**Webex Teams BOT TO GOOGLE DRIVE & SMS**

This quick walkthrough is intended to give you a very simple Webex Teams bot – it sends content from a Webex Teams space to a Google spreadsheet, and then texts your cell phone via the Tropo Scripting API. Nothing too fancy, but easy to build on. The commands will be passed to the bot using mentions – so if the following command is typed into a space where the bot is a participant:

@botname hello

The bot will retrieve that command, process it, post the message to a blank Google spreadsheet and then relay the content to the Tropo API, which will then send a text to your cell phone with an alert.

To start, you’ll need to register your bot – we’ve written fairly extensively on this process, so rather than reinvent the wheel, we recommend reviewing the following blog to get your bot created and ready to go - you’ll want to pause at step 3b as we’ll have different server side code:

<https://developer.webex.com/blog/blog-details-8110.html>

Once your bot is created, we can move on to the next step – the bot code that catches the messages from the webhook and sends it off to Google Spreadsheet:

**The Server Code**

The example webhook in the previous blog sends all newly created messages in which the bot is mentioned to the server on port 10080 (all messages not directed to the bot are filtered out); our server side app will be similar to the one in that blog, but differs in what it does with the messages once received. To start, a simple Python program to handle the messages:

from itty import \*

import urllib2

import json

def sendSparkGET(url):

    request = urllib2.Request(url,

                            headers={"Accept" : "application/json",

                                     "Content-Type":"application/json"})

    request.add\_header("Authorization", "Bearer "+bearer)

    contents = urllib2.urlopen(request).read()

    return contents

@post('/')

def index(request):

    webhook = json.loads(request.body)

    print webhook['data']['id']

    result = sendSparkGET('https://api.ciscospark.com/v1/messages/{0}'.format(webhook['data']['id']))

    result = json.loads(result)

    print result

    return "true"

####CHANGE THIS VALUE#####

bearer = "BOT BEARER TOKEN HERE"

run\_itty(server='wsgiref', host='0.0.0.0', port=10080)

So far, same function as the previous bot - for each message that is created, the data hits your server as JSON, so we parse them with the built-in ‘json’ python library. We have a ‘result’ object that’s JSON parsed and typically contain a ‘text’ field, unless it’s an image posted in the Webex Teams space. We’ll use the ‘text’ attribute of the result object to look for specific messages sent to the bot.

**The** **Google Code:**

Next we’ll create a Python command-line application that makes requests to the Google Sheets API; we effectively follow the steps on this page, all the way to run the sample to confirm Auth works:

<https://developers.google.com/sheets/quickstart/python>

Now we’ll move on to modifying the sample to include our Webex Teams webhook content retrieval code – the first portion is to retrieve the message content from the webhook JSON:

We’ll add this portion before our Python server code. In order to maintain the code integrity, I put the above code(“Python program to handle the Webex Teams messages” part) here again.

from itty import \*

import urllib2, urllib

import json

def sendSparkGET(url):

"""

This method is used for:

-retrieving message text, when the webhook is triggered with a message

-Getting the username of the person who posted the message if a command is recognized

"""

request = urllib2.Request(url,

headers={"Accept" : "application/json",

"Content-Type":"application/json"})

request.add\_header("Authorization", "Bearer "+bearer)

contents = urllib2.urlopen(request).read()

return contents

Now we’ll write the message content to the Google Spreadsheet:

@post('/')

def index(request):

"""

When messages come in from the webhook, they are processed here. The message text needs to be retrieved from Spark, using the sendSparkGET() function. The message text is parsed.

"""

webhook = json.loads(request.body)

#print webhook['data']['id']

result = sendSparkGET('https://api.ciscospark.com/v1/messages/{0}'.format(webhook['data']['id']))

result = json.loads(result)

msg = None

if webhook['data']['personEmail'] != bot\_email:

in\_message = result.get('text', '').lower()

in\_message = in\_message.replace(bot\_name, '')

print (in\_message)

main(in\_message)

return "true"

####CHANGE THESE VALUES#####

bot\_email = "Your Bot Username"

bot\_name = "Your Bot Display Name"

bearer = "Your Bot Bearer Token"

def main(in\_message):

"""

This method is used for:

-retrieving message text, when the webhook is triggered with a message

-Then post the message to a blank Google Spreadsheet to notify a user there came a message from his Webex Teams bot.

-Creates a Sheets API service object:

https://docs.google.com/spreadsheets/d/1ZDu8T6x77KUSh-Hq4fzW9NCgTx0EU\_rWI7lWuPCOEjc/edit

"""

print("This is the message mentioned your bot: {0}".format(in\_message))

#The above print function is used to verify whether the retrieved message content is correct via terminal.

credentials = get\_credentials()

http = credentials.authorize(httplib2.Http())

discoveryUrl = ('https://sheets.googleapis.com/$discovery/rest?'

'version=v4')

service = discovery.build('sheets', 'v4', http=http,

discoveryServiceUrl=discoveryUrl)

spreadsheetId = 'Your Google spreadsheetId'

rangeName = 'A1'

body = {"values":[[in\_message]]}

result = service.spreadsheets().values().append(

spreadsheetId=spreadsheetId, range=rangeName, valueInputOption = 'RAW', insertDataOption = 'INSERT\_ROWS', body = body).execute()

return 'ok'

The above code with the Python server will look like this, which is a fully functional app on its own – if you don’t want to send an SMS, just want to work with a Google Spreadsheet (or other type of Google doc) you can pause here:

from \_\_future\_\_ import print\_function

from itty import \*

import urllib2, urllib

import json

import httplib2

import os

import json

from apiclient import discovery

import oauth2client

from oauth2client import client

from oauth2client import tools

try:

import argparse

flags = argparse.ArgumentParser(parents=[tools.argparser]).parse\_args()

except ImportError:

flags = None

# If modifying these scopes, delete your previously saved credentials

# at ~/.credentials/sheets.googleapis.com-python-quickstart.json

#SCOPES = '<https://www.googleapis.com/auth/spreadsheets.readonly'>

#SCOPES = '<https://www.googleapis.com/auth/drive'>

SCOPES = 'https://www.googleapis.com/auth/spreadsheets'

CLIENT\_SECRET\_FILE = 'client\_secret.json'

APPLICATION\_NAME = 'Google Sheets API Quickstart'

def get\_credentials():

"""Gets valid user credentials from storage.

If nothing has been stored, or if the stored credentials are invalid,

the OAuth2 flow is completed to obtain the new credentials.

Returns:

Credentials, the obtained credential.

"""

home\_dir = os.path.expanduser('~')

credential\_dir = os.path.join(home\_dir, '.credentials')

if not os.path.exists(credential\_dir):

os.makedirs(credential\_dir)

credential\_path = os.path.join(credential\_dir,

'sheets.googleapis.com-python-quickstart.json')

store = oauth2client.file.Storage(credential\_path)

credentials = store.get()

if not credentials or credentials.invalid:

flow = client.flow\_from\_clientsecrets(CLIENT\_SECRET\_FILE, SCOPES)

flow.user\_agent = APPLICATION\_NAME

if flags:

credentials = tools.run\_flow(flow, store, flags)

else: # Needed only for compatibility with Python 2.6

credentials = tools.run(flow, store)

print('Storing credentials to ' + credential\_path)

return credentials

def sendSparkGET(url):

"""

This method is used for:

-retrieving message text, when the webhook is triggered with a message

-Getting the username of the person who posted the message if a command is recognized

"""

request = urllib2.Request(url,

headers={"Accept" : "application/json",

"Content-Type":"application/json"})

request.add\_header("Authorization", "Bearer "+bearer)

contents = urllib2.urlopen(request).read()

return contents

@post('/')

def index(request):

"""

When messages come in from the webhook, they are processed here. The message text needs to be retrieved from Webex Teams, using the sendSparkGET() function. The message text is parsed.

"""

webhook = json.loads(request.body)

#print webhook['data']['id']

result = sendSparkGET('https://api.ciscospark.com/v1/messages/{0}'.format(webhook['data']['id']))

result = json.loads(result)

msg = None

if webhook['data']['personEmail'] != bot\_email:

in\_message = result.get('text', '').lower()

in\_message = in\_message.replace(bot\_name, '')

print (in\_message)

main(in\_message)

return "true"

####CHANGE THESE VALUES#####

bot\_email = " Your Bot Username"

bot\_name = " Your Bot Display Name"

bearer = " Your Bot Bearer Token"

def main(in\_message):

"""

This method is used for:

-retrieving message text, when the webhook is triggered with a message

- Then post the message to a blank Google Spreadsheet to notify a user there came a Webex Teams message from his bot.

-Creates a Sheets API service object:

https://docs.google.com/spreadsheets/d/1ZDu8T6x77KUSh-Hq4fzW9NCgTx0EU\_rWI7lWuPCOEjc/edit

"""

print("This is the message mentioned your bot: {0}".format(in\_message))

#The above print function is used to verify whether the retrieved message content is correct via terminal.

credentials = get\_credentials()

http = credentials.authorize(httplib2.Http())

discoveryUrl = ('https://sheets.googleapis.com/$discovery/rest?'

'version=v4')

service = discovery.build('sheets', 'v4', http=http,

discoveryServiceUrl=discoveryUrl)

spreadsheetId = 'Your Google spreadsheetId'

rangeName = 'A1'

body = {"values":[[in\_message]]}

result = service.spreadsheets().values().append(

spreadsheetId=spreadsheetId, range=rangeName, valueInputOption = 'RAW', insertDataOption = 'INSERT\_ROWS', body = body).execute()

return 'ok'

run\_itty(server='wsgiref', host='0.0.0.0', port=10080)

Don’t forget to fill in your personal content for all the variables:

* bot\_email(Your Bot Username)
* bot\_name(Your Bot Display Name)
* bearer(Your Bot Bearer Token)
* APPLICATION\_NAME(The application name of your Google Developers Console project)
* spreadsheetId(The Webex Teams messages from Bot will be posted to)

If you want to send an SMS alert to go with it, though, we’ll need to create a Tropo application. There are a couple of steps to start with Tropo – first you’ll need to register a Tropo account:

<http://www.tropo.com/register>

This will send an email with a verification link; click the link, and then go log into the Tropo portal. Next, you’ll need to send an email to [support@tropo.com](mailto:support@tropo.com) requesting outbound permissions (they’re disabled by default as fraud protection). Once support team activates your account for outbound permissions, you’ll move on to creating a Tropo Scripting application to send an SMS:

<https://www.tropo.com/docs/scripting/creating-your-first-application>

<https://www.tropo.com/docs/scripting/quickstarts/sending-text-messages>

In specific, we’ll create a Python application with a hardcoded phone number like so:

call("14075551212", {

"network":"SMS"})

say("Spark Message from Bot: " + msg)

Next we’ll add on to our Webex Teams + Google bot code here to send the text as well:

def send\_tropo(spark\_msg):

"""

This method is used for:

- Sending the retrieved message to some one's cell phone via a Tropo scripting SMS application just to notify a Webex Teams message from his Bot was received in the Google Spreadsheet.

"""

url = 'https://api.tropo.com/1.0/sessions'

headers = {'accept':'application/json','content-type':'application/json'}

values = {'token':'messaging token of your Tropo SMS application', 'msg': spark\_msg }

data = json.dumps(values)

req = urllib2.Request(url = url , data = data, headers = headers)

response = urllib2.urlopen(req)

return 'ok'

Enjoy the example - you can also get the complete code on [Github](https://github.com/ciscospark/Spark-API-Demos/blob/master/SparkBotToGoogleWithSMS/SparkBotToGoogleWithSMS.py)! [Let us know](https://developer.ciscospark.com/support.html) if you have any questions. (I’m trying to upload the complete code to github and add the hyper link here.)

Don’t forget to fill in your personal content for all the variables:

* bot\_email(Your Bot Username)
* bot\_name(Your Bot Display Name)
* bearer(Your Bot Bearer Token)
* APPLICATION\_NAME(The application name of your Google Developers Console project)
* spreadsheetId(The Webex Teams messages from Bot will be posted to)
* token(messaging token of your Tropo SMS application)

and your own cell phone number in the Tropo Application.   
  
To run the server application, open a command terminal, and navigate to the folder where you saved this Python script - we named ours

WebexTeams BotToGoogleWithSMS.py - then run:

python WebexTeams BotToGoogleWithSMS.py

As long as all of the right ports are available, if someone posts a message like “hello” and “world” with my bot mentioned, we will see this in the terminal:

itty starting up (using wsgiref)...

Listening on http://0.0.0.0:10080...

Use Ctrl-C to quit.

hello

This is the message mentioned your bot: hello

127.0.0.1 - - [31/Aug/2016 06:16:04] "POST / HTTP/1.1" 200 4

world

This is the message mentioned your bot: world

127.0.0.1 - - [31/Aug/2016 06:16:23] "POST / HTTP/1.1" 200 4

Also, we can see the two strings “hello” and “world” are inserted in A1 sheet of my spreadsheet and my cell phone received two messages stating “Spark Message mentioned your bot is:hello!” and “Spark Message mentioned your bot is:world!” Please check the below screenshots:



