Proactive events – Raspberry pi Alexa notifications

Objective:

Get raspberry Pi to send notification to Alexa device in python

e.g Weather station, or Intruder alert

Two parts:

1. Alexa notifications
2. Raspberry pi code - python

Part 1

Some references:

https://developer.amazon.com/en-US/docs/alexa/smapi/proactive-events-api.html

https://developer.amazon.com/en-US/docs/alexa/smapi/schemas-for-proactive-events.html

"you must develop your skill using ASK CLI (Alexa Skills Kit Command Line Interface), because some features supported in SMAPI are not supported on the Amazon Developer Portal."

AMAZON: https://developer.amazon.com/blogs/alexa/post/7822f3ee-1735-4eaa-9aa6-5b8e39953c07/proactiveeventsapi-launch-announcement

See also Dabble labs video and github:

https://www.youtube.com/watch?v=oMcHTMZDTVQ

https://github.com/dabblelab/14-alexa-proactive-events-example-skill

Setting up the CLI Setting up the ASK CLI: https://youtu.be/lcJ_K7dTjW0

Amazon determine the types of notification:

Event description	Example notification	Event name
Weather alert	Tornado alert	AMAZON.WeatherAlert.Activated
Sports event update	England 5, Wales 1	AMAZON.SportsEvent.Updated
Message reminder	3 new messages.	AMAZON.MessageAlert.Activated
Order status update	Order has been sent	AMAZON.OrderStatus.Updated
Reservation confirmation	Dentist confirmed.	AMAZON.Occasion.Updated
Trash collection reminder	Tuesday is blue binday.	AMAZON.TrashCollectionAlert.Activated
Media content notification	Band plays on Sunday	AMAZON.MediaContent.Available
Game invitation notification	Invitation for cards.	AMAZON.SocialGameInvite.Available

We'll use the notification AMAZON.MessageAlert.Activated

User notification: "You have <messageGroup.count> <state.freshness> <state.status> message/messages from <messageGroup.creator.name>."

3 `NEW`, `OVERDUE, UNREAD`, `FLAGGED` messages from John

Lets start. I use Visual Studio coder. Using ASK CLI

C:\Users\john\Documents\Alexa> ***ask new***

Please follow the wizard to start your Alexa skill project ->
? Choose the programming language you will use to code your skill: ***Python***
? Choose a method to host your skill's backend resources: ***AWS Lambda***
Host your skill code on AWS Lambda (requires AWS account).
? Choose a template to start with: ***Hello world***
Alexa's hello world skill to send the greetings to the world!
? Please type in your skill name: ***intruder alert***
? Please type in your folder name for the skill project (alphanumeric): ***intruderalert***
Project for skill "intruder alert" is successfully created at ..Documents\Alexa\ intruderalert

Project initialized with deploy delegate "@ask-cli/lambda-deployer" successfully.

Change to the intruderalert folder and run **ask deploy**

You'll get a successfully deployed message.

If you want to now, you can find the skill at developer.amazon.com.

Open up intruder alert in VS code
In skill package > interaction models, change the en-US.json code invocation:

```
"invocationName": "intruder alert",
```

My Alexa devices are registered in the UK, so I add en-GB.json (I copy the en-US one)

Add the new locale to the skill.json as follows:

```
"en-US": {
  "summary": "Sample Short Description",
  "examplePhrases": [
    "Alexa open hello world",
    "hello",
    "help"
  ],
  "name": "intruder alert",
  "description": "Sample Full Description"
},
"en-GB": {
  "summary": "UK Short Description",
  "examplePhrases": [
    "Alexa open hello world",
    "hello",
    "help"
  ],
  "name": "intruder alert",
  "description": "Sample Full Description"
}
},
```

Open skill.json and copy the uri

```
"apis": {
  "custom": {
    "endpoint": {
```

```
          "uri": "arn:aws:lambda:us-east-1:5591444306262:function:ask-
sendnotification-default-default-1614362517059"
        }
      }
    },
```

Still in the skill manifest (skill.json) , under the apis code:

```
"apis": {
    …
},
```

Add the following:

```
"permissions": [
  {
    "name": "alexa::devices:all:notifications:write"
  }
 ],
 "events": {
  "publications": [
   {
    "eventName": "AMAZON.MessageAlert.Activated"
   }
  ],
  "endpoint": {
   "uri": "arn:aws:lambda:us-east-1:0000000000000:function:sampleSkill"
  },
  "subscriptions": [
   {
    "eventName": "SKILL_PROACTIVE_SUBSCRIPTION_CHANGED"
   }
  ],
  "regions": {
   "NA": {
    "endpoint": {
     "uri": "arn:aws:lambda:us-east-1:0000000000000:function:sampleSkill"
    }
   }
  }
 },
```

Replacing the ***arn:aws:lambda:us-east-1:0000000000000:function:sampleSkill*** with the one you copied.

Don’t forget the ending comma

Edit helloworld.py to add the proactive event handler:
Add the following line of code in the sb = SkillBuilder() section

```
sb.add_request_handler(LaunchRequestHandler())
sb.add_request_handler(HelloWorldIntentHandler())
sb.add_request_handler(ProactiveEventHandler())
sb.add_request_handler(HelpIntentHandler())
```

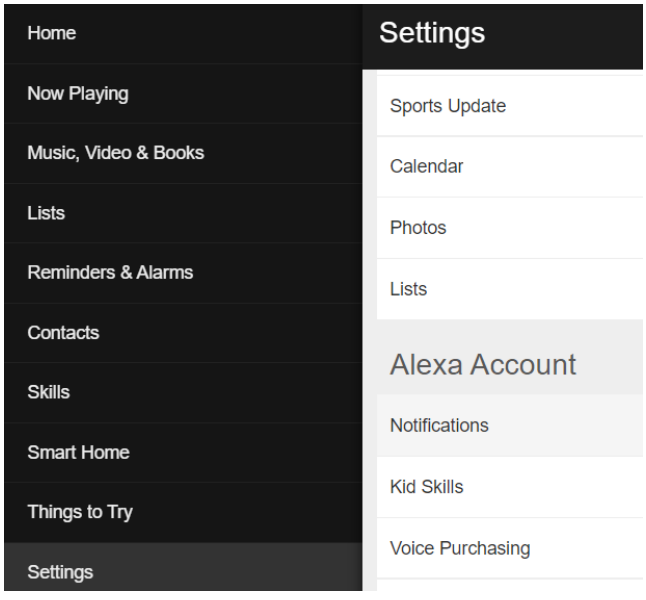
and add the proactive event handler (say after the HelloWorldIntent handler):

```
class ProactiveEventHandler(AbstractRequestHandler):
    """Handler for Hello World Intent."""
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return
ask_utils.is_request_type("AlexaSkillEvent.ProactiveSubscriptionChanged")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        logger.info("Proactive event changed userId, api endpoint and subscription")
        logger.info(ask_utils.request_util.get_user_id(handler_input))
        logger.info(handler_input.request_envelope.context.system.api_endpoint)
        logger.info(handler_input.request_envelope.request.body.subscriptions)

        return (
            handler_input.response_builder
                .response
        )
```

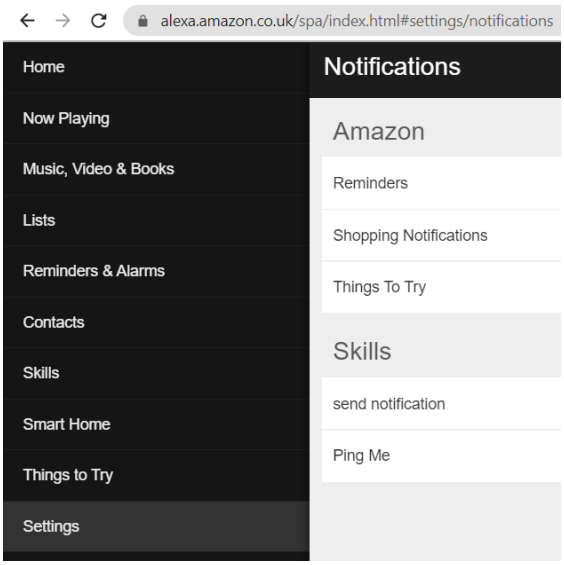
And run **ask deploy** again

Now we will enable notifications in our Alexa Skill. Your end user will do this via their aAlexa skill.

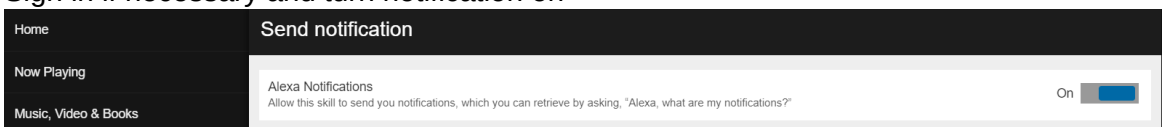
Go to Alexa.amazon.com
Settings > Alexa Account > notifications



If it doesn’t appear, check your locale.



Sign in if necessary and turn notification on



This triggers the ProactiveEventHandler with AlexaSkillEvent.ProactiveSubscriptionChanged and 'AMAZON.MessageAlert.Activated' event.

If you open cloudwatch and look at the logs, you'll see

```
79 Proactive event changed userId, api endpoint and subscription
79 amzn1.ask.account.AHIBPIPN2CEHUC3HEFBN4MNRLSPRPNCZMZ2CVXV6GII…
79 https://api.eu.amazonalexa.com
79 [{'event_name': 'AMAZON.MessageAlert.Activated'}]
```

And under subscriptions,

'event_name': 'AMAZON.MessageAlert.Activated'

The other logs give us the endpoint and user information:
Endpoint: https://api.eu.amazonalexa.com
User: amzn1.ask.account.AHTK5FZOL6TPHV4WHFNB3BJDRX5TI53MJ …

You can now use that information to keep a record of the user's consent to notifications and to send messages to individuals. We'll just send a multicast message later, so won't save the user ID

When the user (or you) turns off the subscription, you get the (error) message:

logger.info(handler_input.request_envelope.request.body.subscriptions)
AttributeError: 'NoneType' object has no attribute 'subscriptions'

You will need to check for that in your code if you are going to add and remove users from a database

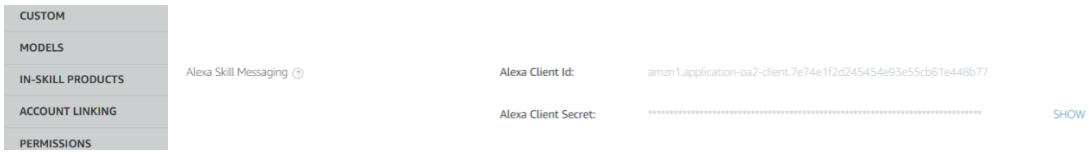
Part 2 Now we need to send a notification

This is done by sending POST requests to the Amazon servers.

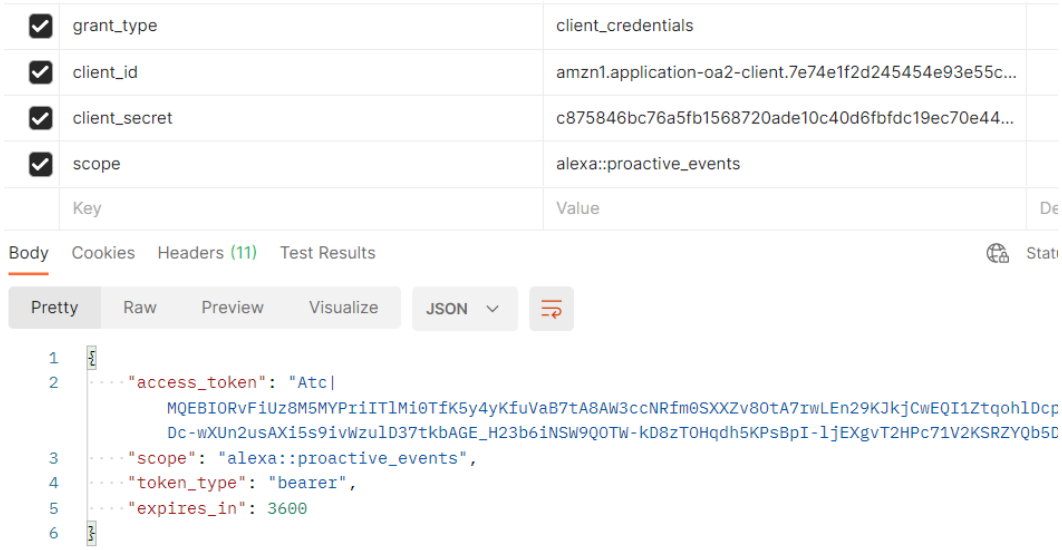
First, we **obtain an access token**, then we use that token to send the notification.

See “Request format to obtain access token” in https://developer.amazon.com/en-US/docs/alexa/smapi/proactive-events-api.html

To request a token, we send a POST message to Amazon with our **Client id** and **Client secret**. We get these from developer console > alexa skill > model > permissions



You can use postman to get access token and then send notification. See the Dabble lab video for that. Here’s the response you get when requesting a token:



We send a request to the URI: **https://api.amazon.com/auth/O2/token**

(If you’ve used LWA, you might recognise this)

With the header:

```
headers = {
    "Content-Type": "application/json;charset=UTF-8"
}
```

And a body (see above) with:

```
token_params = {
    "grant_type" : "client_credentials",
    "scope": "alexa::proactive_events",
    "client_id": CLIENT_ID,
```

```
    "client_secret": CLIENT_SECRET
}
```

1. **Python code to obtain Access Token**

We write a python program to do this. If you need the requests package, see here: https://pypi.org/project/requests/

Install using: **python -m pip install requests**

I've put my aws credentials in a separate file called **credentials.py**. Remember get these from developer > skill > models >permissions >

It looks like this:

```
key = {
    'CLIENT_ID' : 'put your client id here',
    'CLIENT_SECRET' : 'put your client secret here'
}
```

2. **Get access token**

Here's the code:

```
import time
import datetime
import json
import requests
import credentials

# constants
UTC_FORMAT = "%Y-%m-%dT%H:%M:%S.00Z"
TOKEN_URI = "https://api.amazon.com/auth/o2/token"

# Token access constants
CLIENT_ID = credentials.key['CLIENT_ID']
CLIENT_SECRET = credentials.key['CLIENT_SECRET']

def get_access_token():

    token_params = {
        "grant_type" : "client_credentials",
        "scope": "alexa::proactive_events",
        "client_id": CLIENT_ID,
        "client_secret": CLIENT_SECRET
    }

    token_headers = {
        "Content-Type": "application/json;charset=UTF-8"
    }

    response = requests.post(TOKEN_URI, headers=token_headers,
data=json.dumps(token_params), allow_redirects=True)

    print("Token response status: " + format(response.status_code))
    print("Token response body  : " + format(response.text))
```

```
    if response.status_code != 200:
        print("Error calling LWA!")
        return None

    access_token = json.loads(response.text)["access_token"]
    return access_token

def main():

    """Main function that requests Auth token from AMAZON """

    token = get_access_token()
    print (token)

if __name__ == "__main__":
    main()
```

Here's the output:

We want a response code of 200. The access token is the all the highlighted part that begins with Atc|

The response text is JSON format

```
{
        "access_token": "Atc|….. xoFKIQWpM",
        "scope": "alexa::proactive_events",
        "token_type": "bearer",
        "expires_in": 3600
}
```

The token expires in 3600 seconds (1 hour)

3. **Send the notification**.

We will use the access token we received to POST to the URI:
https://api.eu.amazonalexa.com/v1/proactiveEvents/**stages/development**

This time the header includes the token:

```
    header = {
```

```
        "Authorization": "Bearer {}".format(token),
        "Content-Type": "application/json;charset=UTF-8"
    }
```

The body contains our event(s), These can be unicast (to a single user) or multicast.

The payload will contain our event notification, in our case AMAZON.MessageAlert.Activated

This is described at https://developer.amazon.com/en-US/docs/alexa/smapi/proactive-events-api.html with the Message alert event in more detail at

https://developer.amazon.com/en-US/docs/alexa/smapi/schemas-for-proactive-events.html#message-alert

We'll send a body and event payload like this:

```
{
   "timestamp": "2021-03-01T15:17:58Z",
   "referenceId": "someUniqueReferenceNumber"
   "expiryTime": "2021-03-02T15:17:58Z",
   "event": {
      "name": "AMAZON.MessageAlert.Activated",
      "payload": {
         "state": {
            "status": "UNREAD",
            "freshness": "NEW"
         },
         "messageGroup": {
            "creator": {
               "name": "Johns python program"
            },
            "count": 1
         }
      }
   },
   "localizedAttributes": [
    {
      "locale": "en-GB",
      "providerName": "Alexa Events Example",
      "contentName": "Some content"
    }
   ],
   "relevantAudience": {
      "type": "Multicast",
      "payload": { }
   }
}
```

If you need to send to a specific user (type: Unicast), you will have to retrieve the userID from wherever you saved it when you handled the Proactive Event.

To get a timestamp and expiry time, I use the code:

```
timestamp  = time.strftime(UTC_FORMAT, time.gmtime(seconds))
seconds +=3600          # 1 hour for demo
expiry_time =  time.strftime(UTC_FORMAT, time.gmtime(seconds))
```

and to get a reference Id, I use uuid

```
reference_id = str(uuid.uuid4())
```

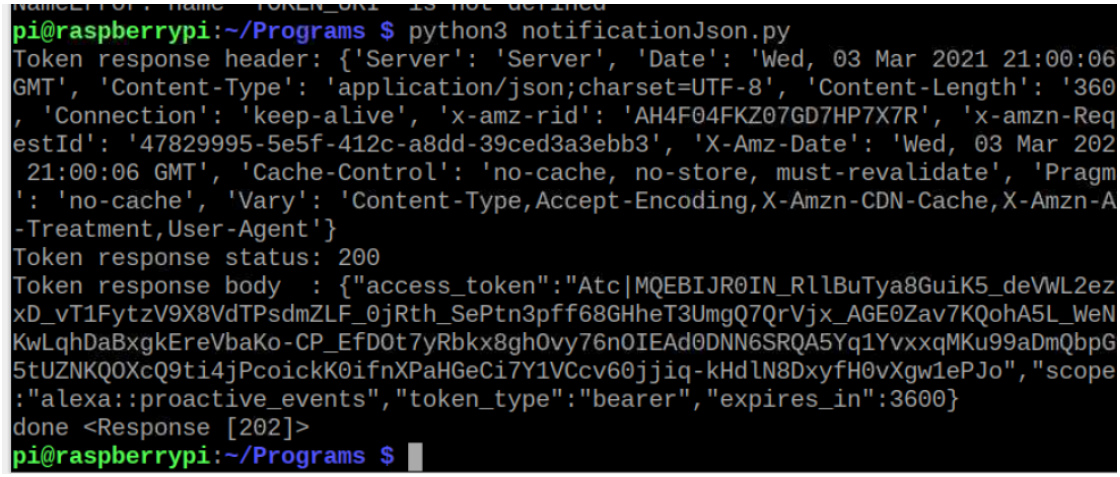
The notification will be "You have 1 NEW UNREAD message from John's python program.

Here's the whole code:

```
import time
import uuid
import datetime
import json
import requests
import credentials

# Token access constants
CLIENT_ID = credentials.key['CLIENT_ID']
CLIENT_SECRET = credentials.key['CLIENT_SECRET']
UTC_FORMAT = "%Y-%m-%dT%H:%M:%S.00Z"
TOKEN_URI = "https://api.amazon.com/auth/o2/token"
ALEXA_URI = "https://api.eu.amazonalexa.com/v1/proactiveEvents/stages/development"

def get_access_token():

   token_params = {
      "grant_type" : "client_credentials",
      "scope": "alexa::proactive_events",
      "client_id": CLIENT_ID,
      "client_secret": CLIENT_SECRET
   }

   token_headers = {
      "Content-Type": "application/json;charset=UTF-8"
   }

   response = requests.post(TOKEN_URI, headers=token_headers,
data=json.dumps(token_params), allow_redirects=True)

   if response.status_code != 200:
      print("Error calling LWA!")
      return None

   access_token = json.loads(response.text)["access_token"]
   return access_token

token = get_access_token()

headers = {
      "Authorization": "Bearer {}".format(token),
      "Content-Type": "application/json;charset=UTF-8"
   }

seconds = time.time()
timestamp  = time.strftime(UTC_FORMAT, time.gmtime(seconds))
```

```
reference_id = str(uuid.uuid4())
seconds +=3600            # 1 hour for demo
expiry_time =  time.strftime(UTC_FORMAT, time.gmtime(seconds))

params = {
    "timestamp": timestamp,
    "referenceId": reference_id,
    "expiryTime": expiry_time,

    "event": {
        "name": "AMAZON.MessageAlert.Activated",
        "payload": {
            "state": {
                "status": "UNREAD",
                "freshness": "NEW"
            },
            "messageGroup": {
                "creator": {
                    "name": "Johns python program"
                },
                "count": 1
            }
        }
    },
    "localizedAttributes": [
     {
        "locale": "en-GB",
        "providerName": "Alexa Events Example",
        "contentName": "Some content"
     }
    ],
    "relevantAudience": {
        "type": "Multicast",
        "payload": { }
    }
}

response = requests.post(ALEXA_URI, headers=headers, data=json.dumps(params),
allow_redirects=True)
print("done" , response)
```

And here’s the response



And my Alexa pings with a notification message

Response 202 is good!

You can now implement that into any code that you want to send a notification.

Best of luck

References:

https://github.com/alexa/alexa-cookbook/blob/master/feature-demos/skill-demo-proactive-events/order.js

https://github.com/alexa/alexa-cookbook/blob/master/feature-demos/skill-demo-proactive-events/order.js

https://github.com/alexa/alexa-smarthome/blob/master/sample_async/python/sample_async.py

https://medium.com/swlh/get-started-with-amazon-alexa-skills-proactive-events-api-5b082bcb282c

POST that tx and rx an access token back from amazon
See https://stackoverflow.com/questions/53795375/amazon-alexa-proactive-events-request-in-python-flask-ask

And https://stackoverflow.com/questions/55837349/alexa-skill-proactive-events-python-implementation