ENZO

Get things done.

LAB: Simple Security System with Raspberry Pi, Enzo Online and Azure Cognitive Services

Create a simple security monitoring system using a Raspberry Pi, Azure Cognitive Services and Enzo Online.

Created by: Herve Roggero, Azure MVP

Released On: 4/14/2018

info@enzounified.com

The purpose of this lab is to create a simple security monitoring system that uses a few key services to assist with image analysis and log storage for auditing purposes. This project leverages small IoT devices that can easily be placed in sensitive areas to monitor activity. This is a conceptual project and should be considered a simple prototype to demonstrate the use of key cloud technologies. This lab is divided in the following sections: Configuration, Image Analysis, Auditing, and Alerting.

This Lab uses the following technologies:

- Raspberry Pi 3 or higher
- Python 3
- Camera
- Azure Cognitive Services
- Enzo Online
- Twilio
- Azure Storage
- PuTTY (to copy/paste code into the Raspberry Pi: https://www.putty.org/)

Pre-Requisites

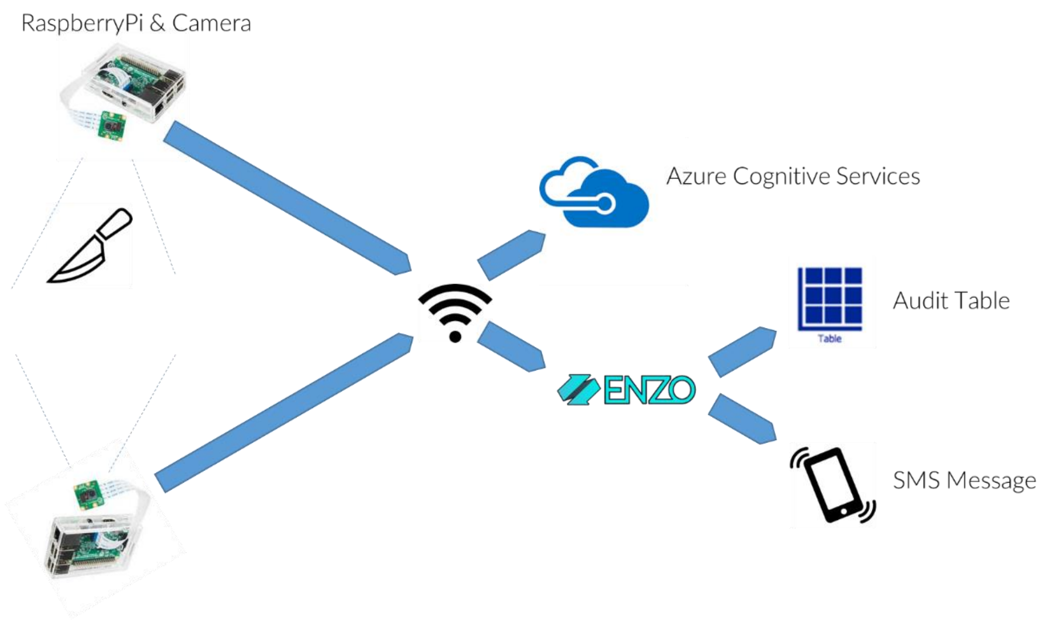
Before starting this lab you should have the following:

- Hardware
  - A Raspberry Pi with the Raspbian operating system installed and configured
  - Python 3 installed (should come with Raspbian)
  - A Raspberry Pi camera (Kuman for Raspberry Pi)
    - https://goo.gl/jDFhFq
  - XRDP to use Remote Desktop Services if you are using a Windows machine
    - Use this command to install XRDP
      sudo apt-get install xrdp
  - A USB keyboard to control the Raspberry Pi
  - A USB monitor to view the Raspberry Pi screen
  - A WiFi or wired network for the Raspberry Pi to connect to the Internet
- Services
  - A Microsoft Azure account (https://portal.azure.com)
    - You can create a free account
  - An Enzo Online account (https://portal.enzounified.com)
    - You can create a free account
  - A Twilio account to send text messages (https://www.twilio.com)
    - You can create a free account

Overview

The lab uses a camera attached to a Raspberry Pi device to take pictures frequently, and analyze each picture using the Microsoft Azure Cognitive Service. If the picture contains a person, and/or a dangerous item, such as a knife, the code running on the Raspberry Pi will log the information in an Azure Table data store through Enzo Online. The device will also send an SMS text message using Twilio services through Enzo Online. Using Enzo Online removes the need for complex SDKs and makes developing against complex cloud resources fast and easy.

While this lab only requires a single Raspberry Pi, it will work with multiple devices without modification as shown in the diagram below.



Configuration

Configure the Camera

The first step in this lab is to configure the camera. You will need to follow the instructions found here to install the module and enable the camera:

https://www.raspberrypi.org/documentation/configuration/camera.md

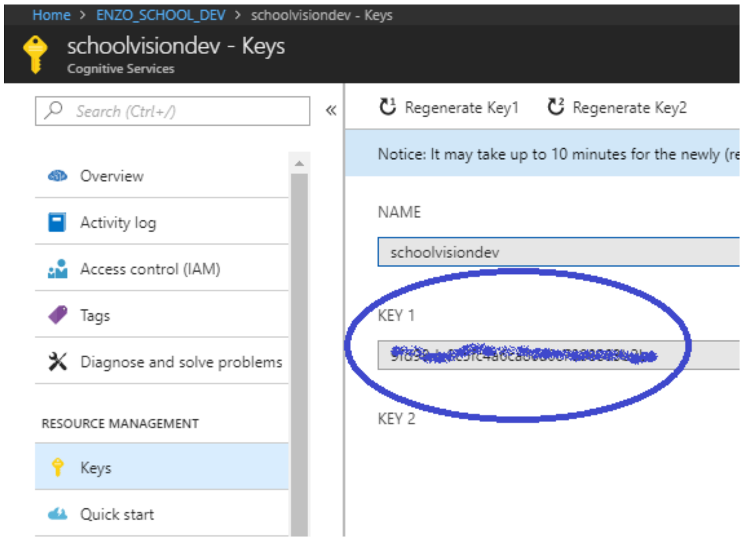
Create a Pictures directory on the Desktop

From your Raspberry Pi Desktop, create a Folder called **Pictures**. Unix is case-sensitive, so make sure to respect the upper-case 'P' in the folder name.

Image Analysis

Azure Cognitive Service

You will need to create a Cognitive service in your Azure subscription so you can analyze images. Please see instructions on the Microsoft Azure website to create the Cognitive service in Azure (https://docs.microsoft.com/en-us/azure/cognitive-services/cognitive-services-apis-create-account). Once created, you will need to note one of security Keys provided.



NAME
schoolvisiondev

KEY 1
[redacted]

KEY 2

Code

Let's create the first part of the code that communicates with the Azure Cognitive Service to analyze a picture that the camera will take. This code takes a picture in the direction the camera is facing, and saves it to a local file so that the image can be inspected for troubleshooting purposes. The camera should be angled in a way that the image is not upside down, as this could affect the quality of the analysis.

Create a new file (ex: securitysystem.py) on your Raspberry Pi desktop and double-click to edit it. The Python editor will start automatically. Type the following code in your file; it imports the necessary libraries and declares the variables needed for this lab. Calling the Azure Cognitive Service is very simple from Python as it only requires the use of an HTTP Post; no specific libraries or SDKs are needed.

The interesting part of this code is the call to the Azure Cognitive service, which is made by issuing a POST request: *conn.request("POST", analyze_uri % params, body, headers).* Once the call is made, the code inspects a JSON response looking for a tags array: *tags = (parsed['description']['tags'])*. If a tag contains the word 'knife' the method returns true, indicating that a warning should be issued.

Note: Python is sensitive to leading spaces/tabs. Make sure the indentation is exactly as shown in the code below.

Note: Information highlighted in blue means that it may need to be updated for the code to work.

```
import http.client, urllib.request, urllib.parse, urllib.error, base64, json, uuid
from picamera import PiCamera
from time import sleep
import time
import requests
import sys

#cognitive settings
subscription_key = "YOUR_COGNITIVE_SERVICE_KEY"
uri_base = 'southcentralus.api.cognitive.microsoft.com'
analyze_uri = "/vision/v1.0/analyze?%s"

#other settings
fileName = "/home/pi/Desktop/Pictures/image.jpg"
headers = dict()
headers['Ocp-Apim-Subscription-Key'] = subscription_key
headers['Content-Type'] = "application/octet-stream"

camera = PiCamera()

def capturePicture(fipathToFileInDiskle):
        camera.capture(fipathToFileInDiskle)

def isDetected(pathToFileInDisk, headers):
        params= urllib.parse.urlencode({
                "visualFeatures": "Categories,Description",
                "launguage": "en",
        })

        with open( pathToFileInDisk, "rb" ) as f:
                inputdata = f.read()
        body = inputdata
        iswarning = False

        try:
                conn = http.client.HTTPSConnection(uri_base)
                conn.request("POST", analyze_uri % params, body, headers)
                response = conn.getresponse()
                data = response.read().decode('utf-8')
                #print(data)
                parsed = json.loads(data)
                tags = (parsed['description']['tags'])
                print ("response:")

                for c in tags:
                        #print(c)
                        if c == 'knife':
```

```
                        iswarning = True
                        break

                if iswarning:
                    print('KNIFE DETECTED')

                conn.close()

            except Exception as e:
                print('Error:')
                print(e)

            return iswarning



        # Main code starts here
        print('starting...')

        while True:
            try:
                print("calling camera...")
                capturePicture(fileName)
            except:
                print("Error:", sys.exc_info()[0])

            sleep(5)
```

Auditing

In the previous section the code calls the Microsoft Azure service to analyze a picture taken from the camera, and determines whether a specific tag has been detected by the service (such as a knife). In this section we will save status information to an Azure Table which will be used for auditing purposes.

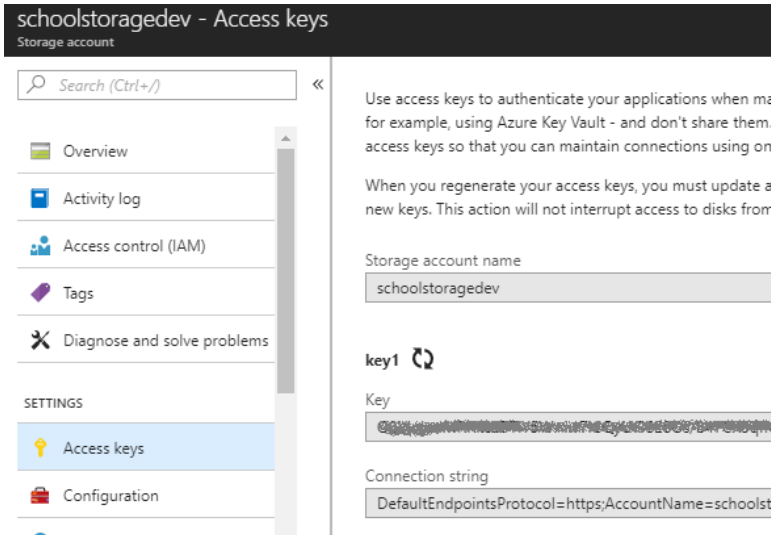
Every time the status changes (knife detected or not), a new record will be added. To achieve this we are using the Enzo Online service which requires its own access key (called the AuthToken). Similarly to the Azure Cognitive service, you call the Enzo service using simple HTTP commands by setting headers and optionally an HTTP body. The sections in yellow below represent new code from the previous version, and the blue highlights are the security tokens.

Before you can call Enzo Online, you must configure an Azure Storage service in Azure, and create a configuration setting in Enzo Online that points to the Azure Storage.

Configuration

Azure Storage

First you need to create an Azure Storage account in your Azure account. To create a storage account, follow the instructions provided by Microsoft: https://docs.microsoft.com/en-us/azure/storage/common/storage-create-storage-account. Note that you will need to obtain your storage access keys to configure Enzo Online; they are found in the Access Keys tab of your storage account.



schoolstoragedev - Access keys
Storage account

Use access keys to authenticate your applications when ma
for example, using Azure Key Vault - and don't share them.
access keys so that you can maintain connections using on

When you regenerate your access keys, you must update a
new keys. This action will not interrupt access to disks from

Storage account name
schoolstoragedev

key1

Key
[redacted]

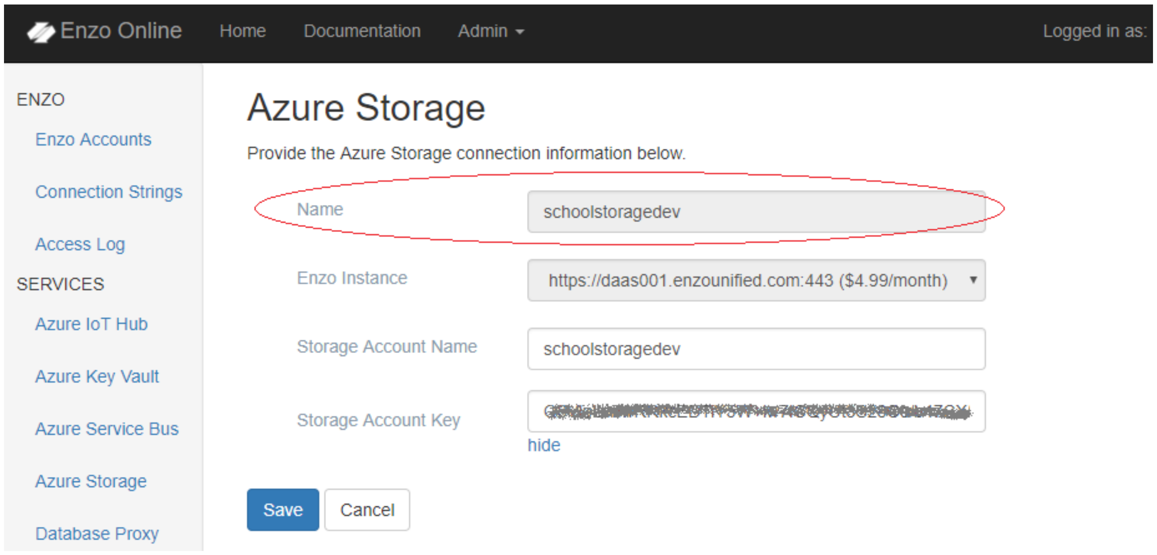
Connection string
DefaultEndpointsProtocol=https;AccountName=schoolst

Enzo Online

Once you created the Azure Storage account, go to the Enzo portal (https://portal.enzounified.com) and create a new Azure Storage service with the following values:

- Name: schoolstoragedev
- Enzo Instance: <leave the default value>
- Storage Account Name: The storage account name you created in Azure
- Storage Account Key: The primary or secondary storage account key in Azure

Once created, the configuration setting should look like this:



Azure Storage

Provide the Azure Storage connection information below.

Name	schoolstoragedev
Enzo Instance	https://daas001.enzounified.com:443 ($4.99/month)
Storage Account Name	schoolstoragedev
Storage Account Key	[redacted]

hide

Code

Once configuration is complete, you can now call Enzo Online from your code. Calling Enzo Online is similar to the call made against the Azure Cognitive service; since no SDK is required it is as simple as making an HTTP request to Enzo. The code will automatically attempt to create an Azure Table if it isn't found at startup by calling the createTable method. Then periodically, as a knife is detected or not, the code will save an entry in the Azure Table saying so to keep an audit.

The saveResult method formats an XML document to be sent to the SaveEntity enzo method; a JSON payload could also be sent instead of XML. The code adds the necessary headers (authToken, _config and tableName), and adds a form post body (data) containing the XML document. The complete source code is here:

Note: Code highlighted in yellow is new code from the previous example.

```
import http.client, urllib.request, urllib.parse, urllib.error, base64, json, uuid
from picamera import PiCamera
from time import sleep
import time
import requests
import sys

#device settings
locationname = 'location001'   #will be used as the Azure table name
deviceid = 'iotdevice-001'      #will be used as the RowKey


#enzo settings
```

```
tableconfig = 'schoolstoragedev'
enzourl="https://daas001.enzounified.com" #saving cloud data through enzo
enzourl_save= enzourl + '/bsc/azurestorage/saveentity' #saving cloud data through enzo
enzourl_tablecreate= enzourl + '/bsc/azurestorage/createtable' #create azure table
enzoguid="YOUR_ENZO_AUTHTOKEN"
xmldata="<root><PartitionKey>{}</PartitionKey><RowKey>{}</RowKey><warning
type='int'>{}</warning></root>"

#cognitive settings
subscription_key = "YOUR_COGNITIVE_SERVICE_KEY"
uri_base = 'southcentralus.api.cognitive.microsoft.com'
analyze_uri = "/vision/v1.0/analyze?%s"

#other settings
fileName = "/home/pi/Desktop/Pictures/image.jpg"
headers = dict()
headers['Ocp-Apim-Subscription-Key'] = subscription_key
headers['Content-Type'] = "application/octet-stream"

lastValue = False
camera = PiCamera()

def capturePicture(fipathToFileInDiskle):
	camera.capture(fipathToFileInDiskle)

def isDetected(pathToFileInDisk, headers):
	params= urllib.parse.urlencode({
		"visualFeatures": "Categories,Description",
		"launguage": "en",
	})

	with open( pathToFileInDisk, "rb" ) as f:
		inputdata = f.read()
	body = inputdata
	iswarning = False

	try:
		conn = http.client.HTTPSConnection(uri_base)
		conn.request("POST", analyze_uri % params, body, headers)
		response = conn.getresponse()
		data = response.read().decode('utf-8')
		#print(data)
		parsed = json.loads(data)
		tags = (parsed['description']['tags'])
		print ("response:")
```

```
        for c in tags:
            #print(c)
            if c == 'knife':
                iswarning = True
                break

        if iswarning:
            print('KNIFE DETECTED')

        conn.close()

    except Exception as e:
        print('Error:')
        print(e)

    return iswarning

def createTable(name):

    iotheaders={'authToken':enzoguid,
          '_config': tableconfig,
          'tablename':name}
    try:
        response=requests.post(enzourl_tablecreate, headers=iotheaders)
        print(response)
    except:
        print("Error:", sys.exc_info()[0])

def saveResult(iswarning):
    valuetosend = 0
    if iswarning:
        valuetosend = 1

    rowKey = uuid.uuid4()
    xmltosend = xmldata.format(deviceid, rowKey, valuetosend)
    print(xmltosend)

    iotheaders={'authToken':enzoguid,
          '_config': tableconfig,
          'tablename':locationname}
    formdata = {'data':xmltosend}

    try:
        response=requests.post(enzourl_save, headers=iotheaders, data=formdata)
    except:
        print("Error:", sys.exc_info()[0])

# Main code starts here
print('starting...')
createTable(locationname)       #create the table every time - will only work the first time

print('creating initial value')
saveResult(False)

while True:
    try:
        print("calling camera...")
        capturePicture(fileName)
        print("calling service...")
        iswarning = isDetected(fileName, headers)
        if lastValue != iswarning:
            print("calling Enzo...")
            saveResult(iswarning)
            lastValue = iswarning
    except:
        print("Error:", sys.exc_info()[0])

    sleep(5)
```

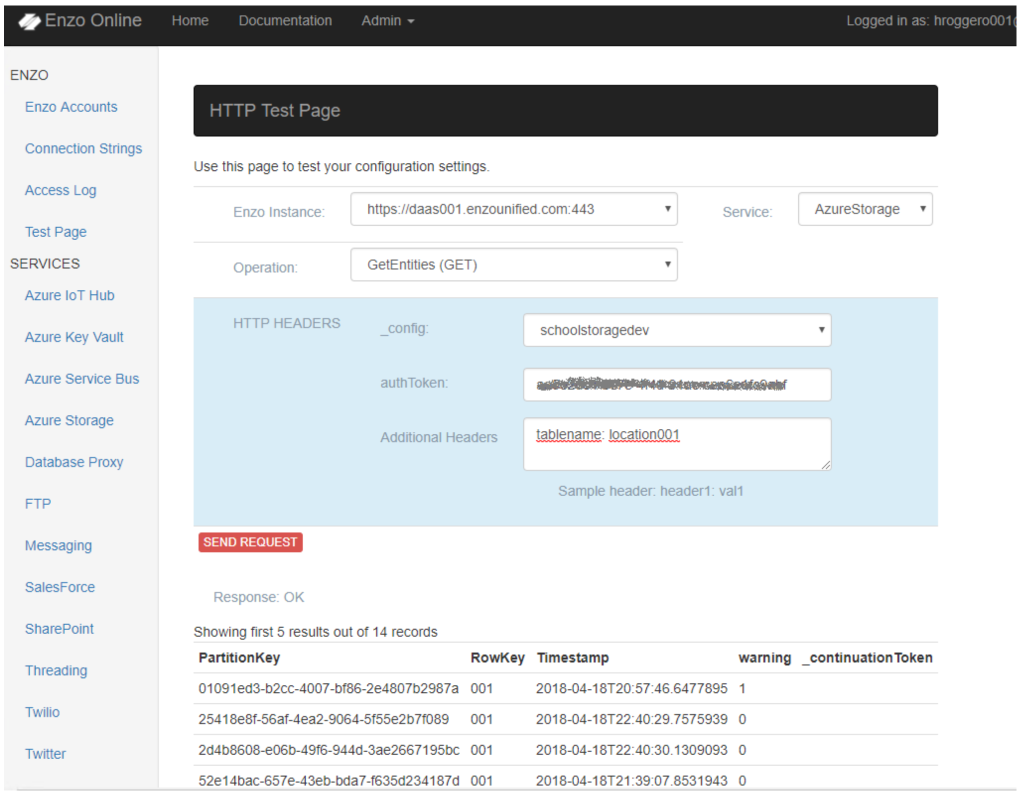
Viewing Auditing Information

To verify that you are indeed saving data in your Azure Table, you can use the Test Page provided in the Enzo Portal:

- From the Enzo Online portal click on Test Page from the left menu
- Choose the Azure Storage service
- Select the GetEntities operation
- In the Additional Headers section, type the following:
  tablename: location001

- Click on Send Request

You should see an output similar to this (only a few records are displayed by default):



HTTP Test Page

Use this page to test your configuration settings.

Enzo Instance: https://daas001.enzounified.com:443

Service: AzureStorage

Operation: GetEntities (GET)

HTTP HEADERS

_config: schoolstoragedev

authToken: [redacted]

Additional Headers: tablename: location001

Sample header: header1: val1

Response: OK

Showing first 5 results out of 14 records

PartitionKey	RowKey	Timestamp	warning	_continuationToken
01091ed3-b2cc-4007-bf86-2e4807b2987a	001	2018-04-18T20:57:46.6477895	1	
25418e8f-56af-4ea2-9064-5f55e2b7f089	001	2018-04-18T22:40:29.7575939	0	
2d4b8608-e06b-49f6-944d-3ae2667195bc	001	2018-04-18T22:40:30.1309093	0	
52e14bac-657e-43eb-bda7-f635d234187d	001	2018-04-18T21:39:07.8531943	0	

Alerting

Now that you can capture images and save audit information, let's configure the system to send an alert using the Twilio services. The simplest way to use Twilio is to send a SMS message (note that you could also use the Messaging service to do so, however Twilio is a more robust approach).

Signup for Twilio

To signup for Twilio, visit the Twilio website and sign up (https://www.twilio.com/try-twilio). Once you have signed up, you will need to retrieve your Account SID and Twilio AuthToken from the Dashboard. You will use this information later.

You will also need a Twilio phone number; in order to obtain a Twilio phone number, go to the Learn &* Build / Build tab on the Twilio portal, and click on **Get a number**. As long as you have a trial account, you can obtain a free Twilio phone number to send SMS text messages from.

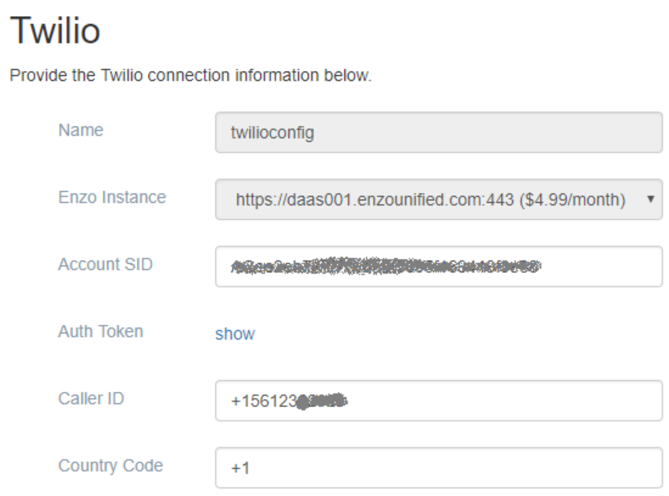
Twilio Configuration in Enzo Online

In the Enzo Online portal, select the Twilio service and create a new configuration with the following information:

- Name: twilioconfig
- AccountSID: YOUR_TWILIO_SID
- Auth Token: YOUR_TWILIO_AUTH_TOKEN

- Caller Id: TWILIO_PHONE_NUMBER (add +1 in front of the area code for a US phone number; use your country code if not in the US)
- Country Code: +1 [note: this is your country code]

Your Twilio configuration should look like this:


Twilio

Provide the Twilio connection information below.

Name	twilioconfig
Enzo Instance	https://daas001.enzounified.com:443 ($4.99/month)
Account SID	[redacted]
Auth Token	show
Caller ID	+156123[redacted]
Country Code	+1

Code

We are ready to send a message from the Raspberry Pi when a knife is detected; the code will only send a message if there was no knife detected previously to avoid sending too many messages.

The approach to send a Twilio message is conceptually the same than when we saved audit data to the Azure Table. We simply call Enzo's Twilio endpoint (bsc/twilio/sendsms) using the correct headers (see the online help documentation for a list of headers: https://portal.enzounified.com/Docs/Documentation.html#docTwilio).

The relevant section of the code looks like this, where the phoneNumbers variable is the phone number that will receive the SMS message:

```
        iotheaders={'authToken':enzoguid,
               '_config': tableconfig,
               'phones': phoneNumbers}
        formdata = {'message': 'Warning: a knife was detected!'}

        try:
            response=requests.post(enzourl_sms, headers=iotheaders, data=formdata)

    except:
        print("Error:", sys.exc_info()[0])
```

The complete source code is below:

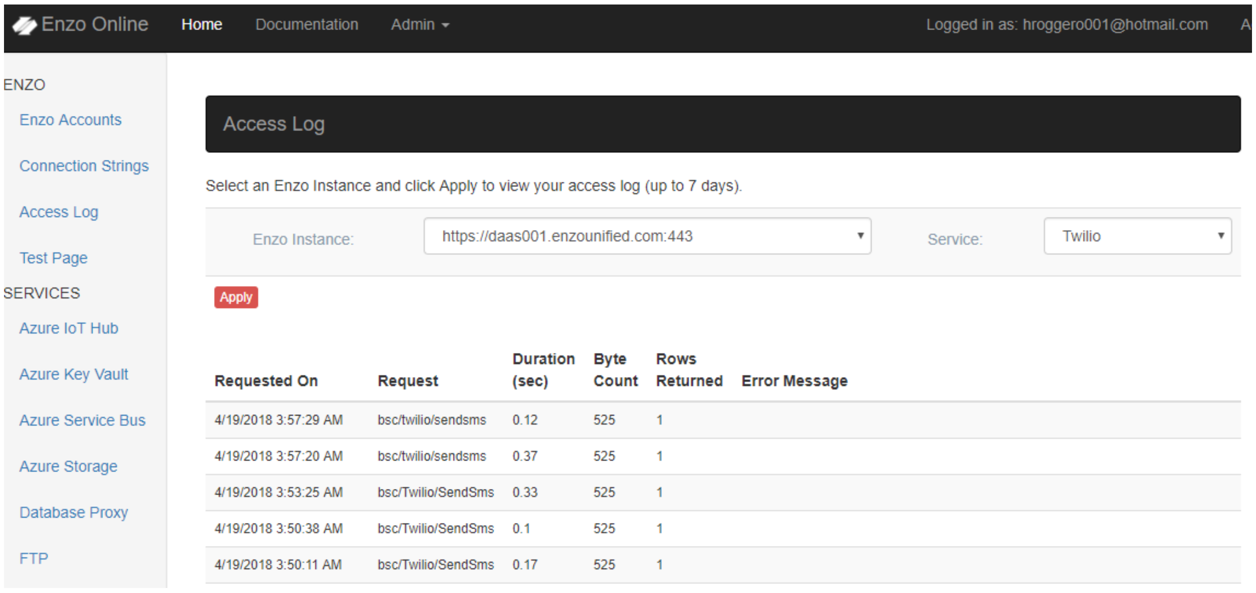
```
import http.client, urllib.request, urllib.parse, urllib.error, base64, json, uuid
```

```
from picamera import PiCamera
from time import sleep
import time
import requests
import sys

#device settings
locationname = 'location001'   #will be used as the Azure table name
deviceid = 'iotdevice-001'      #will be used as the RowKey

#enzo settings
tableconfig = 'schoolstoragedev'
enzourl="https://daas001.enzounified.com" #saving cloud data through enzo
enzourl_save= enzourl + '/bsc/azurestorage/saveentity' #saving cloud data through enzo
enzourl_tablecreate= enzourl + '/bsc/azurestorage/createtable' #create azure table
enzoguid="YOUR_ENZO_AUTHTOKEN"
xmldata="<root><PartitionKey>{}</PartitionKey><RowKey>{}</RowKey><warning
type='int'>{}</warning></root>"
enzourl_sms = enzourl + '/bsc/twilio/sendsms'
phoneNumbers = 'ENTER_YOUR_CELL_PHONE'
twilioconfig = 'twilioconfig'

#cognitive settings
subscription_key = "YOUR_COGNITIVE_SERVICE_KEY"
uri_base = 'southcentralus.api.cognitive.microsoft.com'
analyze_uri = "/vision/v1.0/analyze?%s"

#other settings
fileName = "/home/pi/Desktop/Pictures/image.jpg"
headers = dict()
headers['Ocp-Apim-Subscription-Key'] = subscription_key
headers['Content-Type'] = "application/octet-stream"

lastValue = False
camera = PiCamera()

def capturePicture(fipathToFileInDiskle):
    camera.capture(fipathToFileInDiskle)

def isDetected(pathToFileInDisk, headers):
    params= urllib.parse.urlencode({
        "visualFeatures": "Categories,Description",
        "launguage": "en",
    })
    with open( pathToFileInDisk, "rb" ) as f:
        inputdata = f.read()
    body = inputdata
    iswarning = False

    try:
        conn = http.client.HTTPSConnection(uri_base)
        conn.request("POST", analyze_uri % params, body, headers)
        response = conn.getresponse()
        data = response.read().decode('utf-8')
        #print(data)
        parsed = json.loads(data)
        tags = (parsed['description']['tags'])
        print ("response:")

        for c in tags:
            #print(c)
            if c == 'knife':
                iswarning = True
                break

        if iswarning:
            print('KNIFE DETECTED')

        conn.close()

    except Exception as e:
        print('Error:')
        print(e)

    return iswarning

def createTable(name):

    iotheaders={'authToken':enzoguid,
            '_config': tableconfig,
            'tablename':name}
    try:
        response=requests.post(enzourl_tablecreate, headers=iotheaders)
        print(response)
    except:
        print("Error:", sys.exc_info()[0])

def saveResult(iswarning):
    valuetosend = 0
    if iswarning:
        valuetosend = 1

    rowKey = uuid.uuid4()
    xmltosend = xmldata.format(deviceid, rowKey, valuetosend)
    print(xmltosend)

    iotheaders={'authToken':enzoguid,
            '_config': tableconfig,
            'tablename':locationname}
    formdata = {'data':xmltosend}

    try:
        response=requests.post(enzourl_save, headers=iotheaders, data=formdata)
    except:
        print("Error:", sys.exc_info()[0])

def sendSMS():

    iotheaders={'authToken':enzoguid,
            '_config': twilioconfig,
            'phones': phoneNumbers}
    formdata = {'message': 'Warning: a knife was detected!'}

    try:
        response=requests.post(enzourl_sms, headers=iotheaders, data=formdata)
    except:
        print("Error:", sys.exc_info()[0])

# Main code starts here
print('starting...')
createTable(locationname)    #create the table every time - will only work the first time

print('creating initial value')
saveResult(False)

while True:
    try:
        print("calling camera...")
        capturePicture(fileName)
        print("calling service...")
        iswarning = isDetected(fileName, headers)
        if lastValue != iswarning:
            print("calling Enzo...")
            saveResult(iswarning)
```

```
                sendSMS()
                lastValue = iswarning
    except:
        print("Error:", sys.exc_info()[0])

    sleep(5)
```

Monitoring Activity Through Enzo Online Portal

Last but not least, let's use the Enzo Online portal to see the activity generated by the Raspberry Pi. To view the calls made to Enzo Online, use the Access Log tab. Select the Twilio service for example and click on Apply. You should see an output similar to the picture below. Choosing Azure Storage would give you the calls made to the Azure Storage table.



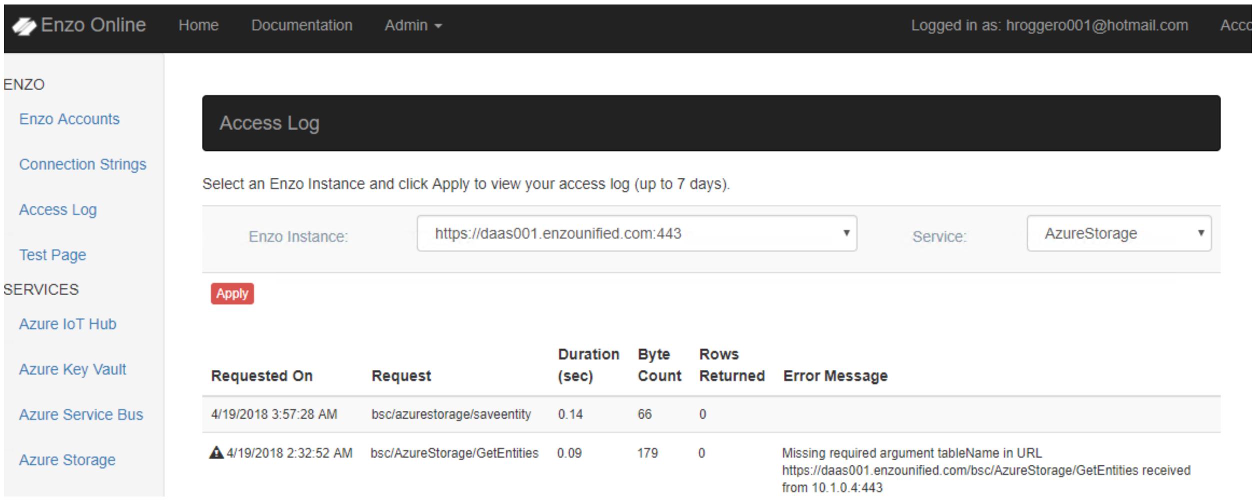
Select an Enzo Instance and click Apply to view your access log (up to 7 days).

Enzo Instance: https://daas001.enzounified.com:443

Service: Twilio

Requested On	Request	Duration (sec)	Byte Count	Rows Returned	Error Message
4/19/2018 3:57:29 AM	bsc/twilio/sendsms	0.12	525	1	
4/19/2018 3:57:20 AM	bsc/twilio/sendsms	0.37	525	1	
4/19/2018 3:53:25 AM	bsc/Twilio/SendSms	0.33	525	1	
4/19/2018 3:50:38 AM	bsc/Twilio/SendSms	0.1	525	1	
4/19/2018 3:50:11 AM	bsc/Twilio/SendSms	0.17	525	1	

This log will also return most errors that occurred when calling Enzo Online. For example, the call to **GetEntities** to the Azure Storage service failed because the **tableName** header was missing.



Select an Enzo Instance and click Apply to view your access log (up to 7 days).

Enzo Instance: https://daas001.enzounified.com:443

Service: AzureStorage

Apply

Requested On	Request	Duration (sec)	Byte Count	Rows Returned	Error Message
4/19/2018 3:57:28 AM	bsc/azurestorage/saveentity	0.14	66	0	
⚠ 4/19/2018 2:32:52 AM	bsc/AzureStorage/GetEntities	0.09	179	0	Missing required argument tableName in URL https://daas001.enzounified.com/bsc/AzureStorage/GetEntities received from 10.1.0.4:443