Receive an SMS with Python

In this post, we'll show you how to receive SMS messages on your Vonage virtual number.

To receive inbound SMS, you need to create a publicly-accessible webhook and configure your Vonage account to use it. We'll cover that process in this post and you can find the source code on Github.

Prerequisites

You'll use Python 3 and Flask to write your webhook. You can install `flask` with the pip3 package manager:

Lastly, you'll need to install our Vonage CLI. You will use this to purchase a Vonage virtual number (if you haven't already got one) and to send a test SMS.

When you have all these things, you're ready to start coding!

Create the Webhook for Inbound SMS

When Vonage receives an SMS on your virtual number, it looks to see if you have configured a webhook on your account so that it can forward the SMS to your application. You can configure this webhook to apply either to a specific number or your account as a whole.

Vonage makes either a `GET` or `POST` request depending on which "HTTP Method" you have configured in your account settings. Because you want your webhook to work regardless of your account settings you will code the handler to accept both types of request.

Create a file called `sms-receive.py` and enter the following code:

```
from flask import Flask, request, jsonify
from pprint import pprint

app = Flask(__name__)

@app.route('/webhooks/inbound-sms', methods=['GET', 'POST']
def inbound_sms():
    if request.is_json:
        pprint(request.get_json())
    else:
        data = dict(request.form) or dict(request.args)
        pprint(data)

    return ('', 204)

app.run(port=3000)
```

This code captures any request sent to the `/webhooks/inbound-sms` endpoint and unpacks it:

- **`request.is_json`**: Checks to see if the request is in JSON format. By default a request is considered to

include JSON data if the **`mimetype`** is **`application/json`** or **`application/*+json`**
- **`request.args`**: Retrieves the key/value pairs in the URL query string.
- **`request.form`**: Retrieves the key/value pairs in the body, from a HTML post form, or any request that isn't JSON-encoded.

This approach enables your application to retrieve the message details from either a `GET` or `POST` request.

Having parsed the incoming request data it pretty-prints it to the terminal using `pprint` and then sends a `HTTP 204` status response (success, no content) to Vonage. It is important to return this response otherwise Vonage will keep trying to redeliver.

Make Your Webhooks Accessible

For Vonage's APIs to make requests to your webhook endpoint it must be accessible over the public internet.

A great tool for exposing your local development environment to the internet is `ngrok`. Our tutorial shows you how to install and use it.

Launch `ngrok` using the following command:

Make a note of the public URLs that `ngrok` creates for you. These will be similar to (but different from) the following:

```
http://066d53c9.ngrok.io -> localhost:3000
https://066d53c9.ngrok.io -> localhost:3000
```

On their free plan, every time you restart `ngrok` the URLs change and you will have to update your delivery receipt URL. This is extra work you could do without, so leave it running for the duration of this tutorial.

Purchase a Vonage Virtual Number

If you don't already have a Vonage virtual number to use for this example, you'll need to buy one. You can do this in the developer dashboard but it's often more convenient to perform these account management tasks using our CLI. So that's what we'll do here.

To check which numbers are available, use `vonage numbers:search [COUNTRYCODE]`, passing it your two-character country code. For example, `GB` for Great Britain or `US` for the USA. You want to ensure that the number you purchase is able to receive SMS:

```
vonage numbers:search [COUNTRYCODE]
```

Choose a number from the list and buy it using the following command:

```
vonage numbers:buy [NUMBER] [COUNTRYCODE]
```

You will be prompted to confirm your purchase. Make a note of the number that you bought.

Configure Your Vonage Account

The next thing you need to do is configure your account with the webhook URL so that Vonage can forward any inbound SMS to your application.

This is another task that you could perform either in the dashboard or by using our CLI. Here's how to do it with the CLI. Replace the following placeholders with your own values:

- **`VONAGE_VIRTUAL_NUMBER`**: Your own virtual number, which should include the international dialling code and omit the leading zero, for example: **`447700900001`**.
- **`WEBHOOK_URL`**: Your webhook endpoint with the **`ngrok`** URL as the domain name. For example: **`http://abc123.ngrok.io/webhooks/inbound-sms`**.

```
nexmo link:sms VONAGE_VIRTUAL_NUMBER WEBHOOK_URL
```

This configures your virtual number to use your webhook. If you want to create a "catch all" webhook to capture SMS sent to any of your numbers, visit your account settings page and enter the URL in the "Inbound

messages" textbox under "Default SMS Setting". Vonage always uses the number-specific webhook if it has been set.

Try it Out

With `ngrok` running on port 3000 in one terminal window, launch your Python application in another:

Use our CLI to send a test SMS to your Nexmo number:

```
nexmo sms -f VONAGETEST VONAGE_VIRTUAL_NUMBER "This is a te
```

In the terminal window that is running your Python application you should see that your webhook received the inbound SMS:

```
{'keyword': ['THIS'],
 'message-timestamp': ['2019-05-17 12:49:51'],
 'messageId': ['17000002444B0FD5'],
 'msisdn': ['VONAGETEST'],
 'text': ['This is a test'],
 'to': ['447700900001'],
 'type': ['text']}
127.0.0.1 - - [17/May/2019 13:50:50] "GET /webhooks/inbound
```

Further Reading

If you want to learn more about the SMS API, check out

the following resources:

- SMS API overview
- SMS API reference
- Inbound SMS guide
- Concatenation and encoding