        print "Removing %s because %s" % (reg_id, err_code)

    # if some registration ID's have recoverably failed
    if res.needs_retry():
        # construct new message with only failed regids
        retry_msg = res.retry()
        # you have to wait before attemting again. delay()
        # will tell you how long to wait depending on your
        # current retry counter, starting from 0.
        print "Wait or schedule task after %s seconds" % res.delay(retry)
        # retry += 1 and send retry_msg again

except FCMAuthenticationError:
    # stop and fix your settings
    print "Your Google API key is rejected"
except ValueError, e:
    # probably your extra options, such as time_to_live,
    # are invalid. Read error message for more info.
    print "Invalid message/option or invalid FCM response"
    print e.args[0]
except Exception:
    # your network is down or maybe proxy settings
    # are broken. when problem is resolved, you can
    # retry the whole message.
    print "Something wrong with requests library"
```

3.2 fcmclient Package

Firebase Cloud Messaging client built on top of requests library.

3.2.1 `fcmclient` Package

`fcmclient.api.`**`FCM_URL = 'https://fcm.googleapis.com/fcm/send'`**

Default URL to FCM service.

`class` `fcmclient.api.`**`FCM`**(*api_key*, *url='https://fcm.googleapis.com/fcm/send'*, *backoff=1000*, ***options*)

Create new connection.

Parameters

- **`api_key`** – (str) Google API key

- **url** – (str) FCM server URL.
- **backoff** – (int) initial backoff in milliseconds.
- **options** – (kwargs) options for ‘requests

send(*message*)

Send message.

The message may contain various options, such as `time_to_live`. Your request might be rejected, because some of your options might be invalid. In this case a `ValueError` with explanation will be raised.

Arguments *message* (`Message`): plain text or JSON message.

Returns *Result* interpreting the results.

Raises

- `requests.exceptions.RequestException` on any network problem.
- `ValueError` if your FCM request or response is rejected.
- *FCMAuthenticationError* your API key is invalid.

class fcmclient.api.**JSONMessage**(*registration_ids*, *data=None*, *message_title=None*, *message_body=None*, *payload=None*, ***options*)

Multicast message, uses JSON format.

Arguments

- ***registration_ids* (list): registration ID’s of target** devices.
- *data* (dict): key-value pairs, payload of this message.
- *message_title* (str): a title for the notification
- *message_body* (str): the message body of the alert.
- *options* (dict): FCM options.

Refer to FCM for more explanation on available options.

Options

- *collapse_key* (str): collapse key/bucket.
- *time_to_live* (int): message TTL in seconds.
- ***delay_while_idle* (bool): hold message if** device is off-line.
- *restricted_package_name* (str): declare package name.
- *dry_run* (bool): pretend sending message to devices.

__getstate__()

Returns `dict` with `__init__` arguments.

If you use `pickle`, then simply pickle/unpickle the message object. If you use something else, like JSON, then:

```
# obtain state dict from message
state = message.__getstate__()
# send/store the state
# recover state and restore message. you have to pick the
right class
message_copy = JSONMessage(**state)
```

Returns *kwargs* for *JSONMessage* constructor.

registration_ids
Target registration ID's.

class fcmclient.api.**Result**(*message*, *response*, *backoff*)
Result of send operation. You should check *canonical()* for any registration ID's that should be updated. If the whole message or some registration ID's have recoverably failed, then *retry()* will provide you with new message. You have to wait *delay()* seconds before attempting a new request.

backoff(*retry=0*)
Computes exponential backoff for given retry number.

canonical
New registration ID's as mapping `{registration_id: canonical_id}`.

You have to update registration ID's of your subscribers by replacing them with corresponding canonical ID. Read more here.

delay(*retry=0*)
Time to wait in seconds before attempting a retry as a float number.

This method will return value of Retry-After header if it is provided by FCM. Otherwise, it will return (backoff * 2^retry) with some random shift. Google may black list your server if you do not honor Retry-After hint and do not use exponential backoff.

failed
Unrecoverably failed regisration ID's as mapping `{ registration_id: error code}`.

This method lists devices, that have failed with something else than:

- `Unavailable` – look for *retry()* instead.
- `NotRegistered` – look for *not_registered* instead.

Read more about possible error codes.

needs_retry()
True if *retry()* will return message.

not_registered
List all registration ID's that FCM reports as `NotRegistered`. You should remove them from your database.

retry()
Construct new message that will unicast/multicast to remaining recoverably failed registration ID's. Method returns None if there is nothing to retry. Do not forget to wait for *delay()* seconds before new attempt.

success
Successfully processed registration ID's as mapping `{ registration_id: message_id}`.

class fcmclient.api.**FCMAuthenticationError**
Raised if your Google API key is rejected.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index

Symbols

B

C

D

F

J

N

R

S