H@B Twilio Workshop Documentation

Release 0.1

Kyle Conroy

October 03, 2013

CONTENTS

Today we're going to learn how to use Twilio to build a simple but powerful SMS voting application. You'll be able to use this application the next time your friends or student group need to hold elections or decide whether to go to La Val's or Bongo Burger for lunch.

CHAPTER

ONE

INITIAL SETUP

Before we can build a Twilio application, there are some steps you'll need to complete. This setup shouldn't take more than five minutes.

1.1 Install a Text Editor

Now that you've signed, up, we need to make sure you can edit the workshop code. **If you already have a text-editor or IDE of choice, skip this section**.

- Windows - Download and install Notepad++
- OS X - Download and install Text Wrangler
- Linux - Install gedit via your package manager

1.2 Install Python

Open up a terminal or command prompt window and type the following

```
$ python --version
```

If the output contains either Python 2.6.x or Python 2.7.x, your Python installation is ready to go. Some of you may have Python 3.x installed. Sadly, the twilio-python helper library only works with Python 2.6 or Python 2.7.

Find and download the installation for your operating system.

- Python 2.7.3 Windows Installer
- Python 2.7.3 Windows X86-64 Installer
- Python 2.7.3 OS X Installer
- Python 2.7.3 compressed source tarball

More downloads are available on the Python downloads page.

Once you are finished, opening up Terminal (OS X) or Powershell (Windows) and verify the output is now the same

```
$ python --version
Python 2.7.3
```

1.3 Download Workshop Materials

Download the workshop materials as a zipfile. You can also clone this repository if you have git installed.

```
$ git clone https://github.com/twilio/calworkshop.git
```

To verify that everything is working correctly, run `check.py` and make sure the output matches below

```
$ cd calworkshop
$ python check.py
:)
```

CHAPTER

TWO

TWILIO QUICKSTART

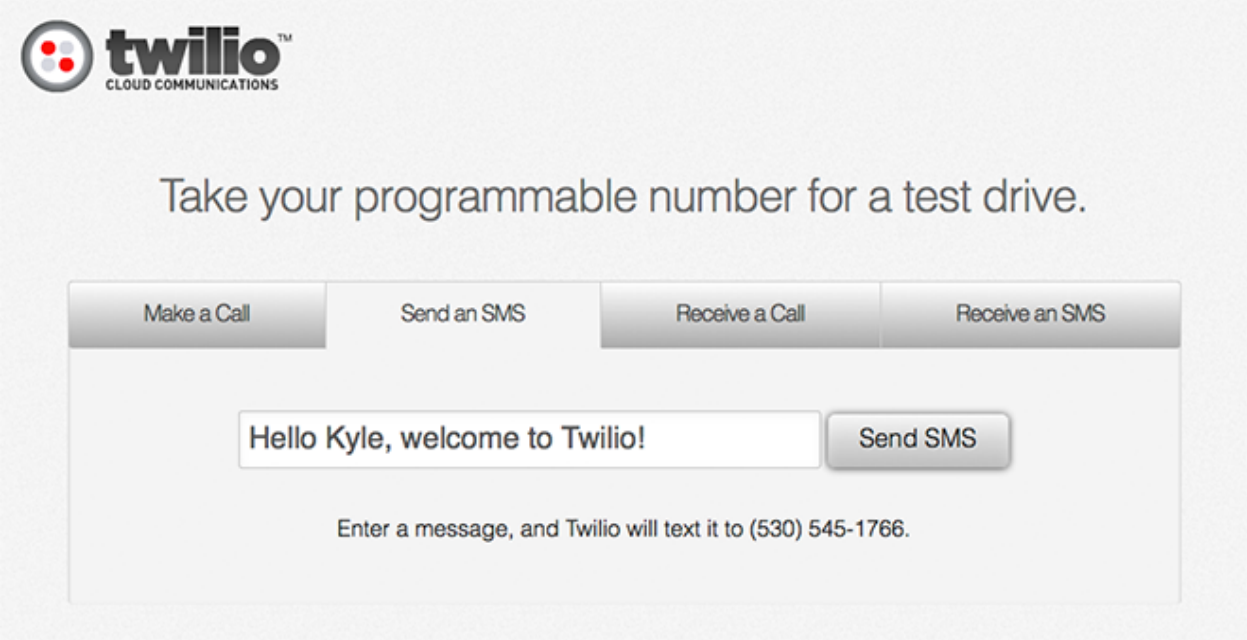
The next section will take you on a whirlwind tour of the features of the Twilio API. You'll learn how to make phones ring, send text messages, and record voice messages.

Make sure you've completed the *Initial Setup* section before continuing.

2.1 Create a Twilio Account

First, sign up for a free Twilio account. You won't need a credit card, but you will need a phone number to prove you aren't a robot. Once you've signed up, you'll have your own Twilio phone number. We'll use this number for the rest of the workshop.

After you've created your account and verified your phone number, you should end up at a screen that looks like this.



This is your first chance to test out what Twilio can do. Send yourself a text message and receive a call. Congratulations, you've used Twilio for the first time!

However, how would you do this from your own code? I'm glad you asked.

2.2 Hello World - SMS

Let's send a text message using Python and the Twilio REST API. Open the `send_sms.py` file in your text editor. First, replace the dummy account credentials with those of your own. Your account credentials can be found on the top of your Twilio account dashboard.

```
from twilio.rest import TwilioRestClient

TWILIO_ACCOUNT_SID = ''
TWILIO_AUTH_TOKEN = ''
```

On the next line, set `TO_NUMBER` to the number you used to sign up with Twilio. During your free trial, you're only allowed to make calls and send messages to numbers you've verified.

Set `FROM_NUMBER` to your new Twilio number. If you can't remember it, check the numbers section of your account dashboard.

Pick any message less than 140 characters to serve as the body.

```
TO_NUMBER = ''        # Your verified phone number
FROM_NUMBER = ''      # Your Twilio phone number
BODY = 'Hello World' # SMS message
```

With the above information, we construct a Twilio REST API client. We'll use this to create and send a new text message.

```
client = TwilioRestClient(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)
client.sms.messages.create(to=TO_NUMBER, from_=FROM_NUMBER, body=BODY)
```

We're now ready to send a SMS message.

```
$ python send_sms.py
```

Your phone should be getting a message in a few seconds.

2.3 Hello World - Voice

It's time to make a call to your phone using the REST API. Open `make_call.py` in your text editor and, just like the last section, fill in your account credentials and phone numbers details.

Again, we construct a Twilio REST API client, but this time, we create a new call with it.

```
TWIML_URL = 'http://twimlets.com/message?Message=Hello+World'

client = TwilioRestClient(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)
client.calls.create(to=TO_NUMBER, from_=FROM_NUMBER, url=TWIML_URL)
```

Run the script to start the call.

```
$ python make_call.py
```

Your phone should start ringing momentarily.

2.4 Introduction to TwiML

We've successfully made a phone ring, but how do we actually control call flow? TwiML is the answer. TwiML is a set of instructions you can use to tell Twilio what to do when you receive an incoming call or SMS.

When someone makes a call or sends an SMS to one of your Twilio numbers, Twilio will look up the URL associated with that phone number and make a request to that URL. Twilio will read TwiML instructions at that URL to determine what to do: record the call, play a message for the caller, prompt the caller to press digits on their keypad, etc.

TwiML is an XML-based language and consists of five basic verbs.

- Say
- Play
- Gather
- Record
- Dial

To see how these verbs work, let's take a look at the last section. When we called your phone, a robot answered with a "Hello World" message. We now know that TwiML powered that call, so let's take a look. Open http://twimlets.com/message?Message=Hello+World in your browser.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response><Say>Hello World</Say></Response>
```

This TwiML will do text-to-speech and say "Hello World". For outgoing calls, we choose the TwiML URL at the time of the call. For incoming calls, we set a TwiML URL that is fetched every time someone calls into our Twilio number.

Go to your Twilio numbers page and click on your phone number. Change the "Voice URL" field to the Hello World URL

```
http://twimlets.com/message?Message=Hello+World
```

Now when you call your number, you should hear a "Hello World" greeting.

2.4.1 Twimlbin

TwiML can be hosted anywhere. It can be a static XML document or created dynamically by a web application. To make developing Twilio applications easier, you can host your TwiML on Twimlbin.

To create a new bin, go to the Twimlbin homepage and click "Create a new Twimlbin". You can then use set your phone number's Voice URL to the "Public" URL of your Twimlbin.

For the next sections, copy the sample TwiML and paste it into your Twimlbin for easy experimentation.

2.5 Call Forwarding

The Dial verb allows you to connect calls to other people. The following TwiML will forward any call to your Twilio phone number to your personal. Once you've wired up this TwiML to your number, get a neighbor to test it out.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Say>Please wait while we forward your call</Say>
  <Dial>YOUR PHONE NUMBER</Dial>
</Response>
```

2.6 Voice Mailbox

Recording audio is accomplished through the Record verb. The Record verb will play a beep and wait until a user presses # or hangs up. Copy this TwiML into your bin and save. You can now leave messages on your number.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Say>After the beep, record your message</Say>
  <Record/>
</Response>
```

After you're done recording your message, hang up. Twilio begins processing the recording right after your done. Head to your recording log to listen to your message.

2.7 Private Conference Line

Many times during project assignments, you just need to get everyone on the same page. You can now have your own private conference line using the Conference noun and Dial verb. Put the following TwiML into your bin and save. Give your Twilio number to a few people around you. Have everyone call in and start up a conversation.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Dial>
    <Conference>my-conference-room</Conference>
  </Dial>
</Response>
```

The text inside the Conference element is the conference room name. Since the name stays the for every phone call, all participants end up in the same conference.

2.8 Swiss-Army Phone Number

Equipped with the knowledge of TwiML, you can now bend your Twilio phone number to your will. You've forwarded a call, recorded a message, and started a private conference line. Your phone is now yours to control.

But don't think we're done yet, our second act will be creating an application for *SMS Polling and Voting*.

2.9 Additional Information

- Twilio REST API - Calls Resource
- Twilio REST API - SMS/Messages Resource
- Twilio TwiML

CHAPTER

THREE

SMS POLLING AND VOTING

Whether at a hackathon or a student group meeting, you'll often need to vote on items. Elections, food, or nominations - all these situations can be handled via SMS voting.

We'll create a simple Twilio application to record and report votes via SMS.

3.1 Ballot Format

For this poll, ballots don't need a format. To vote, text your choice to your Twilio number.

For example, to vote for Cal, text the following:

```
cal
```

To see all the votes, we'll use a simple Python script and the twilio-python helper library.

```
from twilio.rest import TwilioRestClient

client = TwilioRestClient("ACCOUNT_SID", "AUTH_TOKEN")

for msg in client.sms.messages.iter():
    print msg.body
```

The `iter` function efficiently fetches all your SMS messages via the Twilio REST API. Under the covers, the function fetches pages of the SMS Messages list resource as they are needed.

However, this script will fail if you have multiple Twilio phone numbers. To fix this, we'll filter messages based on the phone number they were sent to.

```
from twilio.rest import TwilioRestClient

TWILIO_PHONE_NUMBER = "(999) 999 - 9999"

client = TwilioRestClient("ACCOUNT_SID", "AUTH_TOKEN")

for msg in client.sms.messages.iter(to=TWILIO_PHONE_NUMBER):
    print msg.body
```

Still, we're only seeing the contents of the messages.

3.2 Tallying Votes

In our election, participants can only vote once. Therefore, each message should count for a single vote. We'll use a default dictionary to keep track of votes.

A Python `defaultdict` is a regular dictionary, but with default values for the keys. For example, a regular dictionary will throw a `KeyError` if you access a key that doesn't exist.

```
>>> s = {}
>>> s['hey']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'hey'
```

A `defaultdict` will instead return the default value for the type of object it contains.

```
>>> s = defaultdict(int)
>>> s['hey']
0
```

Instead of just printing the message body, we now use the message body as a key for the vote dictionary.

```
from twilio.rest import TwilioRestClient
from collections import defaultdict

TWILIO_PHONE_NUMBER = "(999) 999 - 9999"

votes = defaultdict(int)

client = TwilioRestClient("ACCOUNT_SID", "AUTH_TOKEN")

for msg in client.sms.messages.iter(to=TWILIO_PHONE_NUMBER):
    votes[msg.body] += 1

for vote, total in votes.items():
    print "{} {}".format(vote, total)
```

We can now see tallies for each vote. However, this code is very brittle. Let's normalize the message bodies so that similar votes (like for `foo` and `Foo`) count for the same option.

```
from twilio.rest import TwilioRestClient
from collections import defaultdict

TWILIO_PHONE_NUMBER = "(999) 999 - 9999"

votes = defaultdict(int)

client = TwilioRestClient("ACCOUNT_SID", "AUTH_TOKEN")

for msg in client.sms.messages.iter(to=TWILIO_PHONE_NUMBER):
    votes[msg.body.upper()] += 1

for vote, total in votes.items():
    print "{} {}".format(vote, total)
```

3.3 Preventing Cheaters

Cheaters never prosper, and currently they don't get caught either. Any person can vote any number of times. We'll keep track of every number that's voted, only allowing them a single vote. To do this, phone numbers will be added to a set and checked before each vote is tallied.

```
from twilio.rest import TwilioRestClient
from collections import defaultdict

TWILIO_PHONE_NUMBER = "(999) 999 - 9999"

votes = defaultdict(int)
voted = set()

client = TwilioRestClient("ACCOUNT_SID", "AUTH_TOKEN")

for msg in client.sms.messages.iter(to=TWILIO_PHONE_NUMBER):
    if msg.from_ in voted:
        continue

    votes[msg.body.upper()] += 1
    voted.add(msg.from_)

for vote, total in votes.items():
    print "{} {}".format(vote, total)
```

3.4 Graphing the Results

No election is complete without graphs. Let's take the results from the previous section and make some pretty graphs. We'll use the Google Graph API due to its simplicity and price (free).

```
import urllib
from twilio.rest import TwilioRestClient
from collections import defaultdict

TWILIO_PHONE_NUMBER = "(999) 999 - 9999"

votes = defaultdict(int)
voted = set()

client = TwilioRestClient("ACCOUNT_SID", "AUTH_TOKEN")

for msg in client.sms.messages.iter(to=TWILIO_PHONE_NUMBER):
    if msg.from_ in voted:
        continue

    votes[msg.body.upper()] += 1
    voted.add(msg.from_)

url = "https://chart.googleapis.com/chart"

options = {
    "cht": "pc",
    "chs": "500x500",
    "chd": "t:" + ",".join(map(str, votes.values())),
```

```
    "chl": "|".join(votes.keys()),
}

print url + "?" + urllib.urlencode(options)
```

3.5 Existing Solutions

Wedgies is a very similar concept built on top of Twilio that provides quite a few cool features beyond basic SMS polling, like Twitter polls, embedded polls, and realtime results.