pushka Documentation

Release 0.1.0

Alexey Kinëv

March 02, 2016

Contents

1 Install **3**

2 Quickstart **5**

3 Contents **7**
- 3.1 Sending emails . . . 7
- 3.2 Sending SMS . . . 7
- 3.3 Core API . . . 7

4 Indices and tables **11**

Python Module Index **13**

Push notifications, SMS and emails on top of asyncio.

Supported services:

- Email: Amazon SES
- SMS: Twilio
- Push notifications: Parse (deprecated)

CHAPTER 1

Install

The easiest way is install via pip:

```
pip install pushka
```

CHAPTER 2

Quickstart

If you are new to asyncio, please read some intro on it first! And here's a basic example for sending email message via Amazon SES:

```
import asyncio
import pushka

# NOTE: client uses `SendEmail` method, so user's policy must grant
# `ses:SendEmail` permission.
access_id='***' # Amazon SES access ID
secret_key='***' # Amazon SES secret key

mail_to = ['to@example.com'] # Some email to receive mail
mail_from = 'from@example.com' # Sender's email address

loop = asyncio.get_event_loop()
run = loop.run_until_complete

mailer = AmazonSESService(
    access_id=access_id,
    secret_key=secret_key,
    loop=loop)

run(mailer.send_mail(
        text="Some text",
        subject="Some subject",
        recipients=mail_to,
        sender=mail_from))
```

CHAPTER 3

Contents

3.1 Sending emails

class pushka.**AmazonSESService**(*access_id*, *secret_key*, *base_url=None*, *loop=None*, *default_sender=None*)

Amazon SES API client.

The client uses *SendEmail* method, so user's policy must grant *ses:SendEmail* permission.

send_mail(*text=None*, *subject=''*, *recipients=None*, *sender=None*, *html=None*, *attachments=None*, *reply_to=None*, *cc=None*, *bcc=None*, *return_path=None*, ***kwargs*)

Compose and send mail coroutine.

3.2 Sending SMS

class pushka.**TwilioSMSService**(*, *loop*, *account*, *token*, *default_sender=None*)

Twilio powered SMS sender, subclass of *BaseSMSService*

Parameters

- **loop** – asyncio event loop or Tornado IOLoop
- **account** (*str*) – Twilio account identifier
- **token** (*str*) – Twilio secret token
- **default_sender** (*str*) – Default sender phone number

send_sms(*, *text*, *recipients*, *sender=None*)

Send SMS asynchronously.

See *BaseSMSService.send_sms()* docs for parameters reference.

3.3 Core API

class pushka.**BaseService**(*, *loop*)

Base sender service class.

Parameters **loop** – asyncio event loop or Tornado IOLoop

new_http_client()
Create new http client.

If service is running on Tornado IOLoop, then tornado.httpclient.AsyncHTTPClient is used. When asyncio event loop is specified aiohttp client is used, in that case aiohttp library is required to be installed.

class pushka.**BasePushService**(*, *loop*)
Push notifications sender interface class mixin.

Provides empty methods for push notifications sender, which should be implemented in subclasses.

Usage example:

```
from pushka import BasePushService

class ParsePushService(BasePushService):
    """Send push notifications via Parse service."""
    @asyncio.coroutine
    def send_push(self, token=None, device_type=None, tags=None,
                  alert=None, badge=None, sound=None, **kwargs):
        do_something()
        # ...
```

add_tags(*token=None*, *tags=None*)
Add tags for registered device token, it may also be considered as subscribing to channel. Returns updated list of tags. Coroutine.

add_target(*token=None*, *device_type=None*, *tags=None*)
Register target device push token. Coroutine.

del_tags(*token=None*, *tags=None*)
Remove tags for registered device token, it may also be considered as unsubscribing from channel. Returns updated list of tags. Coroutine.

del_target(*token=None*, *device_type=None*, *tags=None*)
Unregister target device push token. Coroutine.

get_tags(*token=None*, *tags=None*)
Get tags for registered device token. Returns list of tags. Coroutine.

send_push(*token=None*, *device_type=None*, *tags=None*, *alert=None*, *badge=None*, *sound=None*, ***kwargs*)
Send push message to single or multiple devices specified by filtering options. Coroutine.

Parameters

- **token** (*str*) – Device token string
- **device_type** (*str*) – Device type: 'ios' / 'android'
- **tags** (*list*) – List of string tags / channels
- **alert** (*str*) – Notification alert text
- **badge** (*int, optional*) – Application badge number to set
- **sound** (*str, optional*) – Sound name to play, use 'default' for system sound

class pushka.**BaseMailService**(*, *loop*, *default_sender=None*)
Email sender base class.

Provides empty method for email sender, which should be implemented in subclasses.

Parameters

- **loop** – asyncio event loop or Tornado IOLoop
- **default_sender** (*str*) – default sender's email address

Usage example:

```
from pushka import BaseMailService

class SESMailService(BaseMailService):
    """Send email via Amazon SES service."""
    @asyncio.coroutine
    def send_mail(self, subject='', body=None, recipients=None, sender=None,
                  html=None, attachments=None, reply_to=None,
                  cc=None, bcc=None, **kwargs):
        do_something()
        # ...
```

send_mail(*text=None*, *subject=''*, *recipients=None*, *sender=None*, *html=None*, *attachments=None*, *reply_to=None*, *cc=None*, *bcc=None*, *return_path=None*, ***kwargs*)
Send email message. Coroutine.

Parameters

- **text** (*str*) – Plain text message in UTF-8 encoding
- **subject** (*str*) – Subject string, must be single line string
- **recipients** (*list*) – List of recipients email adresses
- **sender** (*str, optional*) – Sender address or default will be used
- **html** (*str, optional*) – HTML message in UTF-8 encoding
- **reply_to** (*str, optional*) – Reply-to address
- **cc** (*str, optional*) – Cc list
- **bcc** (*str, optional*) – Bcc list

class pushka.**BaseSMSService**(*, *loop*, *default_sender=None*)
SMS sender base class.

Provides empty method for SMS sender, which should be implemented in subclasses.

Parameters

- **loop** – asyncio event loop or Tornado IOLoop
- **default_sender** (*str*) – default sender's phone number

Usage example:

```
from pushka import BaseSMSService

class TwilioSMSService(BaseSMSService):
    """Send SMS via Twilio service."""
    @asyncio.coroutine
    def send_sms(self, text=None, recipients=None, sender=None):
        ...
```

send_sms(*, *text*, *recipients*, *sender=None*)
Send SMS. Coroutine.

Parameters

- **text** (*str*) – Plain text message in UTF-8 encoding

- **recipients** (*list*) – List of recipients phone numbers in format ['+13334445577', ...]
- **sender** (*str*) – Sender phone number or default one will be used

class pushka.**ParsePushService**(*, *loop*, *app_id*, *api_key*, *gcm_sender_id*)

Push notifications sender, which uses Parse service.

Parameters

- **loop** – asyncio event loop or Tornado IOLoop
- **app_id** – Parse application ID
- **api_key** – Parse Rest API key for application

add_target(*, *token*, *device_type*, *tags=None*)

Register device token at Parse service.

See *BasePushService.add_target()* method docs for parameters reference.

send_push(*, *alert*, *device_type*, *token=None*, *tags=None*, *badge=None*, *sound=None*, *title=None*)

Send push notification via Parse service.

See *BasePushService.send_push()* method docs for parameters reference.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

p

pushka, 3

A

add_tags() (pushka.BasePushService method), 8
add_target() (pushka.BasePushService method), 8
add_target() (pushka.ParsePushService method), 10
AmazonSESService (class in pushka), 7

B

BaseMailService (class in pushka), 8
BasePushService (class in pushka), 8
BaseService (class in pushka), 7
BaseSMSService (class in pushka), 9

D

del_tags() (pushka.BasePushService method), 8
del_target() (pushka.BasePushService method), 8

G

get_tags() (pushka.BasePushService method), 8

N

new_http_client() (pushka.BaseService method), 7

P

ParsePushService (class in pushka), 10
pushka (module), 1

S

send_mail() (pushka.AmazonSESService method), 7
send_mail() (pushka.BaseMailService method), 9
send_push() (pushka.BasePushService method), 8
send_push() (pushka.ParsePushService method), 10
send_sms() (pushka.BaseSMSService method), 9
send_sms() (pushka.TwilioSMSService method), 7

T

TwilioSMSService (class in pushka), 7