**Alexa APL Video skill**

Full code available at:

See: <https://github.com/jallwork/Alexa_APL_Video>

You tube videos start at:

**Part One – Basic APL video**

**Part Two - Integrate APL and code**

**Part Three - Adding play/pause controls**

**Part Four - Adding Speech Commands**

Let’s see how to add Video to our Alexa skills using APL.

If you don’t know what APL is, take a look at my video at:

<https://www.youtube.com/watch?v=84d8c8_LJM0> or <https://youtu.be/fa9pZ-IQh9E>

To show the video, we’ll add a Video component to our APL. This itself doesn't have any controls, instead it provides the events and commands necessary to control the player using APL Media commands.

If you want to find out more, see:

<https://developer.amazon.com/en-US/docs/alexa/video/understand-the-video-skill-api.html>

Or:

<https://developer.amazon.com/docs/video-skills-multimodal-devices/introduction.html>

And APL video:

<https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/apl-video.html>

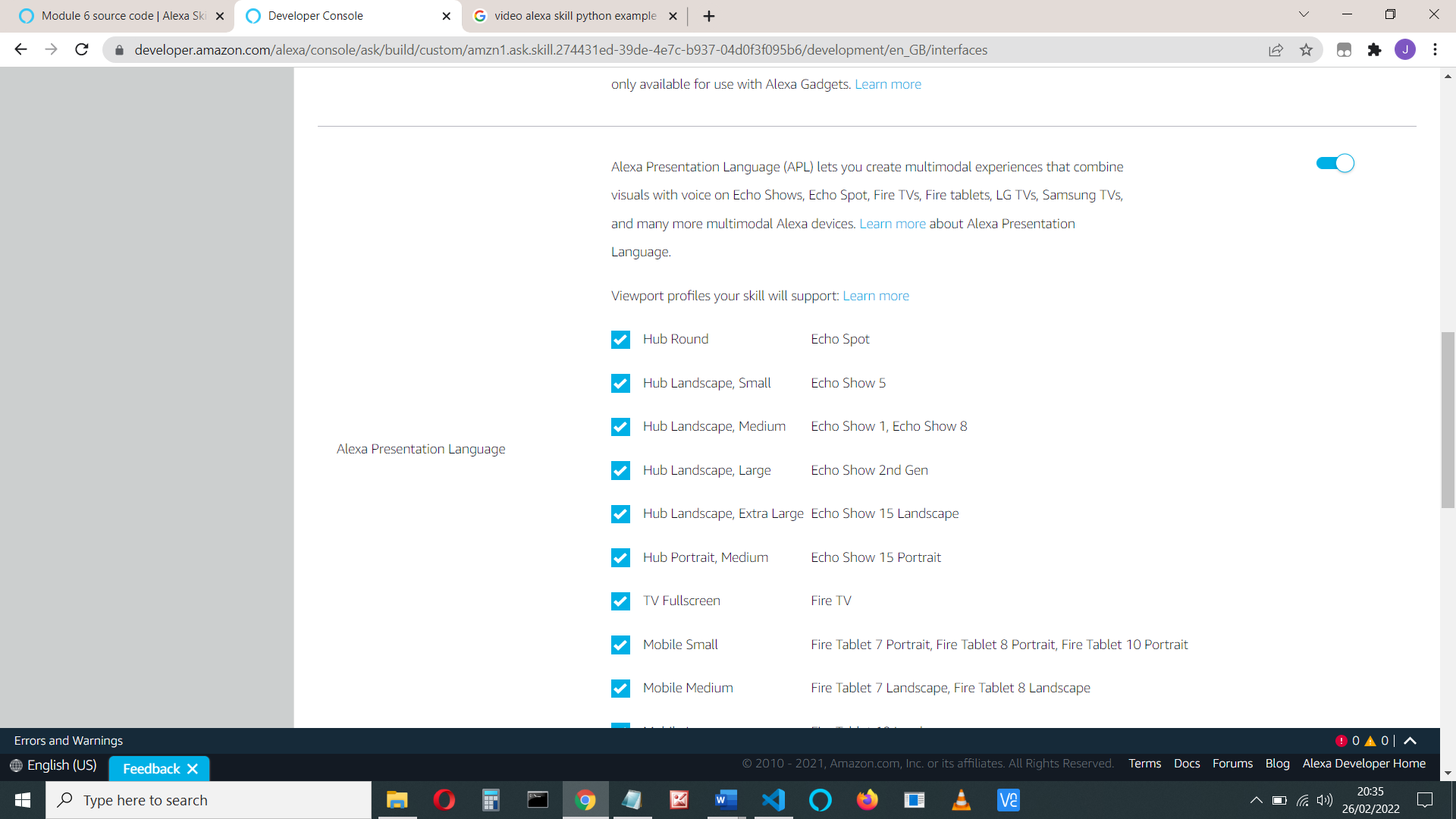
**Part One – Basic APL video**

Create new skill in developer:

Python > Create from scratch, I called it **Johns video skill**

**Now add APL interface**.

Click Interfaces and select Alexa Presentation Language

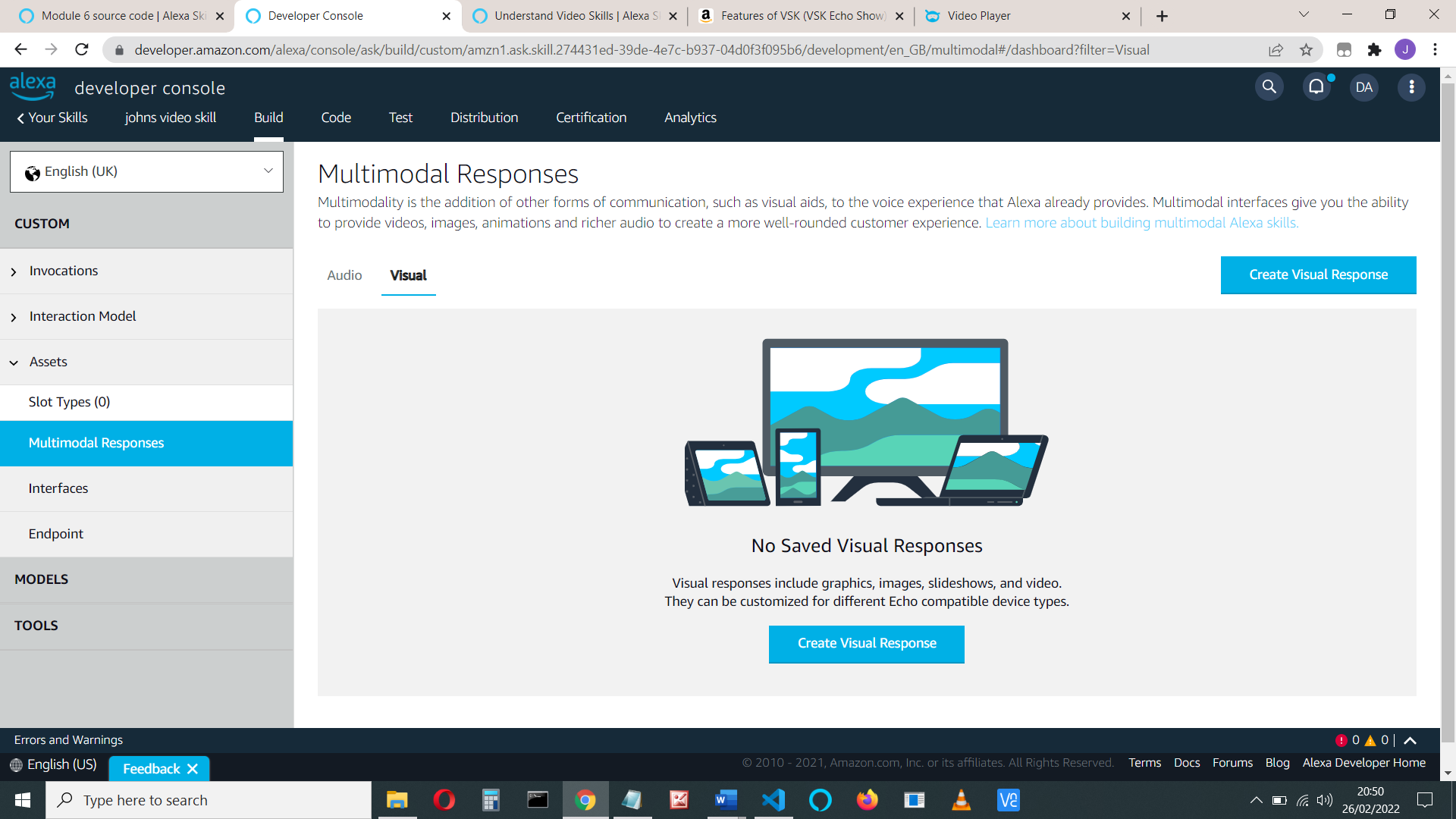


You can choose which devices you want to support. (All of them)

Save and build your model

Now we’re going to build and test the APL code using the Multimodal Response

Click Build > Multimodal Response > Video



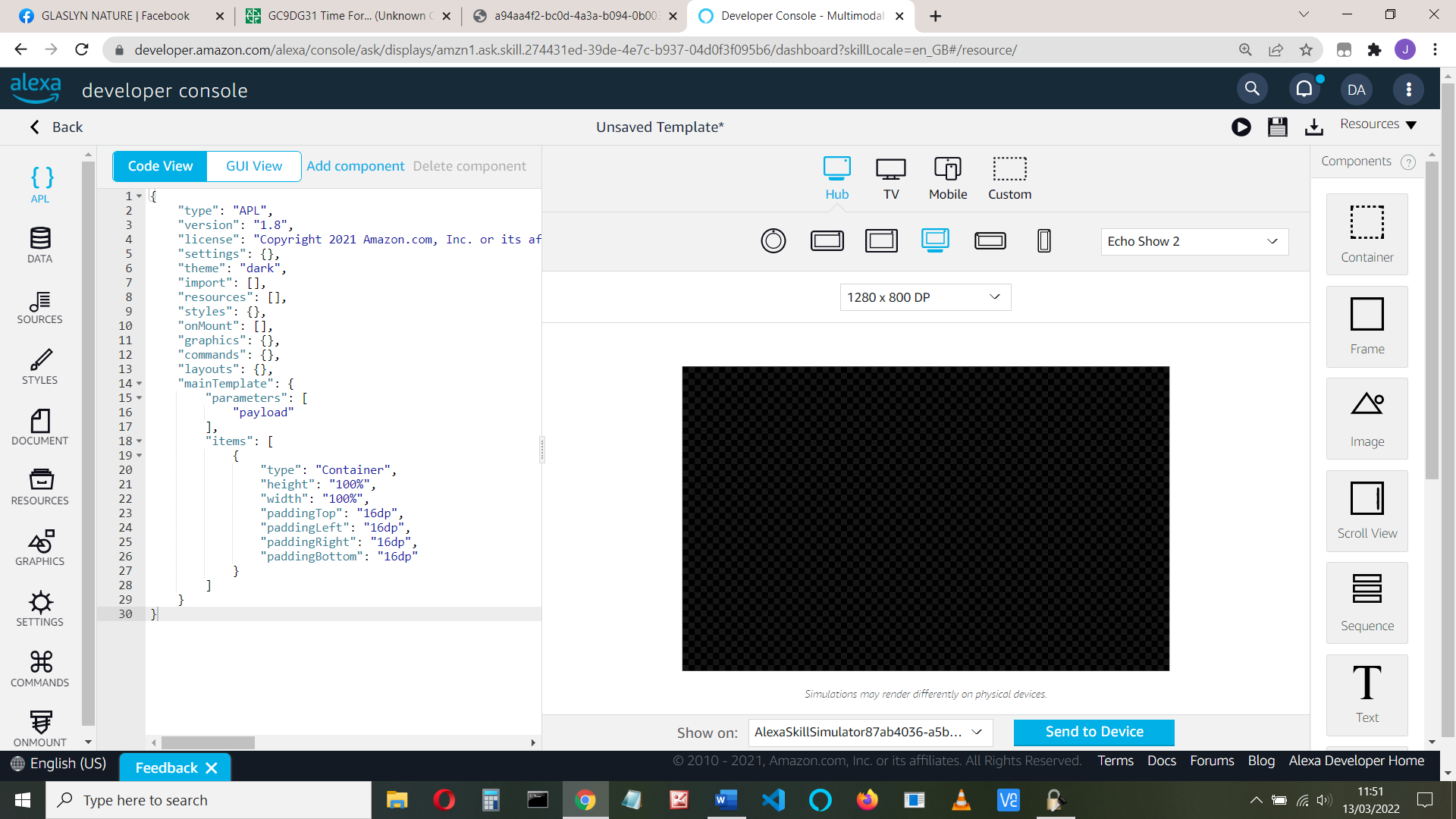
Choose Create Visual Response, then Blank document.

Let’s create our own and add some code. First, we’ll just add a video in a container and play some video, then we’ll add controls.

**Either:**

1. Drag the container icon onto the screen (see [1] below).

Select the Code View [2] and see that it has added the container:



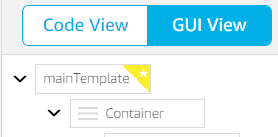
2

1

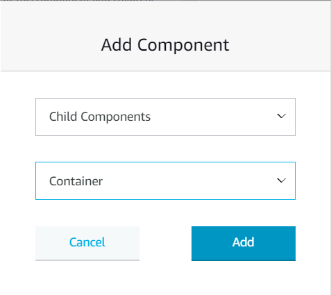
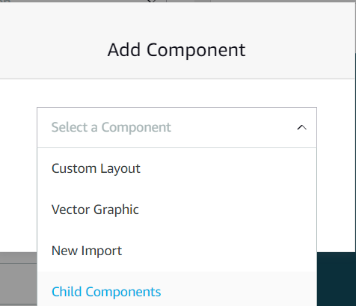
1. Use Code view to add/edit your code (if you know how to do that)

OR

1. Use the GUI View to add components.



You’d select the mainTemplate, then ‘Add component’ then Child components and Container:



Now drag a video element  onto the container, or add using Add component.

A video component should have been added to your JSON (scroll down in the code from the container [1] to find it):

"items": [

{

"type": "Video",

"height": "80%",

"width": "100%"

}

Add height and width if they are not there. Set the height to "80%" and width to "100%"

If it hasn’t been added, now is a good time to ensure the latest APL version is used. Change the import.

Your full code should look like this:

{

"type": "APL",

"version": "1.9",

**"import": [**

**{**

**"name": "alexa-layouts",**

**"version": "1.5.0"**

**}**

**],**

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"type": "Container",

"height": "100%",

"width": "100%",

"alignItems": "center",

"items": [

{

"type": "Video",

"height": "80%",

"width": "100%"

}

]

}

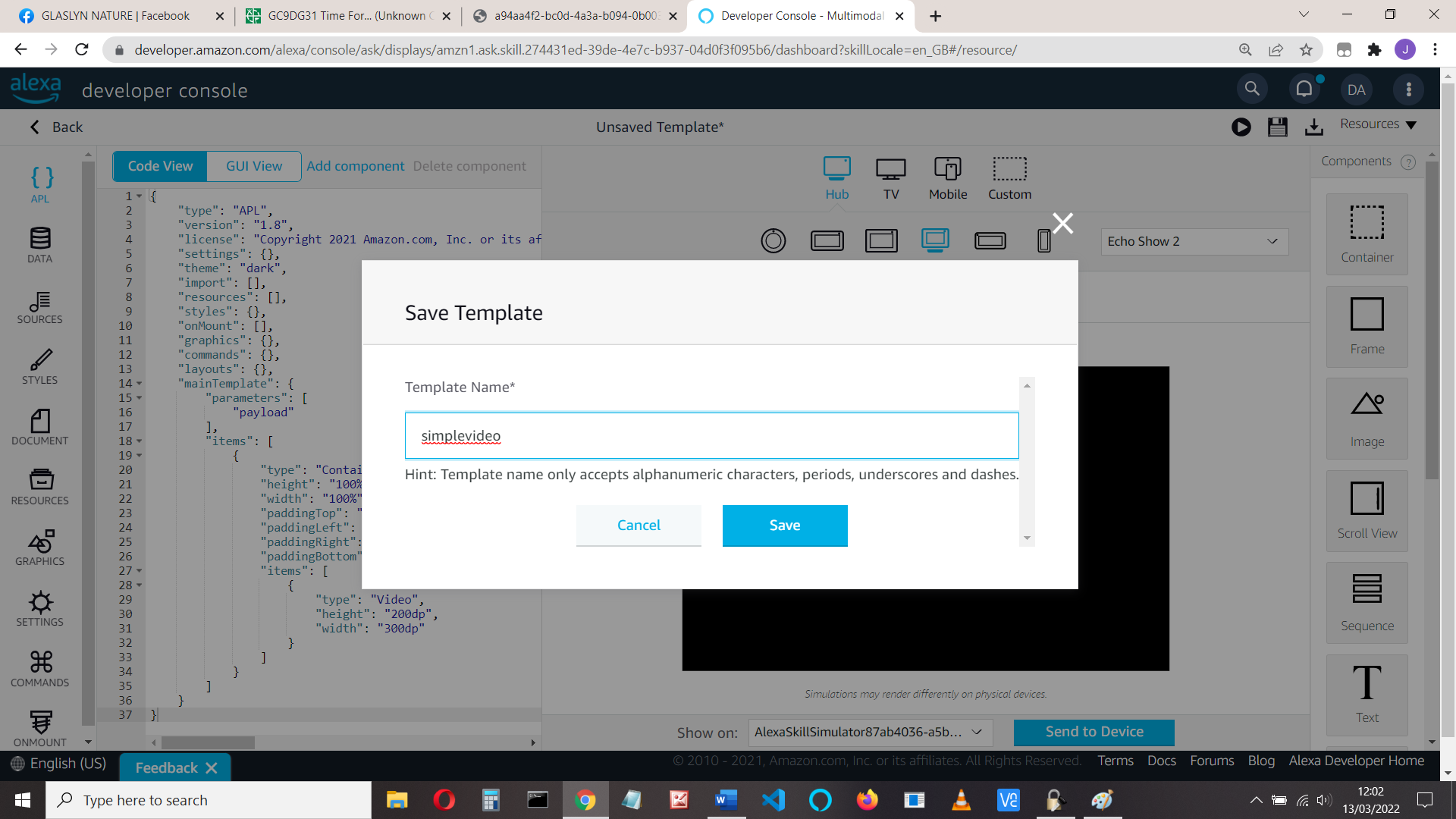
]

}

}

I’ve made the height of the video component 80% to make room for a control later on.

Click the disk symbol  to save the template:



Now we need to add the video to be played.

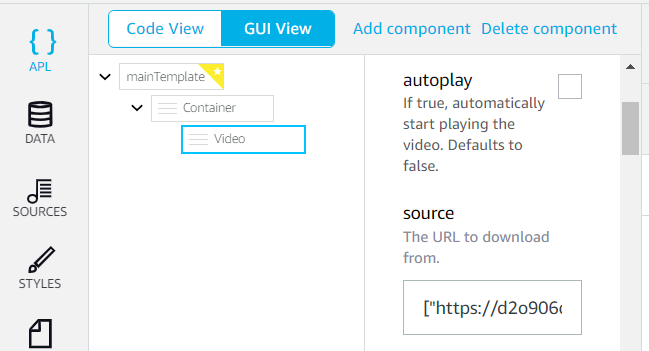
I’m using the Big Buck video from:

"https://test-videos.co.uk/vids/bigbuckbunny/mp4/h264/1080/Big\_Buck\_Bunny\_1080\_10s\_1MB.mp4"

It’s only 10 seconds long.

<http://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4>

You can edit the JSON, but’ we’ll use the GUI view, which displays the possible options for that component.



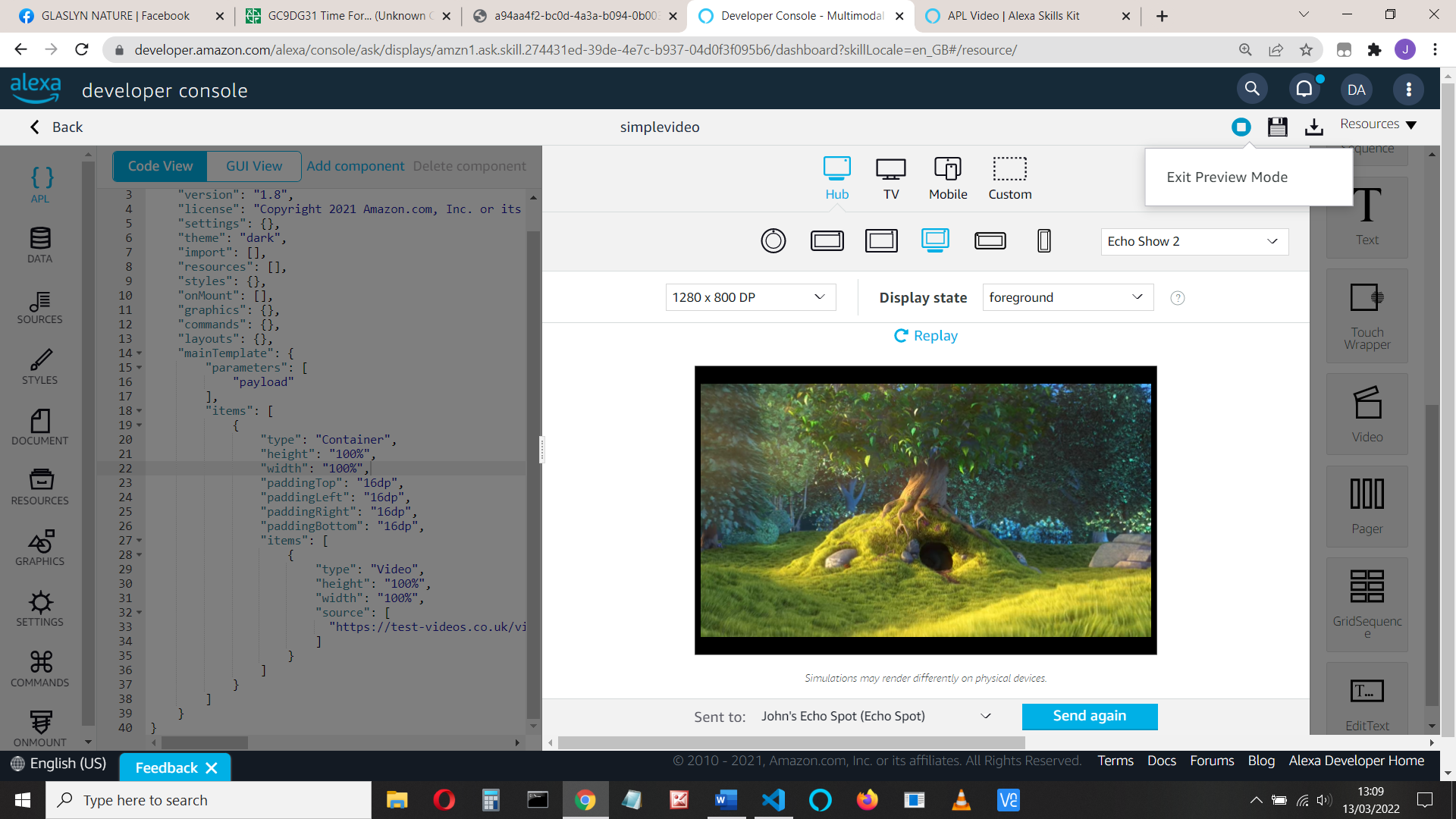
Add into the source box:

**https://test-videos.co.uk/vids/bigbuckbunny/mp4/h264/1080/Big\_Buck\_Bunny\_1080\_10s\_1MB.mp4**

Select autoplay ‘true’ as well

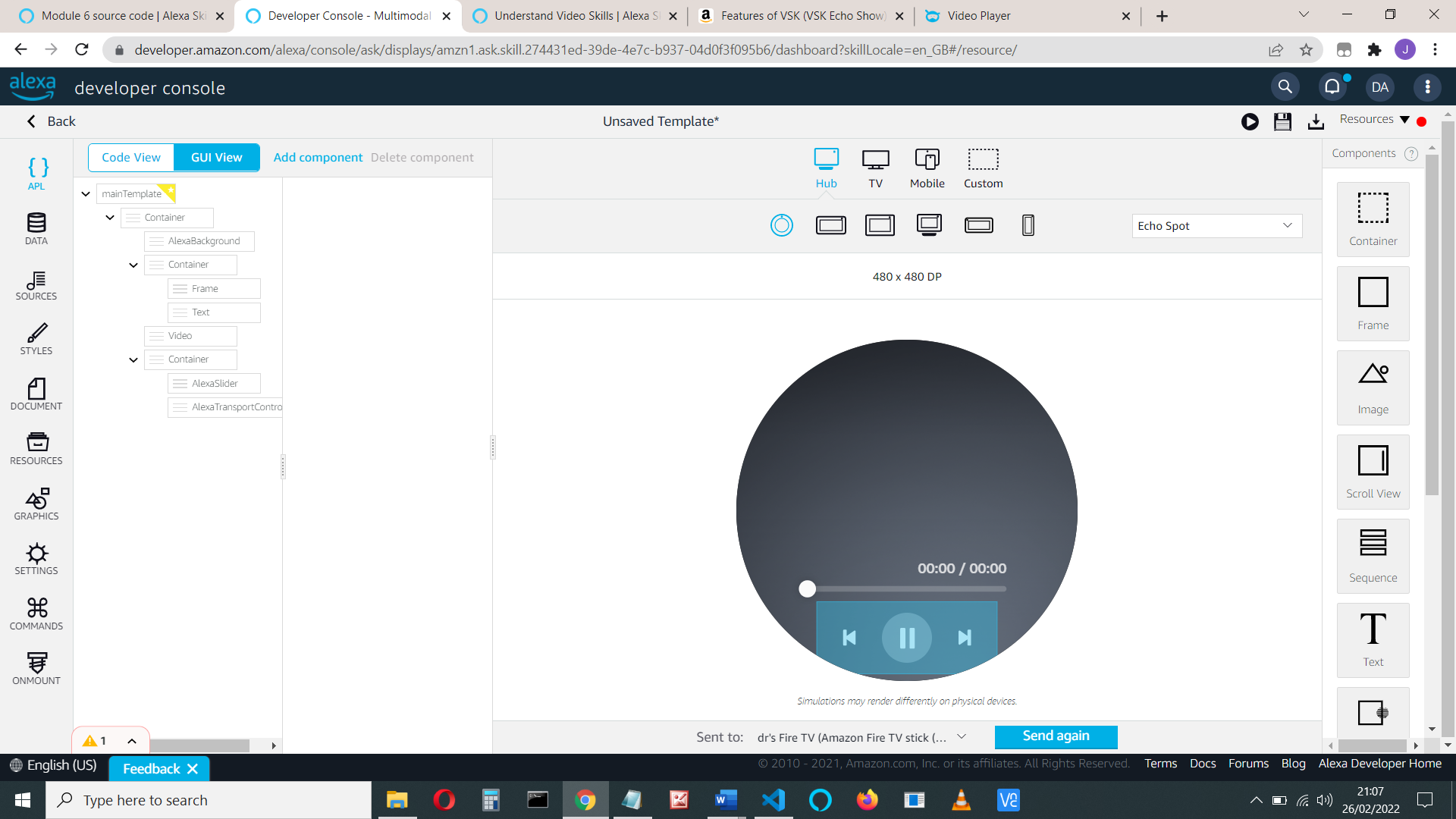
Save the template.

You can now press the play button or choose send to a device.



It should play, if not make sure you set autoplay = true.

You can also see how it will display on various Alexa devices



(This shows the slider and video controls we’ll add later)

My JSON is now like this:

{

"type": "APL",

"version": "1.9",

"import": [

{

"name": "alexa-layouts",

"version": "1.5.0"

}

],

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"type": "Container",

"height": "100%",

"width": "100%",

"alignItems": "center",

"items": [

{

"**autoplay": true,**

**"source":** "https://test-videos.co.uk/vids/bigbuckbunny/mp4/h264/1080/Big\_Buck\_Bunny\_1080\_10s\_1MB.mp4",

"type": "Video",

"width": "100%",

"height": "80%"

}

]

}

]

}

}

**Part Two - Integrate APL and code**

Before we add any more videos or controls, let’s add the APL to our code. We’ll display the APL video when the user says ’Hello’

Change to the code tab (Click the back button, then the code tab).

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

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

**from ask\_sdk\_model.interfaces.alexa.presentation.apl import ExecuteCommandsDirective**

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."""**

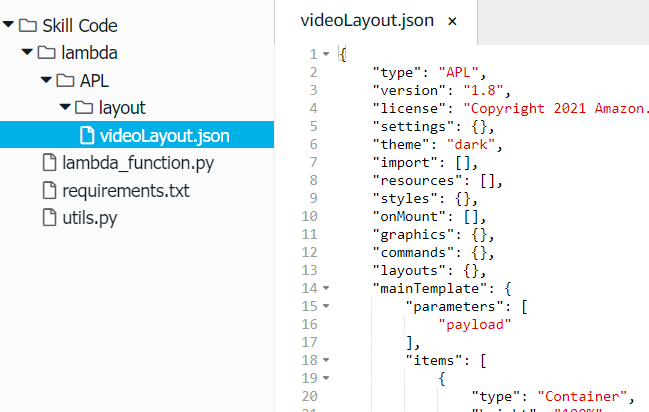
**with open(file\_path) as f:**

**return json.load(f)**

We’ll use that to load our APL JSON file

Create a new folder called ‘apl’ in our lambda folder, and within that a ‘layouts’ folder

In there, create a file called videoLayout.json. Copy all the JSON code into this file. It should look like this:



**Modify the HelloWorld intent.**

First, we’ll add a check in our code to see if the user’s device can show videos.

When the user's device has a screen that supports APL, the handler input request envelope context.system.device.supported\_interfaces includes **alexa\_presentation\_apl**

We use the (python) code:

if handler\_input.request\_envelope.context.system.device.supported\_interfaces.alexa\_presentation\_apl is not None:

OK

(else: we don’t have a screen)

Add the directives to the response – a RenderDocumentDirective for the JSON layout code and an ExecuteCommandsDirective for the action to take (play)

All of the HelloWorldIntent handler becomes:

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!"

if handler\_input.request\_envelope.context.system.device.supported\_interfaces.alexa\_presentation\_apl is not None:

return (

handler\_input.response\_builder

.add\_directive(

RenderDocumentDirective(

token="videoID",

document=\_load\_apl\_document("./APL/layout/videoLayout.json"),

datasources={}

)

)

.add\_directive(

ExecuteCommandsDirective(

token="videoID",

commands = [

{

"type": "ControlMedia",

"componentId": " videoPlayerId",

"command": "play"

}

]

)

)

.response

)

else:

speak\_output = "Sorry your device can't show videos"

return (

handler\_input.response\_builder

.speak(speak\_output)

.response

)

Note:

When your response includes both directives, the device displays the specified document first, then runs the commands, provided the token matches in both directives

Save, deploy and test to see that it works. Note that "autoplay" has to be true for it to play.

Try typing ‘hello’ whilst it’s playing. The video re-starts again at the beginning.

What happens at the end of the video? Let’s add another video (or two) for it to play

Change the source code in the videoLayout.json file to:

"source": [

"https://test-videos.co.uk/vids/bigbuckbunny/mp4/h264/1080/Big\_Buck\_Bunny\_1080\_10s\_1MB.mp4",

"https://d2o906d8ln7ui1.cloudfront.net/videos/AdobeStock\_277864451.mov",

"https://d2o906d8ln7ui1.cloudfront.net/videos/AdobeStock\_292807382.mov",

"https://d2o906d8ln7ui1.cloudfront.net/videos/AdobeStock\_320293418.mov"

],

Try it - It automatically plays the next video.

**Databinding**

We should really put the source outside of the layout code. We’ll bind the source to the data

Return to your **Visual Multimodal Responses**, Code view and change the source line to

"source" : "${payload.track.url}"

and in the data tab add the JSON code:

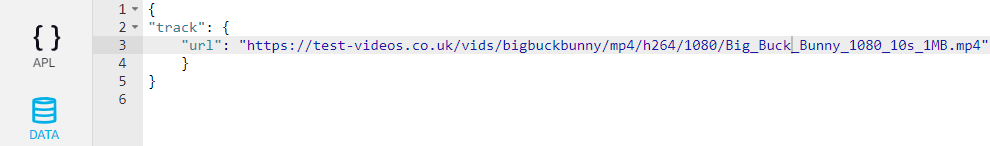
{

"track": {

"url": "https://test-videos.co.uk/vids/bigbuckbunny/mp4/h264/1080/Big\_Buck\_Bunny\_1080\_10s\_1MB.mp4"

}

}



Test this code works by using the preview tab.

You can change the datasource to show more videos as we did before:



**Code**

To do this to your code, change the “source” in your videoLayout.json as above to

"${payload.track.url}"

and change the RenderDocumentDirective to include the datasources as follows:

RenderDocumentDirective(

token="videoID",

document=\_load\_apl\_document("./APL/layout/videoLayout.json"),

datasources={

"track": {

"url": [

"https://test-videos.co.uk/vids/bigbuckbunny/mp4/h264/1080/Big\_Buck\_Bunny\_1080\_10s\_1MB.mp4",

"https://d2o906d8ln7ui1.cloudfront.net/videos/AdobeStock\_277864451.mov",

"https://d2o906d8ln7ui1.cloudfront.net/videos/AdobeStock\_292807382.mov",

"https://d2o906d8ln7ui1.cloudfront.net/videos/AdobeStock\_320293418.mov"

]}

}

Of course, you can save the datasource as a separate file and use load\_apl\_document as we do with the document

**Part Three - Adding play/pause controls**

If you’re happy, let’s move on and add **Transport controls**. See

<https://developer.amazon.com/en-US/docs/alexa/alexa-design/layout-transport-controls.html>

This provides a play/pause button and either next/previous or skip 10/30 seconds buttons:



Remember, you should also add voice controls for these.

We’ll work on a new JSON. Return to your Visual Multimodal Responses and clone your original JSON.

Edit the clone. I’ve called it controls.

We’ll add the controls at the base of our video (so make them large enough and also with enough color contrast)

Ensure you have the latest version of APL, i.e.

"import": [{

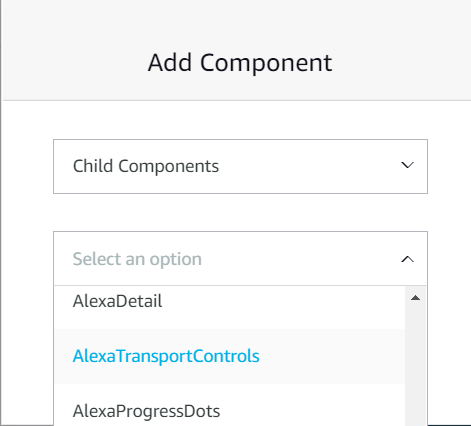
"name": "alexa-layouts",

"version": "1.5.0"

}],

Either use the GUI View to add a transport control or the Code view to add it as follows:

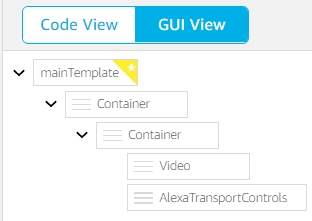
Select the container and add the AlexaTransportControl component



Notice how many more components we can add with the new version of alexa-layouts

If you don’t see this, check that you have added the import mentioned above.

GUI View



Code View (mainTemplate only shown):

"items": [

{

"source" : "${payload.track.url}",

"autoplay": true,

"type": "Video",

"width": "100%",

**"height": "80%"**

},

**{**

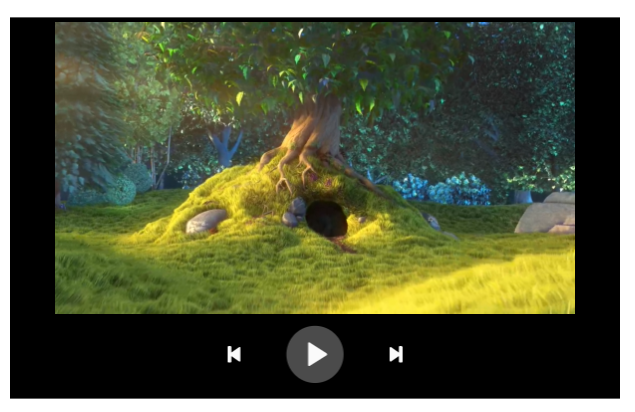
**"type": "AlexaTransportControls",**

**"justifyContent": "center"**

**}**

The height of the Video component has been reduced to make room for the controls.

You should see the controls. They don’t control the video yet though (it doesn’t know which video component to control):



Now let’s add the video control and onPlay and onPause events to update the control.

The events are described here:

<https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/apl-commands-media.html#command>

There are also next, previous, rewind, seek and setTrack commands

Add them and an ID to your Video component:

"id": "videoPlayerId",

(the full APL is below)



It’s set it up so that the video is playing and the first click on the controls pauses the video.

To the transport control needs that so we know which video component is being controlled, so add a mediaComponentId () and a link to the (controlling) play/pause toggle button.

**{**

**"type": "AlexaTransportControls",**

**"justifyContent": "center",**

**"mediaComponentId": "videoPlayerId",**

**"playPauseToggleButtonId": "playPauseToggleButtonId"**

**}**

The full APL is:

{

"type": "APL",

"version": "1.9",

"import": [

{

"name": "alexa-layouts",

"version": "1.5.0"

}

],

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"type": "Container",

"height": "100%",

"width": "100%",

"alignItems": "center",

"items": [

{

"source": "${payload.track.url}",

"autoplay": true,

"type": "Video",

"width": "100%",

"height": "80%",

"shrink": 1,

**"id": "videoPlayerId",**

**"onPause": [{**

**"type": "SetValue",**

**"componentId": "playPauseToggleButtonId",**

**"property": "checked",**

**"value": false**

**}],**

**"onPlay": [{**

**"type": "SetValue",**

**"componentId": "playPauseToggleButtonId",**

**"property": "checked",**

**"value": true**

**}]**

},

{

"type": "AlexaTransportControls",

"justifyContent": "center",

"mediaComponentId": "videoPlayerId",

"playPauseToggleButtonId": "playPauseToggleButtonId"

}

]

}

]

}

}

Check that the play and pause and next and previous track work.

**Adding a progress slider**

Let’s add a slider to show how far along the track we are.

We’ll use an Alexa slider responsive component (AlexaSlider). This displays an interactive progress bar. *Users can drag the bar to scrub content or change settings.*

There’s a good slider APL example at:

<https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/apl-alexa-slider-layout.html#alexaslider-example>

This shows a slider that updates text as the user moves the thumb, and has a button to reset the slider to zero

The Slider uses milliseconds to represent the total time and the progress. We’ll set:

1. videoTotalValue to the total number of milliseconds.
2. videoProgressValue to determine the position of the slider. The user will also be able to drag the slider to a new position for the video
3. sliderThumbPosition to determine the position that the user has dragged the slider to.

Clone your controls json (I’ve called mine ‘slider’)

Add an AlexaSlider to your JSON under the Transport controls:

Add the following values for it (either ion the GUI or code)

"id": "slider",

"progressValue": "${videoProgressValue}",

"totalValue": "${videoTotalValue}",

"positionPropertyName": "sliderThumbPosition",

"metadataDisplayed": true,

"metadataPosition": "above\_right",

"width": "75%",

"paddingBottom": "90dp",

The AlexaSlider uses ‘bound’ data so we need to add the bind values. Add these to the main container:

"bind": [

{

"name": "sliderThumbPosition",

"value": 0

},

{

"name": "videoProgressValue",

"type": "number",

"value": 0

},

{

"name": "videoTotalValue",

"type": "number",

"value": 0

}

],

We need to update these when a new track is loaded and plays. The Video component sends events when playing. We’ll use these events to update the slider:

For the Video component we’ll add:

* onTrackUpdate

When a track is updated, we’ll set new (total and progress) values for the slider

* onPlay

We already set the ‘playPauseToggleButtonId checked’ to ‘true’. We’ll also update the videoTotalValue

* onTrackUpdate

When a new track is loaded, we’ll set new slider values for videoTotalValue (to event.duration) and videoProgressValue to zero.

For the slider we’ll add:

* onUpCommand

We’ll send a seek command to the video component to seek to the position ‘sliderThumbPosition – videoProgressValue’

As the slider data is bound to the data, it will automatically update when these values changed.

Add an onTrackUpdate

"onTrackUpdate": [

{

"type": "SetValue",

"property": "videoTotalValue",

"value": "${event.duration}"

},

{

"type": "SetValue",

"property": "videoProgressValue",

"value": 0

}

],

Add an onTimeUpdate

"onTimeUpdate": [

{

"type": "SetValue",

"property": "videoProgressValue",

"value": "${event.currentTime}"

},

{

"type": "SetValue",

"componentId": "slider",

"property": "progressValue",

"value": "${videoProgressValue}"

},

{

"type": "SetValue",

"property": "videoTotalValue",

"value": "${event.duration}"

}

]

And add after the OnPlay for the ‘playPauseToggleButtonId checked’ true code

{

"type": "SetValue",

"property": "videoTotalValue",

"value": "${event.duration}"

}

Add an onUpCommand to the slider:

"onUpCommand": [

{

"type": "ControlMedia",

"componentId": "videoPlayerId",

"command": "seek",

"value": "${sliderThumbPosition - videoProgressValue}"

}

]

And there we have it:

{

"type": "APL",

"version": "1.9",

"import": [

{

"name": "alexa-layouts",

"version": "1.5.0"

}

],

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"alignItems": "center",

"items": [

{

"source": "${payload.track.url}",

"autoplay": true,

"type": "Video",

"width": "100%",

"height": "80%",

"shrink": 1,

"id": "videoPlayerId",

"onPause": [

{

"type": "SetValue",

"componentId": "playPauseToggleButtonId",

"property": "checked",

"value": false

}

],

"onPlay": [

{

"type": "SetValue",

"componentId": "playPauseToggleButtonId",

"property": "checked",

"value": true

},

{

"type": "SetValue",

"property": "videoTotalValue",

"value": "${event.duration}"

}

],

"onTrackUpdate": [

{

"type": "SetValue",

"property": "videoTotalValue",

"value": "${event.duration}"

},

{

"type": "SetValue",

"property": "videoProgressValue",

"value": 0

}

],

"onTimeUpdate": [

{

"type": "SetValue",

"property": "videoProgressValue",

"value": "${event.currentTime}"

},

{

"type": "SetValue",

"componentId": "slider",

"property": "progressValue",

"value": "${videoProgressValue}"

},

{

"type": "SetValue",

"property": "videoTotalValue",

"value": "${event.duration}"

}

]

},

{

"type": "AlexaTransportControls",

"justifyContent": "center",

"mediaComponentId": "videoPlayerId",

"playPauseToggleButtonId": "playPauseToggleButtonId"

},

{

"type": "AlexaSlider",

"sliderId": "slider",

"progressValue": "${videoProgressValue}",

"totalValue": "${videoTotalValue}",

"positionPropertyName": "sliderThumbPosition",

"metadataDisplayed": true,

"metadataPosition": "above\_right",

"width": "75%",

"paddingBottom": "90dp",

"onUpCommand": [

{

"type": "ControlMedia",

"componentId": "videoPlayerId",

"command": "seek",

"value": "${sliderThumbPosition - videoProgressValue}"

}

]

}

],

"justifyContent": "center",

"type": "Container",

"bind": [

{

"name": "sliderThumbPosition",

"value": 0

},

{

"name": "videoProgressValue",

"type": "number",

"value": 0

},

{

"name": "videoTotalValue",

"type": "number",

"value": 0

}

],

"width": "100%",

"height": "100%"

}

]

}

}

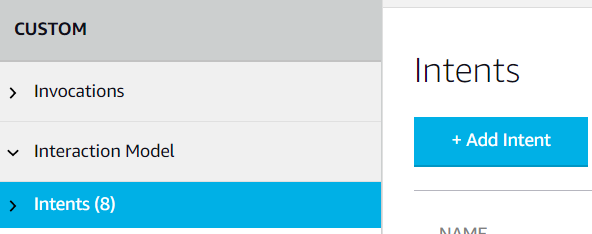
Try it. Copy the APL to your code and try it on a device too.

We should really add touch controls and speech commands. We’ll add next, previous and play spoken commands

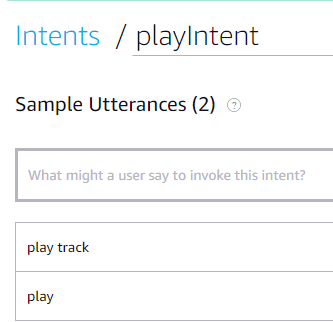
**Part Four - Adding Speech Commands**

Return to the build tab

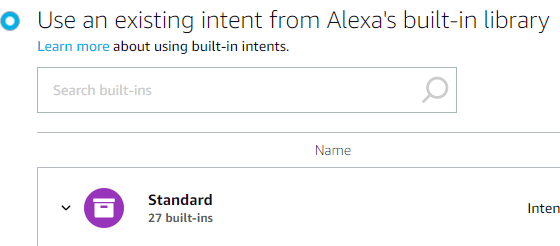
Select Intents and Add Intent



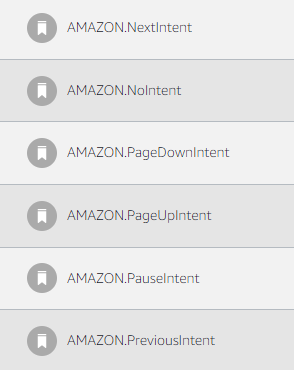
Add a PlayIntent and some utterances (play , play track)



Add two more intents – this time use the Standard intents – next and previous



Find and add the Next and Previous Intents:



Add some utterances for these too, (next, next track) and (previous, go back) etc.

Save and Build the model.

Return to the code and add the code:

Add the sb handlers at the end of the code:

sb = SkillBuilder()

sb.add\_request\_handler(LaunchRequestHandler())

sb.add\_request\_handler(HelloWorldIntentHandler())

**sb.add\_request\_handler(PlayIntentHandler())**

**sb.add\_request\_handler(NextIntentHandler())**

**sb.add\_request\_handler(PreviousIntentHandler())**

And we’ll send a controlMedia command using the executeCommandDirective (play, next and previous) to the videoPlayerId:

class PlayIntentHandler(AbstractRequestHandler):

"""Handler for Play Intent."""

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

# type: (HandlerInput) -> bool

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

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("in play handler")

speak\_output = "Playing"

return (

handler\_input.response\_builder

.add\_directive(

ExecuteCommandsDirective(

token="videoID",

commands = [

{

"type": "ControlMedia",

"componentId": "videoPlayerId",

"command": "play"

}

]

)

)

.speak(speak\_output)

.response

)

class NextIntentHandler(AbstractRequestHandler):

"""Handler for Next Intent."""

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

# type: (HandlerInput) -> bool

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

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("in next handler")

speak\_output = "Skipping to next track"

return (

handler\_input.response\_builder

.add\_directive(

ExecuteCommandsDirective(

token="videoID",

commands = [

{

"type": "ControlMedia",

"componentId": "videoPlayerId",

"command": "next"

}

]

)

)

.speak(speak\_output)

.response

)

class PreviousIntentHandler(AbstractRequestHandler):

"""Handler for Previous Intent."""

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

# type: (HandlerInput) -> bool

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

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("in previous handler")

speak\_output = "Going to previous track"

return (

handler\_input.response\_builder

.add\_directive(

ExecuteCommandsDirective(

token="videoID",

commands = [

{

"type": "ControlMedia",

"componentId": "videoPlayerId",

"command": "previous"

}

]

)

)

.speak(speak\_output)

.response

)

I suppose we should also have a pause too. I’ll leave that as an exercise.

**References**

I can highly recommend the APL ninja web site:

<https://apl.ninja/>

In particular for video: <https://apl.ninja/aruntalkstech/video-player-bfel>

You might find these references useful too:

Dabble labs:

https://www.youtube.com/watch?v=ps8oETMtMZA

**Appendix 1 – full controls APL**

{

"type": "APL",

"version": "1.9",

"theme": "dark",

"import": [

{

"name": "alexa-layouts",

"version": "1.5.0"

}

],

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"items": [

{

"items": [

{

"source" : "${payload.track.url}",

"autoplay": true,

"type": "Video",

"width": "100%",

"height": "80%",

"id": "videoPlayerId",

"onPause": [

{

"type": "SetValue",

"componentId": "alexaPlayPauseToggleButton",

"property": "checked",

"value": true

}

],

"onPlay": [

{

"type": "SetValue",

"componentId": "alexaPlayPauseToggleButton",

"property": "checked",

"value": false

}

]

},

{

"type": "AlexaTransportControls",

"justifyContent": "center",

"mediaComponentId": "videoPlayerId",

"playPauseToggleButtonId": "playPauseToggleButtonId",

"autoplay": true

}

],

"justifyContent": "center",

"type": "Container",

"width": "100%",

"height": "100%",

"alignItems": "center",

"paddingTop": "16dp",

"paddingLeft": "16dp",

"paddingRight": "16dp",

"paddingBottom": "16dp"

}

],

"justifyContent": "center",

"type": "Container",

"width": "100%",

"height": "100%"

}

]

}

}

**Data payload:**

{

"track": {

"url":

[

"https://test-videos.co.uk/vids/bigbuckbunny/mp4/h264/1080/Big\_Buck\_Bunny\_1080\_10s\_1MB.mp4",

"http://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4",

"https://d2o906d8ln7ui1.cloudfront.net/videos/AdobeStock\_277864451.mov",

"https://d2o906d8ln7ui1.cloudfront.net/videos/AdobeStock\_292807382.mov",

"https://d2o906d8ln7ui1.cloudfront.net/videos/AdobeStock\_320293418.mov"

]

}

}

**Appendix 2 Full slider APL** (uses same data as Appendix 1)

{

"type": "APL",

"version": "1.9",

"import": [

{

"name": "alexa-layouts",

"version": "1.5.0"

}

],

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"alignItems": "center",

"items": [

{

"source": "${payload.track.url}",

"autoplay": true,

"type": "Video",

"width": "100%",

"height": "80%",

"shrink": 1,

"id": "videoPlayerId",

"onPause": [

{

"type": "SetValue",

"componentId": "playPauseToggleButtonId",

"property": "checked",

"value": false

}

],

"onPlay": [

{

"type": "SetValue",

"componentId": "playPauseToggleButtonId",

"property": "checked",

"value": true

},

{

"type": "SetValue",

"property": "videoTotalValue",

"value": "${event.duration}"

}

],

"onTrackUpdate": [

{

"type": "SetValue",

"property": "videoTotalValue",

"value": "${event.duration}"

},

{

"type": "SetValue",

"property": "videoProgressValue",

"value": 0

}

],

"onTimeUpdate": [

{

"type": "SetValue",

"property": "videoProgressValue",

"value": "${event.currentTime}"

},

{

"type": "SetValue",

"componentId": "slider",

"property": "progressValue",

"value": "${videoProgressValue}"

},

{

"type": "SetValue",

"property": "videoTotalValue",

"value": "${event.duration}"

}

]

},

{

"type": "AlexaTransportControls",

"justifyContent": "center",

"mediaComponentId": "videoPlayerId",

"playPauseToggleButtonId": "playPauseToggleButtonId"

},

{

"type": "AlexaSlider",

"sliderId": "slider",

"progressValue": "${videoProgressValue}",

"totalValue": "${videoTotalValue}",

"positionPropertyName": "sliderThumbPosition",

"metadataDisplayed": true,

"metadataPosition": "above\_right",

"width": "75%",

"paddingBottom": "90dp",

"onUpCommand": [

{

"type": "ControlMedia",

"componentId": "videoPlayerId",

"command": "seek",

"value": "${sliderThumbPosition - videoProgressValue}"

}

]

}

],

"justifyContent": "center",

"type": "Container",

"bind": [

{

"name": "sliderThumbPosition",

"value": 0

},

{

"name": "videoProgressValue",

"type": "number",

"value": 0

},

{

"name": "videoTotalValue",

"type": "number",

"value": 0

}

],

"width": "100%",

"height": "100%"

}

]

}

}