Receiving SMS Delivery Receipts with Python

Last updated on May 13, 2021

You've already learned how to send an SMS with Python. But unless you have access to the device you've sent it to, how can you be sure that your SMS arrived?

Many networks provide Nexmo with delivery receipts. However, not all networks are created equal. Some will just send you an acknowledgement that they have received your message, but no confirmation that it was actually delivered. Others won't send you anything or even generate fake receipts. So be aware that depending on which countries you are sending SMS to, it can be a bit of a minefield. The Nexmo Knowledge Base has more information.

To request a delivery receipt you need to create a publicly-accessible webhook and configure your Nexmo account to use it. This post will show you how to do that and the source code is available on Github.

Prerequisites

Vonage API Account

To complete this tutorial, you will need a Vonage API

account. If you don't have one already, you can sign up today and start building with free credit. Once you have an account, you can find your API Key and API Secret at the top of the Vonage API Dashboard.

This tutorial also uses a virtual phone number. To purchase one, go to *Numbers* > *Buy Numbers* and search for one that meets your needs.

This tutorial uses Python 3 and Flask to code the webhook. You can install `flask` with the pip3 package manager:

Lastly, you'll need to install the Nexmo CLI. You will use this to send a test SMS.

Once you have all those things you are good to go!

Create the Delivery Receipt Webhook

When a network lets Nexmo know that a message has been delivered, Nexmo can forward that information on to your application as an HTTP request. This can be either a `GET` or `POST` request depending on how you have

configured it in your account settings.

However, we're going to play it safe and accept both `GET` and `POST` requests in this example.

Create a file called `receipt.py` and enter the following code:

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

app = Flask(__name__)

@app.route('/webhooks/delivery-receipt', methods=['GET', 'P
def delivery_receipt():
    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/delivery-receipt` 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.

Using this approach, your application can extract the delivery receipt however that request is made.

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 Nexmo.

Make Your Webhooks Accessible

For Nexmo's APIs to make requests to your webhook endpoints they 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. So leave it running for the duration of this tutorial.

Configure Your Nexmo Account

The next thing you need to do is configure your account with the webhook URL so that Nexmo can forward any delivery receipts it gets from carriers.

Visit the settings page in the developer dashboard. Under "Default SMS Setting" put your `ngrok` (e.g. `http://abc123.ngrok.io/webhooks/delivery-receipt`) URL in the "Delivery receipts" textbox and click the "Save changes" button.

Try it Out

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

Use the Nexmo CLI to send a test SMS to your personal mobile phone. Include the international code in the number but remove any leading zeros. For example, a `GB` number might look like this: `447700900001`.

```
nexmo sms -f DLRTEST YOUR_PERSONAL_NUMBER "This is a test m
```

You will receive a message from `DLRTEST`. In the terminal window that is running your Python application you should see a delivery receipt that resembles the following (but might vary by country and network as outlined earlier):

```
{'err-code': ['0'],
 'message-timestamp': ['2019-05-17 10:57:28'],
 'messageId': ['130000002BFABABC'],
 'msisdn': ['447700900001'],
 'network-code': ['23410'],
 'price': ['0.03330000'],
 'scts': ['1905171057'],
 'status': ['accepted'],
 'to': ['DLRTEST2']}
127.0.0.1 - - [17/May/2019 11:57:29] "GET /webhooks/deliver
{'err-code': ['0'],
 'message-timestamp': ['2019-05-17 10:57:30'],
 'messageId': ['130000002BFABABC'],
 'msisdn': ['447700900001'],
 'network-code': ['23410'],
 'price': ['0.03330000'],
 'scts': ['1905171157'],
 'status': ['delivered'],
 'to': ['DLRTEST2']}
127.0.0.1 - - [17/May/2019 11:57:30] "GET /webhooks/deliver
```

As you can see, this example delivery receipt consists of two parts: an acknowledgement that the SMS was received by the carrier (`status: accepted`) and then

delivered to the recipient (`status: delivered`).

That's all there is to it!

Further Reading

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

- SMS API overview
- Delivery receipts guide
- SMS API reference
- Delivery receipt restrictions