kik-python Documentation

Release 1.5.0

Kik =

Mar 31, 2017

Contents

Release v1.5.0.

kik is a thin wrapper around the Kik API.

CHAPTER 1

User Guide

This part of the documentation provides a simple introduction to the library.

User Guide

Ready to get started? This page gives an overview of how to install and use kik

Installation

You can install the library though pip, either from the command line:

```
$ pip install kik
```

or add the following line to your project's requirements.txt file:

```
kik==1.5.0
```

Example Bot

Here is a minimal echo bot using Flask

```
from flask import Flask, request, Response

from kik import KikApi, Configuration
from kik.messages import messages_from_json, TextMessage

app = Flask(__name__)
kik = KikApi(BOT_USERNAME, BOT_API_KEY)

kik.set_configuration(Configuration(webhook=YOUR_WEBHOOK))

```

```
@app.route('/incoming', methods=['POST'])
def incoming():
    if not kik.verify_signature(request.headers.get('X-Kik-Signature'), request.get_
↪data()):
        return Response(status=403)

    messages = messages_from_json(request.json['messages'])

    for message in messages:
        if isinstance(message, TextMessage):
            kik.send_messages([
                TextMessage(
                    to=message.from_user,
                    chat_id=message.chat_id,
                    body=message.body
                )
            ])

    return Response(status=200)


if __name__ == "__main__":
    app.run(port=8080, debug=True)
```

The API Client

The core of the library is the *KikApi* class, which is used to send requests to the Kik API. The client needs to be instantiated with your bot's username and API key:

```
>>> from kik import KikApi
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
```

Configuration

The Configuration API can be accessed through two functions.

KikApi.get_configuration retrieves your bots current configuration as a *Configuration* object.

```
>>> from kik import KikApi
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> config = kik.get_configuration()
>>> config.webhook
'https://example.com/incoming'
```

KikApi.set_configuration sets your bot's configuration, taking a *Configuration* object.

```
>>> from kik import KikApi, Configuration
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> config = Configuration(webhook='https://example.com/incoming')
>>> kik.set_configuration(config)
<kik.Configuration>
```

Receiving Messages

The library contains two functions that are useful when receiving messages to your webhook

The first is *KikApi.verify_signature* which is takes care of authenticating incoming requests to your webhook.

Just call the method with the provided signature header and the body of the incoming HTTP request:

```
>>> from kik import KikApi
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> kik.verify_signature(SIGNATURE_HEADER, REQUEST_BODY)
True
```

If this method returns *False*, you should ignore the incoming request, as it may be malicious.

Note: *verify_signature* must be called with the raw request body, not the parsed JSON

The second important function for receiving messages is *messages.messages_from_json*, which converts incoming messages into Python objects. After you parse the incoming request as JSON, simply pass the array of messages in the *messages* field to the function to get an array of message objects.

```
>>> from kik.messages import messages_from_json
>>> messages_from_json(messages)
[<kik.messages.TextMessage>, <kik.messages.LinkMessage>]
```

For a complete list of message types you might receive, see the Kik API Documentation.

Sending Messages

Messages are sent using *KikApi.send_messages* for the messaging API.

```
>>> from kik import KikApi
>>> from kik.messages import TextMessage
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> kik.send_messages([
...     TextMessage(
...         to='aleem',
...         chat_id='8c595a879e4140dbecb60f6c6933348bfd940cd9cbd6014e8fa51f24b5c8f74a
↪',
...         body='Test'
...     )
... ])
{}
```

Similarly, messages can be sent through the broadcasting API, using *KikApi.send_broadcast*.

```
>>> from kik import KikApi
>>> from kik.messages import TextMessage
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> kik.send_broadcast([
...     TextMessage(
...         to='aleem',
...         chat_id='8c595a879e4140dbecb60f6c6933348bfd940cd9cbd6014e8fa51f24b5c8f74a
↪',
...         body='Test'
```

```
...     )
... ])
{}
```

Messages are constructed using the *Message* subclasses. in `kik.messages`. These classes directly mirror the API message formats, with the exceptions of snake_case naming, *from* being renamed to *from_user* (as *from* is a reserved keyword in Python), and the handling of attribution and keyboards (explained below).

Attribution

All message types that support attribution are subclasses of *AttributableMessage*. To give custom attribution to these messages, simply assign their `attribution` property to a *CustomAttribution* instance.

```
>>> from kik.messages import CustomAttribution, LinkMessage
>>> message = LinkMessage()
>>> message.attribution = CustomAttribution(
...     name='A Name',
...     icon_url='http://foo.bar/anicon'
... )
```

Additionally, there are special attribution values to make a *PictureMessage* or *VideoMessage* appear to be from the camera or gallery. To achieve these effects, assign the `attribution` property of the message `PresetAttributions.CAMERA` or `PresetAttributions.GALLERY`

```
>>> from kik.messages import PresetAttributions
>>> message = PictureMessage()
>>> message.attribution = PresetAttributions.CAMERA
```

Keyboards

All message types that support keyboards are subclasses of *KeyboardMessage*. These messages contain a `keyboards` array holding any number of *Keyboard* instances.

Currently, the only supported keyboard types is *SuggestedResponseKeyboard*, which must be assigned a `responses` array containing instances of classes subclassing *SuggestedResponse* (e.g. *TextResponse*, *PictureResponse* and *FriendPickerResponse*).

```
>>> from kik.messages import TextMessage, SuggestedResponseKeyboard, \
...     TextResponse
>>> message = TextMessage()
>>> message.keyboards.append(
...     SuggestedResponseKeyboard(
...         to='aleem',
...         hidden=True,
...         responses=[TextResponse('OK')]
...     )
... )
```

Users

The User Profile API is accessed through `KikApi.get_user<kik.KikApi.get_user()`, which retrieves a user's profile from their username.

The function returns a *User*, containing the user's profile

```
>>> from kik import KikApi
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> user = kik.get_user('aleem')
>>> user.first_name
'Johnny'
```

Kik Codes

The Kik Code creation API is accessed through *KikApi.create_code*. This function takes an optional data parameter which will be embedded in the Kik Code, and returned in the *ScanDataMessage* you receive when the user scans the code.

create_code returns a *Code*, which allows you to get a URL for the code.

```
>>> from kik import KikApi
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> code = kik.create_code({'some': 'data'})
>>> code.url()
'https://api.kik.com/v1/code/161d764eeebf050fba373ae8cef9f5052524019a'
```

CHAPTER 2

API Documentation

Complete documentation for specific classes and methods.

API Documentation

API Client

Client for interacting with the Kik API.

class kik.**KikApi**(*bot*, *api_key*)

Generic API Client for all Kik API features.

Parameters

- **bot** (*str*) – Your bot's username
- **api_key** (*str*) – Your bot's API key

create_code(*data=None*)

Creates a Kik Code for your bot.

Parameters **data** (*dict, str*) – (optional) Data to embed in the code, which will be returned in a *scan-data* message when the code is scanned.

Returns A *Code* representing the code.

Return type *kik.Code*

Usage:

```
>>> from kik import KikApi
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> kik.create_code(data='somedata')
<kik.Code>
```

get_configuration()
Retrieves your bot's configuration

Returns A *Configuration* representing the configuration data.

Return type *kik.Configuration*

Usage:

```
>>> from kik import KikApi
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> kik.get_configuration()
<kik.Configuration>
```

get_user(*username*)
Gets a user's profile data.

Parameters **username** (*str*) – List of *Message* to be sent.

Returns A *User* containing the user's profile data.

Return type *kik.User*

Note: In order to fetch a user's profile, the user must be a subscriber to your bot

---

Usage:

```
>>> from kik import KikApi
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> kik.get_user('aleem')
<kik.User>
```

send_broadcast(*messages*)
Sends a batch of messages though the broadcast API.

Parameters **messages** (*list[kik.messages.Message]*) – List of Message to be sent.

Returns A dict containing the response from the API.

Return type dict

Note: Subject to limits on the number of messages sent, documented at https://dev.kik.com/#/docs/messaging#broadcasting.

Usage:

```
>>> from kik import KikApi
>>> from kik.messages import TextMessage
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> kik.send_broadcast([
>>>     TextMessage(
>>>         to='ausername',
>>>         chat_id=
↪'2e566cf66b07d9622053b2f0e44dd14926d89a6d61adf496844781876d62cca6',
>>>         body='Some Text'
>>>     )
>>> ])
{}
```

send_messages(*messages*)
Sends a batch of messages.

Parameters **messages** (*list[kik.messages.Message]*) – List of *Message* to be sent.

Returns A dict containing the response from the API

Return type dict

Note: Subject to limits on the number of messages sent, documented at https://dev.kik.com/#/docs/messaging#sending-messages.

Usage:

```
>>> from kik import KikApi
>>> from kik.messages import TextMessage
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> kik.send_messages([
>>>     TextMessage(
>>>         to='ausername',
>>>         chat_id=
↪'2e566cf66b07d9622053b2f0e44dd14926d89a6d61adf496844781876d62cca6',
>>>         body='Some Text'
>>>     )
>>> ])
{}
```

set_configuration(*config*)
Sets your bot's configuration

Parameters **config** (kik.Configuration) – A *Configuration* containing your bot's new configuration

Returns A *Configuration* containing your bot's new configuration, as confirmed by

the server :rtype: kik.Configuration

Usage:

```
>>> from kik import KikApi, Configuration
>>> kik = KikApi(BOT_USERNAME, BOT_API_KEY)
>>> config = Configuration(webhook='https://example.com/incoming')
>>> kik.set_configuration(config)
<kik.Configuration>
```

verify_signature(*signature*, *body*)
Verifies that a request body correctly matches the header signature. For more on signatures see https://dev.kik.com/#/docs/messaging#receiving-messages.

In Python 3, *body* must be a bytestring

User Model

Model for working with user profiles

class kik.**User**(*first_name*, *last_name*, *profile_pic_url=None*, *profile_pic_last_modified=None*, *timezone=None*, ***kwargs*)

Model representing a Kik user's profie information. Created using *kik.KikApi.get_user()*.

Variables

- **first_name** (*str*) – The user's first name
- **last_name** (*str*) – The user's last name
- **profile_pic_url** (*str*) – URL for the users's profile picture
- **profile_pic_last_modified** – Timestamp indicating when the user's profile picture was last modified, to allow for caching
- **timezone** – String indicating the timezone of the user (ex. America/Toronto)

Vartype profile_last_modified: int

Vartype timezone: str

Code Model

Model for working with Kik Codes

class kik.**Code**(*id*, ***kwargs*)

Model representing a Kik Code. Can be instantiated with a Kik Code ID, or created using *kik.KikApi.create_code()*.

Variables **id** (*str*) – The ID for the Kik Code.

class Colors

Kik Code color mapping, taken from https://dev.kik.com/#/docs/messaging#kik-code-colors

Code.**url**(*color=None*)

Returns the URL for the Kik Code.

Parameters **color** (*int*) – (optional) Sets the color of the Kik Code. For options see *kik.Code.Colors*

Configuration Model

Model for working with your bot's configuration.

class kik.**Configuration**(*webhook*, *features=None*, *static_keyboard=None*)

Model for your bot's configuration.

Parameters

- **webhook** (*str*) – URL the API will send incoming messages to
- **features** (*dict*) – Feature flags to set
- **static_keyboard** (kik.messages.keyboards.Keyboard) – The static keyboard to set

Messages

These classes directly mirror the message types used by the API.

class kik.messages.**Message**(*type*, *to=None*, *id=None*, *chat_id=None*, *mention=None*, *participants=None*, *from_user=None*, *delay=None*, *read_receipt_requested=None*, *timestamp=None*, *metadata=None*)
Parent class for all messages.

class kik.messages.**TextMessage**(*to=None*, *chat_id=None*, *body=None*, *keyboards=None*, *mention=None*, *delay=None*, *type_time=None*, ***kwargs*)
A full link message object, as documented at https://dev.kik.com/#/docs/messaging#text.

class kik.messages.**LinkMessage**(*to=None*, *chat_id=None*, *url=None*, *title=None*, *text=None*, *pic_url=None*, *no_forward=None*, *kik_js_data=None*, *keyboards=None*, *attribution=None*, *mention=None*, *delay=None*, ***kwargs*)
A full link message object, as documented at https://dev.kik.com/#/docs/messaging#link.

class kik.messages.**PictureMessage**(*to=None*, *chat_id=None*, *pic_url=None*, *keyboards=None*, *attribution=None*, *mention=None*, *delay=None*, ***kwargs*)
A full picture message object, as documented at https://dev.kik.com/#/docs/messaging#picture.

class kik.messages.**VideoMessage**(*to=None*, *chat_id=None*, *video_url=None*, *loop=None*, *muted=None*, *autoplay=None*, *no_save=None*, *keyboards=None*, *attribution=None*, *mention=None*, *delay=None*, ***kwargs*)
A full video message object, as documented at https://dev.kik.com/#/docs/messaging#video.

class kik.messages.**StartChattingMessage**(*chat_type=None*, ***kwargs*)
A full start-chatting message object, as documented at https://dev.kik.com/#/docs/messaging#start-chatting.

class kik.messages.**ScanDataMessage**(*data=None*, *chat_type=None*, ***kwargs*)
A full scan-data message object, as documented at https://dev.kik.com/#/docs/messaging#scan-data.

class kik.messages.**StickerMessage**(*sticker_pack_id=None*, *sticker_url=None*, *chat_type=None*, ***kwargs*)
A full sticker message object, as documented at https://dev.kik.com/#/docs/messaging#sticker.

class kik.messages.**IsTypingMessage**(*is_typing=None*, *chat_type=None*, ***kwargs*)
A full is-typing message object, as documented at https://dev.kik.com/#/docs/messaging#is-typing.

class kik.messages.**ReceiptMessage**(*message_ids=None*, ***kwargs*)
Parent class for all receipts.

class kik.messages.**DeliveryReceiptMessage**(***kwargs*)
A full delivery-receipt message object, as documented at https://dev.kik.com/#/docs/messaging#receipts.

class kik.messages.**ReadReceiptMessage**(***kwargs*)
A full read-receipt message object, as documented at https://dev.kik.com/#/docs/messaging#receipts.

class kik.messages.**FriendPickerMessage**(*picked=None*, *chat_type=None*, ***kwargs*)
A friend picker message, as documented at https://dev.kik.com/#/docs/messaging#friend-picker-response-object.

class kik.messages.**UnknownMessage**(*type*, *to=None*, *id=None*, *chat_id=None*, *mention=None*, *participants=None*, *from_user=None*, *delay=None*, *read_receipt_requested=None*, *timestamp=None*, *metadata=None*)
This message type is returned by the message factory when it encounters an unknown message type.

It's *type* attribute is set to the type of the message, and it's *raw_message* attribute contains the raw JSON message received

Message Utilities

kik.messages.**messages_from_json**(*messages*)
Converts incoming JSON format messages into message objects.

Parameters **messages** (*list[dict]*) – A list of messages in JSON format.

Returns A list of messages as Python classes.

Return type list[*kik.messages.Message*]

Attribution

class kik.messages.attribution.**Attribution**
Parent class for all attribution types

class kik.messages.**CustomAttribution**(*name=None*, *icon_url=None*)
Bases: *kik.messages.attribution.Attribution*

Attribution class for custom attributions, as documented at https://dev.kik.com/#/docs/messaging#attribution

Usage:

```
>>> from kik.messages import CustomAttribution, LinkMessage
>>> message = LinkMessage()
>>> message.attribution = CustomAttribution(
>>>     name='A Name',
>>>     icon_url='http://foo.bar/anicon'
>>> )
```

class kik.messages.attribution.**PresetAttribution**(*preset_name*)
Bases: *kik.messages.attribution.Attribution*

Attribution class for the preset attribution types (e.g. “gallery” or “camera”)

class kik.messages.attribution.**PresetAttributions**
List of preset attribution types.

Valid only on *PictureMessage* and *VideoMessage*.

Variables

- **GALLERY** (*kik.message.attribution.PresetAttribution*) – Makes the message appear to be from a user’s gallery.
- **CAMERA** (*kik.message.attribution.PresetAttribution*) – Makes the message appear to be from a camera.

Usage:

```
>>> from kik.messages import PresetAttributions, PictureMessage
>>> message = PictureMessage()
>>> message.attribution = PresetAttributions.CAMERA
```

class kik.messages.attributable_message.**AttributableMessage**(*attribution=None*, ***kwargs*)
Parent class for messages that support attribution.

Keyboards and Responses

class kik.messages.keyboards.**Keyboard**(*type*, *to=None*, *hidden=None*)
Parent class for all keyboards.

class kik.messages.**SuggestedResponseKeyboard**(*to=None*, *hidden=None*, *responses=None*)
Bases: *kik.messages.keyboards.Keyboard*

A suggested response keyboard as documented at https://dev.kik.com/#/docs/messaging#keyboards.

Parameters

- **to** (*str*) – (optional) User who will see this keyboard. If None, the keyboard will be shown to all users who don't have another keyboard set.
- **hidden** (*bool*) – (optional) If True, this keyboard will be hidden until the user chooses to see suggested responses.
- **responses** (*list[kik.message.responses.SuggestedResponse]*) – (optional) A list of SuggestedResponse. Defaults to an empty list.

class kik.messages.responses.**SuggestedResponse**(*type*, *metadata=None*)
Base class for all responses for *SuggestedResponseKeyboard*.

class kik.messages.**TextResponse**(*body*)
Bases: *kik.messages.responses.SuggestedResponse*

A text response, as documented at https://dev.kik.com/#/docs/messaging#suggested-response-keyboard.

class kik.messages.**FriendPickerResponse**(*body=None*, *min=None*, *max=None*, *preselected=None*)
Bases: *kik.messages.responses.SuggestedResponse*

A friend picker response as documented on https://dev.kik.com/#/docs/messaging#friend-picker-response-object

class kik.messages.**PictureResponse**(*pic_url*, *metadata*)
Bases: *kik.messages.responses.SuggestedResponse*

A picture response as documented on https://dev.kik.com/#/docs/messaging#picture-response-object

class kik.messages.keyboard_message.**KeyboardMessage**(*keyboards=None*, *chat_type=None*, ***kwargs*)
Parent class for messages that support keyboards.

Exceptions

exception kik.**KikError**(*message*, *status_code*, *content=None*)
Exception raised by all API errors. The exception message is set to the server's response.

Parameters

- **status_code** (*int*) – Status code returned by the API call
- **content** (*string*) – Content returned by the API call

CHAPTER 3

Indices and tables

- genindex

Index