**Alexa APLA Python**

****

**Alexa APLA Python example**

The APLA interface enables Alexa devices to render an audio response in an APL document

Ref: <https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/apla-interface.html>

APLA document is JSON code:

Alexa’s example:

{

"type": "APLA",

"version": "0.91",

"description": "Simple document that generates speech",

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"type": "Speech",

"content": "Hello world."

}

]

}

}

You send the document to Alexa with the [Alexa.Presentation.APLA.RenderDocument](https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/apla-interface.html" \l "renderdocument-directive) directive.

Ref: <https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/apla-interface.html>

This will add the APLA to the ‘directives’ part of our response (here’s the response in full):

{

"body": {

"version": "1.0",

"response": {

"directives": [

{

"type": "Alexa.Presentation.APLA.RenderDocument",

"token": "pagerToken",

"document": {

"type": "APLA",

"version": "0.91",

"description": "Simple document that generates speech",

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"type": "Speech",

"content": "Hello from A.P.L.A."

}

]

}

},

"datasources": {}

}

],

"type": "\_DEFAULT\_RESPONSE"

},

"sessionAttributes": {},

"userAgent": "ask-python/1.11.0 Python/3.7.12"

}

}

You can use multiple components to create different audio effects.

To add this to our code, we’ll use the **RenderDocumentDirective** from the APLA github:

<https://github.com/alexa/alexa-apis-for-python/tree/master/ask-sdk-model/ask_sdk_model/interfaces/alexa/presentation/apla>

If you’ve used APL before, you may remember using the code:

from ask\_sdk\_model.interfaces.alexa.presentation.apl import (

RenderDocumentDirective, ExecuteCommandsDirective, SpeakItemCommand, AutoPageCommand, HighlightMode)

This time we’ll use:

**from ask\_sdk\_model.interfaces.alexa.presentation.apla import RenderDocumentDirective**

Note **APLA** not APL

We’ll start with a simple example but have the APLA json as a separate document

**Procedure**

* Create an Alexa Hosted skill using Python, then in the next screen choose ‘Start from Scratch’ skill and “Continue with Template”
* Check your invocation command (I’ve called mine Audio skill)
* Select the APL interface - click **Interfaces** and switch Alexa Presentation Language **on**
* Click **Save Interfaces** at the top and **rebuild** the model
* Edit the code to add the APL code:

At the top of your code (after all the ‘from’ statements is a good place) add the following:

**from ask\_sdk\_model.interfaces.alexa.presentation.apla import RenderDocumentDirective**

Below that, add:

**import json**

**def \_load\_apl\_document(file\_path):**

**# type: (str) -> Dict[str, Any]**

**"""Load the apl json document at the path into a dict object."""**

**with open(file\_path) as f:**

**return json.load(f)**

We’ll use that to load our (separate) APLA JSON file

Now create a new file calling it APLA.json in the same folder as your python code:

Type in or paste the following code and save it as APLA.json

{

"type": "APLA",

"version": "0.91",

"description": "Simple document that generates speech",

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"type": "Speech",

"content": "Hello from A.P.L.A."

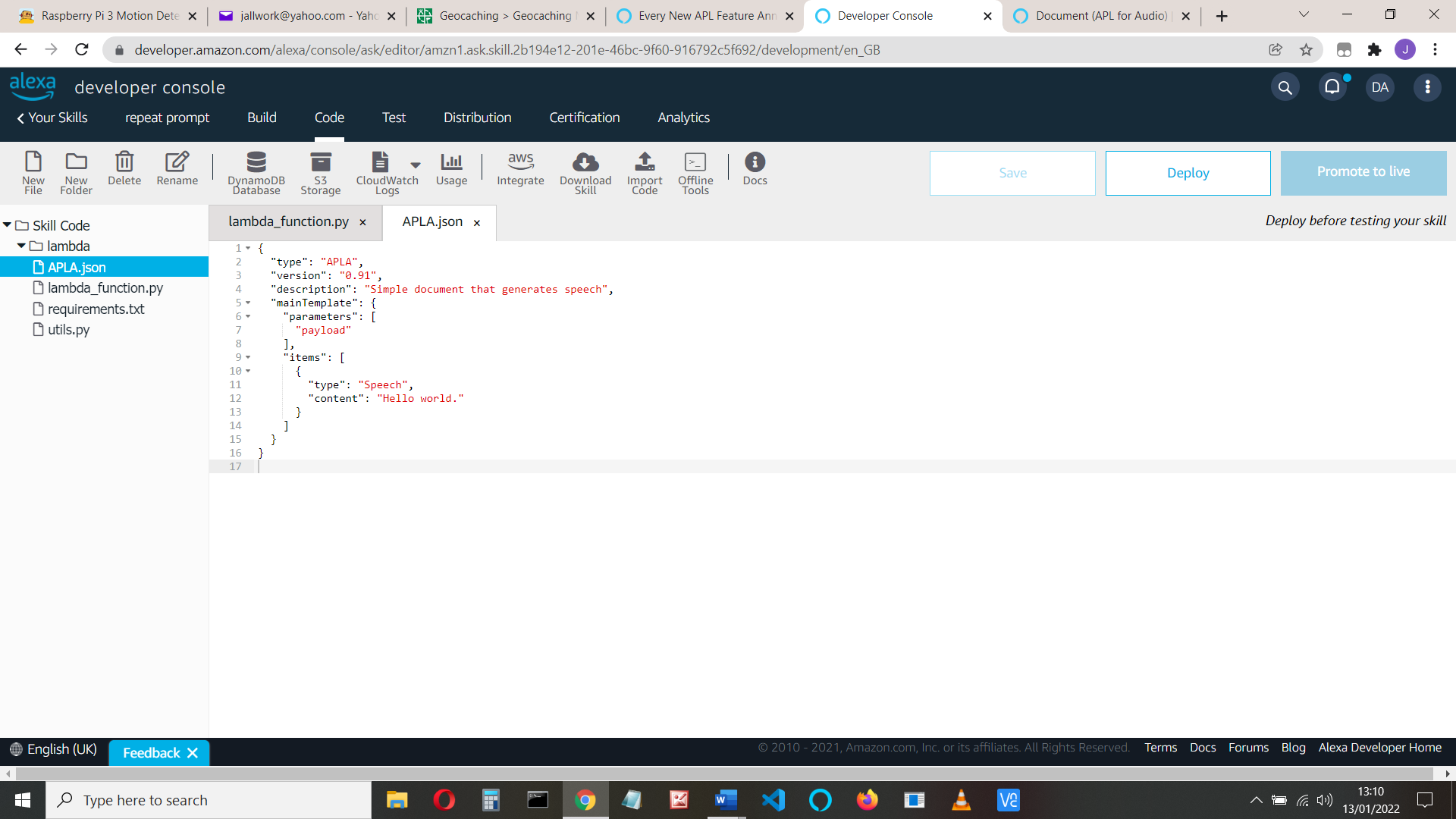
}

]

}

}

So that it looks like this:



Modify the HelloWorld intent to the following:

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!"

# .ask("add a reprompt if you want to keep the session open for the user to respond")

return (

handler\_input.response\_builder

#.speak(speak\_output)

.add\_directive(

RenderDocumentDirective(

token="pagerToken",

document=\_load\_apl\_document("APLA.json"),

datasources={}

)

)

.response

)

Save and deploy your code.

Test the program (click the Test tab) – enable testing and invoke your skill (type in your skill name), then type / say “Hello”

You should hear "Hello from A.P.L.A."

If it doesn’t work, look at your Cloudwatch logs

More on APL display at <https://youtu.be/fa9pZ-IQh9E>

Next video – more involved APLA code and using reprompt