**Trapping out Alexa skill timeout**

****

**Video at: https://youtu.be/OsStzSfnAYc**

**Problem:**

Your skill has only 8 seconds to do its job. It then times out and closes the skill. What can you do if your skill sometimes takes longer than 8 seconds?

Here we’ll discuss how to use Python asyncio to trap the timeout and return to your skill.

Currently ASK uses Python version 3.9, so we’ll use **asyncio.wait\_for()**

<https://docs.python.org/3.9/library/asyncio-task.html#timeouts>

This is their example:

async def eternity():

# Sleep for one hour

await asyncio.sleep(3600)

print('yay!')

async def main():

# Wait for at most 1 second

try:

await asyncio.wait\_for(eternity(), timeout=1.0)

except asyncio.TimeoutError:

print('timeout!')

asyncio.run(main())

# Expected output:

#

# timeout!

In Python 3.11 you will be able to use asyncio.timeout(). Example:

async def main():

try:

async with asyncio.timeout(10):

await long\_running\_task()

except TimeoutError:

print("The long operation timed out, but we've handled it.")

print("This statement will run regardless.")

Note asyncio functions can only be called using asyncio,run(). You can’t call them directly.

**Example:**

Start a new Alexa Hosted skill (I’ve called it ‘long timeout’)

Check your Skill invocation name (from “change me”) if necessary.

Save and build the model.

We’ll modify the HelloWorld intent to call the tasks

At the top of the code add the imports:

import time

import asyncio

import random

Start by adding a random long task before class LaunchRequestHandler. This will wait 0,1,2 or 3 seconds

async def randomTime():

timeout = True

randomSecs = random.randint(0, 3) # 0,1,2 or 3 seconds

await asyncio.sleep(randomSecs)

timeout = False # executed if sleep finishes

return timeout

We’ll call this with a 2 second time out. Sometimes we will complete the (sleep) task and other we will timeout. We pass timeout back – True if we time out, False if the task completes.

We’ll call this with the **asyncio.wait\_for(),** passing the call and timeout (2 seconds)

async def waitForIt():

# Wait for at most 2 seconds

try:

timeout = True # True - we had a timeout. False - task (sleep) completed

timeout = await asyncio.wait\_for(randomTime(), timeout=2.0)

except asyncio.TimeoutError:

logger.info('timeout error')

return timeout

This has to be called (using asyncio.run()) from the HelloWorld intent. We’ll pass back timeout to see if we timed out (True) or False if we executed the task (sleep). We’ll use

timeout = asyncio.run(waitForIt())

Modify the HelloWorld intent as follows. I’ve also added **.ask** so that the skill reprompts:

class HelloWorldIntentHandler(AbstractRequestHandler):

"""Handler for Hello World Intent."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("HelloWorldIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

speak\_output = "Hello World!"

timeout = False # True - we had a timeout, False - task (sleep) complete

# ADD LONG CALL

start = time.time()

timeout = asyncio.run(waitForIt())

end= time.time()

if timeout:

speak\_output = "We timed out. It took " + str(int(end-start)) + " seconds"

else:

speak\_output = "Task completed, we didn't timeout. It took " + str(int(end-start)) + " seconds"

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask("Say hello again and see if we time out")

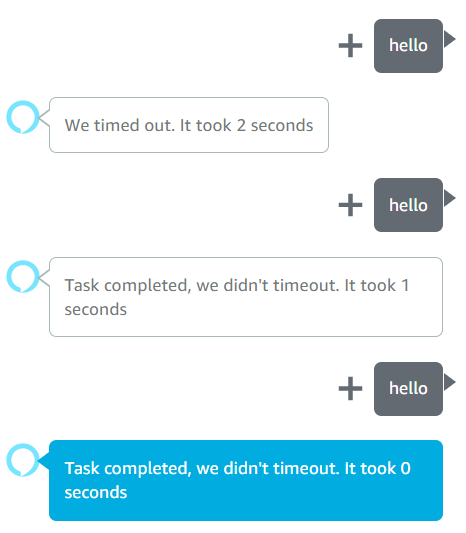
.response

)

I’ve also recorded how long the calls took.

lambda\_function.py at https://github.com/jallwork/AlexaTimeouts/blob/main/lambda\_function.py

Save, deploy and test your code (change to Development). Here’s my result:



You can see that sometimes the task takes less than a second and completes, and other times, times out.

You should now be able to trap out any long running tasks and return to your skill.

See: https://github.com/jallwork/AlexaTimeouts

and my book:

<https://www.elektor.com/programming-voice-controlled-iot-applications-with-alexa-and-raspberry-pi>

