Shawk Documentation

Release latest

Hawkins

Sep 27, 2017

Table of Contents

Shawk is a Python module designed to make sending and receiving SMS text messages easy for devices running Python. With internet of things compatibility in mind, this module has been designed to function fully on Raspberry Pi devices.

If you're new to Shawk, I suggest you check out the *Getting Started* page.

CHAPTER 1

Module Index

- modindex

Getting Started

The idea of Shawk is to provide an easy-to-use interface for sending and receiving text messages. This getting started guide will give you a quick run-down on the highlights of how to use Shawk.

Installing Shawk

Installing Shawk is easy. Simply run *pip install shawk* to download and install the latest version.

To best understand how to use shawk, let's break it down into a few features:

- Creating a client
- Adding/removing contacts
- Sending messages
- Manually retrieving messages
- Automatically retrieving messages
- Defining behavior for new messages

Creating a client

The first thing you'll want to do after installing shawk is create a client. This is the main interface you'll use in Shawk. You can define multiple clients, but for the sake of this tutorial, we'll just use one.

We'll say our Gmail login username is *"username@gmail.com"* and our password is *"password"*.

We can create our client like this:

```
import shawk

user = "username@gmail.com"
password = "password"
client = shawk.Client(user, password, inbox=False)
```

Adding/removing contacts

Adding contacts will be helpful for understanding who texted your client or specifying who you would like to text in an easy and readable manner.

You can add them as such:

```
client.add_contact(1234567890, 'Verizon', 'Josh')
```

You can similarly remove them in any of the following ways:

```
client.remove_contact(contact) # Where contact is some Contact object. More on that␣
↪later.
# or
client.remove_contact(number=1234567890)
# or
client.remove_contact(name='Josh')
```

Sending messages

You're limited to texting only your contacts you've previously defined, or those whose explicit address you have obtained. This is, in-part, a restriction to prevent spam bots from abusing Shawk, but also a means of simplifying Number to Address mappings.

So, after you've defined a contact, you can text them by specifying their name, number, or Contact:

```
client.send('Hey, Josh!', name='Josh')
# or
client.send('Hey, Josh!', number=1234567890)
# or
client.send('Hey, Josh!', contact)
```

Manually retrieving messages

There are two ways to get new messages in Shawk: Manually or Automatically.

We can get the new ones manually by first setting up our inbox and refreshing it:

```
client.setup_inbox(password, auto=False)

client.refresh_inbox()
```

This will handle the IMAP server connection for retrieving new messages to your Gmail account.

You can actually use a distinct Gmail account from the one you use to send messages, but we won't focus on that for this simple tutorial. As always, read the rest of the docs if you'd like to know more about that.

Automatically retrieving messages

You can configure Shawk to automatically retrieve new messages for you pretty easily. This will cause the client to refresh its inbox periodically on time interval (found in client.refresh_interval).

To do this, setup your inbox as such:

```
client.setup_inbox(password, auto=True)
```

Boom. Done. Your client now automatically pings the server and looks for new messages!

You can configure the time interval with *client.set_interval()*, more on that in the docs.

Defining behavior for new messages

By default, shawk will simply print the contents of the new messages when it finds them.

That's not particularly useful, so we'll let you dictate how Shawk *should* be used.

This default behavior can be overridden by defining one or more handler functions that receive a client, message, and a regex match object. These handler function are just callback functions with the *@client.text_handler(regex)* decorator, where regex is some regular expression in string form.

If no regex is provided, then the function is considered the default case handler, and will be used whenever any other handler regex's do not match.

You can define your own behavior as follows:

```
c = shawk.Client('username@gmail.com', 'password', inbox=True, auto=True)

@client.text_handler()
def handler(client, msg):
    print("Hey, we're popular! %s texted us!" % msg.sender)
    print("Received: %s" % msg.text)

    client.send("I got your text!", msg.sender)

# Or with a regex
@client.text_handler('^Print (.*)', 'i') # Starts with "print", matches text␣
↪following. Case insensitive.
def print_dot_z(client, msg, match):
    print(match.group(1))
```

Naturally, you'll do something a bit more meaningful in your handler functions. But since they're just simple python functions, you've got free reign to interface with your scripts however you like.

I hope you've found that Shawk is pretty easy to use, yet very powerful, since it allows your users to provide input and receive output via SMS, for free.

If you encounter any issues or have any questions, you can post them on GitHub://hawkins/shawk.

Configuring Gmail

SMTP

GMail should work with SMTP out of the box, but currently sending messages is limited to **roughly 100 messages per 24 hours**. After your limit is reached, the limit will be reset 24 hours later.

However, you can greatly increase these limits (**10,000 messages per 24 hours** by creating a Google Apps account. You can find out more about this here.

IMAP

1. Create a Google / Gmail account for your application (It doesn't have to be new, but I suggest you do not use your personal / work account).
2. Follow instructions to set up an App password here and be sure to select app "Other (custom name)" and enter anything to help you remember it. I.e., "shawk".

    Be careful though, as this password grants full access to your Google account! Do not share this with anyone and do not commit it!

    I recommend creating a *secure.ini* file which your scripts will read in your password from. Add this file to your *.gitignore* to prevent accidentally commiting it.

3. Follow instructions to enable IMAP access to Gmail here. You only need to complete Step 1, don't worry about Step 2. The Shawk library handles Step 2 for you.

Client

Client is the bread and butter of the Shawk framework. You'll use this to connect to a Gmail account in both SMTP and IMAP protocols.

Typically the workflow will resemble creating a Client, adding contacts, configuring an inbox for IMAP, and listening for messages.

shawk.Client

Define the Client interface in Shawk.

class `shawk.Client.`**`Client`**(*user*, *pwd*, *inbox=True*, *auto=True*)

Bases: `object`

Define the main Shawk interface.

`add_contact`(*number*, *carrier*, *name=None*)

Create a new Contact instance and add to contacts.

You can also pass a list of numbers, carriers, and names to create multiple at once.

`auto_refresh`()

Refresh the inbox automatically on an interval.

`disable_auto_refresh`()

Disable auto refresh of inbox.

`enable_auto_refresh`(*start=True*)

Enable auto refresh of inbox.

Will also begin refreshing now, but can be disabled with *start=False*.

`export_contacts`(*path*)

Export the current contacts to a Shawk CSV file.

get_contact(*message*)
Return the Contact from a given message's sender.

Returns None if sender not in contacts.

get_contact_from_address(*address*)
Return the Contact matching a given address.

Returns None if not in contacts.

get_refresh_interval()
Return the refresh interval for auto refresh.

import_contacts(*path*)
Import contacts from a Shawk CSV file.

print_contacts()
Print the contacts

refresh()
Refresh the inbox only once.

remove_contact(*contact=None*, *number=None*, *name=None*)
Remove a contact from contacts.

send(*message*, *contact=None*, *address=None*, *number=None*, *name=None*, *carrier=None*)
Send a message.

Can determine a contact to use via a number of different specifications, but it is advised to specify a Contact object if possible.

However, passing a specific address takes precedence over any other input.

set_refresh_interval(*time*)
Define the refresh interval for auto refresh.

setup_inbox(*password*, *user=None*, *folder='INBOX'*, *refresh=False*, *auto=False*, *ssl=True*)
Configure an IMAP connection for receiving SMS.

Optionally configure behaviours such as auto-refresh, refresh immediately once configured, or specify a folder.

Folder specifications are useful if you configure your Gmail account to filter messages from certain senders to be moved to a specific folder, that way they don't clutter your Gmail Inbox folder.

text_handler(*pattern=None*, *modifiers=''*)
Define a decorator that accepts a regular expression in string form for handlers.

Sets the default text handler if no string is provided.

Contact

Contact defines the structure of the contacts used in Shawk. These contain an address, a number, and an optional name.

The real magic isn't so much in the contact, but in how the Client uses the contact for simplifiyng receiving and sending messages.

shawk.Contact

Define the Contact representation in Shawk.

class `shawk.Contact.`**`Contact`**(*number*, *carrier*, *name=None*)
Bases: `object`

Define the structure for contacts.

`get_address`()
Return the email address of the Contact.

`get_carrier`()
Return the carrier of the Contact.

`get_name`()
Return the name of the Contact.

`get_number`()
Return the number of the Contact.

Message

Message defines the structure of the messages used in Shawk. These contain a sender, a text body, and an international date format of when the message was received (only valid for received messages).

shawk.Message

Define the Message representation in Shawk.

class `shawk.Message.`**`Message`**(*text*, *sender*, *date=None*)
Bases: `object`

Define the structure for messages.

Gateways

Gateways defines the SMS Carrier and Gateway support for Shawk.

This contains a dictionary of Carrier -> Gateway mappings, as well as a regex for matching an email address to a supported domain.

shawk.Gateways

Define the Gateways supported in Shawk and a regex to match supported address patterns.

`shawk.Gateways.`**`sms_to_mail`**(*number*, *carrier*)
Return the email address of some number and some carrier mapped to a gateway.

Python Module Index

Index