Amazon [Team] | User Guide

Speech Synthesis for Prototypes

INTRODUCTION

Our projects have explored various content modes as a means of direction and feedback. After several multimodal prototypes that have been heavily guided by recorded audio, we have turned our attention to synthesized speech, as it presents far less complexity in terms of authoring, deploying, maintaining, and localizing media files with text for captions.

We first experimented with the HTML5 Web Speech API but quickly found that Amazon Polly provided more life-like speech and a range of options for everything from customizing pronunciation to generating speech marks for captions. It is our tool of choice for speech synthesis going forward.

APPROACH

Our standard guidance for a prototype that uses synthesized speech is that each step or interval in the application can optionally speak one or more sentences of text in a female voice, using the English language in an American accent. Localization of these components can be revisited in later stages of demonstration for other markets.

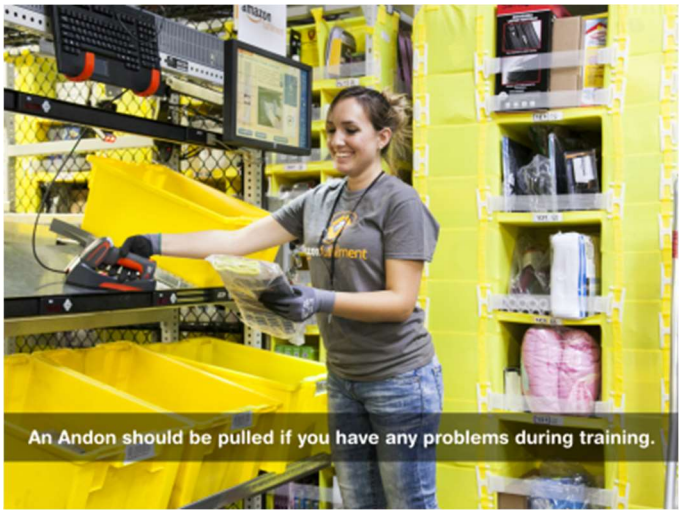
The user should be able to turn on captions optionally, with sentences of text appearing in a black bar on the interface in time with the sentence currently being spoken. For this functionality, we have a method of submitting written text and having speech either generated at runtime, or sound files provided for immediate download and playback. We should include cues in the application code from which to synchronize the appearance of the written text with the spoken word.

Amazon Polly provides the capability to generate these assets at runtime via the AWS SDK, but that scenario is more favorable to situations where unpredictable text is generated or produced from user input at runtime and incurs significantly more cloud transactions, and thus higher AWS fees. We instead use the Polly console to create and download our sound and text files beforehand and write application logic to parse these assets as needed.

METHOD

Amazon Polly takes text input in either plain text form or Speech Synthesis Markup Language (SSML) and outputs the MP3 and associated text files, which contain the text broken into components of choice (words, sentences, or custom breakpoints) with associated time codes for start times of those components.

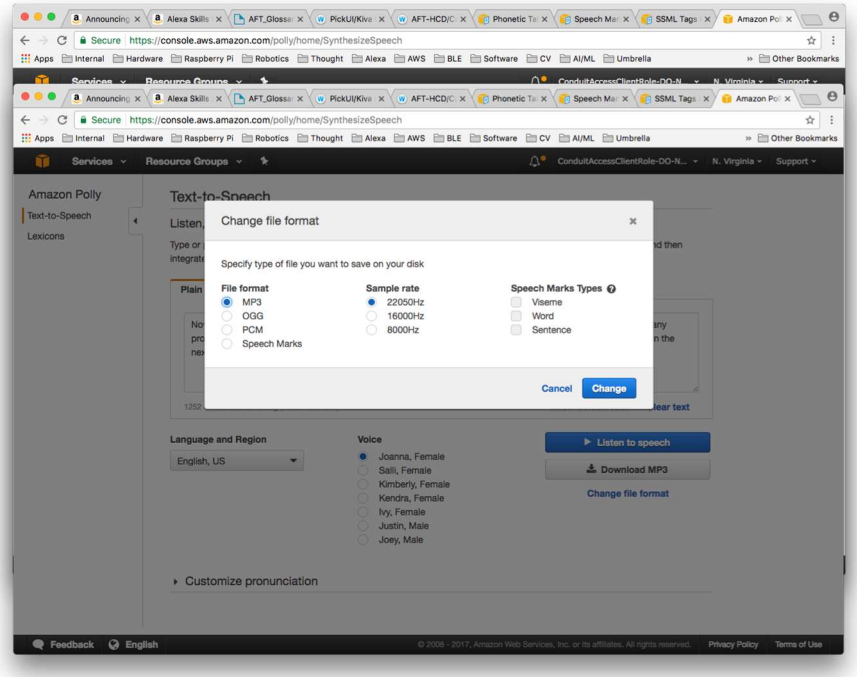
Consider the following screen from a fictional training application:



The full set of text to be spoken on this screen might be, for example, “Now, let's look at your workstation. This is where you'll pick items. An Andon should be pulled if you are having any problems during training. Please take a look to locate the light. Press the Next button when you are ready to learn the next item.”

Manual Asset Generation

1. Log into the AWS Management Console and open the Amazon Polly service. Paste your text into the window as plain text. Select the female voice, Joanna, and English, US region.



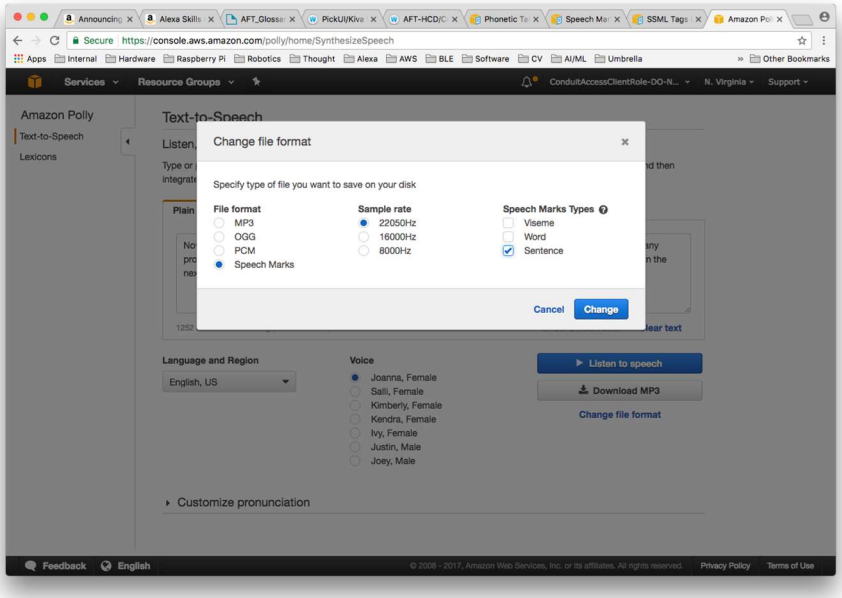
2. Click the **Change file format** link to get the modal window and select MP3 file format and a sample rate. Close the modal and click the **Download MP3** button.

3. Click the **Change file format** link to get the modal window again. Select **Speech Marks** as the file format and **Sentence** under **Speech Marks Type**. Close the modal and click the **Download Speech Marks** button.

4. This would generate a JSON file with metadata like below. In this example, you could consume the data in your prototype to sync actions with sentence start and end times.

```
{"time":0,"type":"sentence","start":7,"end":43,"value":"Now, let's look at your workstation."}
{"time":2638,"type":"sentence","start":44,"end":76,"value":"This is where you'll pick items."}
{"time":4437,"type":"sentence","start":77,"end":150,"value":"An Andon should be pulled if you are having any problems during training."}
{"time":8069,"type":"sentence","start":151,"end":190,"value":"Please take a look to locate the light."}
{"time":10552,"type":"sentence","start":191,"end":255,"value":"Press the Next button when you are ready to learn the next item."}
```

Programmatic Asset Generation



If you want to include the asset generation functionality directly into your prototype application, you can use the AWS SDK to write logic that achieves the same results. For example[1]:

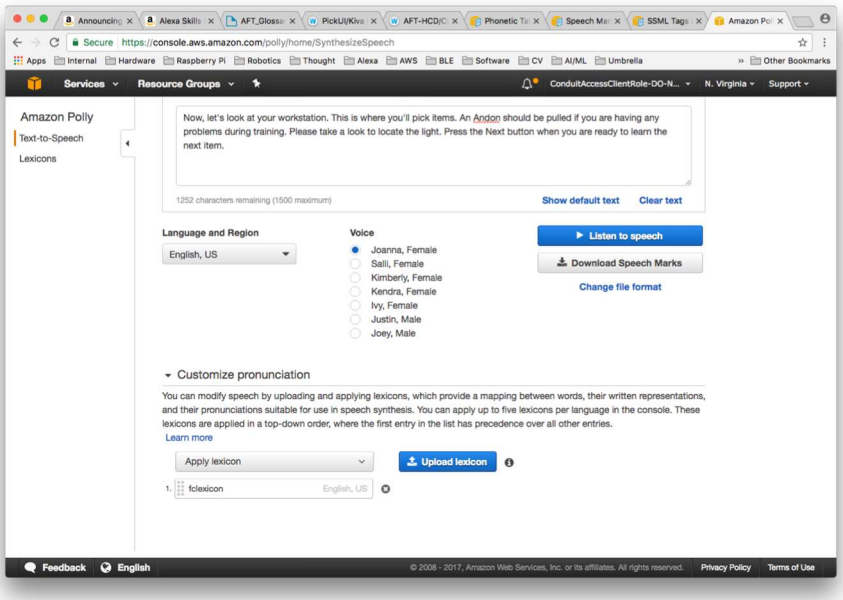
```
var params = {
  LexiconNames: [
    "example"
  ],
```

```
  OutputFormat: "mp3",
  SampleRate: "8000",
  SpeechMarkTypes: [sentence],
  Text: "All Gaul is divided into three parts",
  TextType: "text",
  VoiceId: "Joanna"
};

polly.synthesizeSpeech(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else console.log(data);              // successful response
  /*
   data = {
      AudioStream: <Binary String>, ContentType: "audio/mpeg",
      RequestCharacters: 37
   }
  */
});
```

PRONUNCIATION

As Polly is the deep learning module for speech synthesis in Alexa, it can speak a wide range of words quite naturally but can experience trouble with words it has not been exposed to very often. It will make awkward attempts to speak branded words, or words unique to Amazon or our processes, and acronyms are especially troublesome since we spell some out as we speak, and pronounce others as though they were words.There are a



range of SSML tags that can assist with pronunciations in both general and specific terms, including foreign

languages and access. But with Polly, you can use pronunciation lexicons written in an XML standard format and uploaded by AWS region to customize how a very particular set of words are synthesized to be spoken whenever they are encountered.

Notice in the screen above under **Customize Pronunciation**, you can also upload a lexicon or series of lexicons for a glossary that gets applied to either a specific process. For example, you might need a prototype to utilize a set of spoken words and acronyms for a fulfillment process prototype, which Polly will not speak as desired by default. To do so, you can build a lexicon using Amazon Polly phonemes to create XML with your custom pronunciations to upload. The example below shows a lexicon file with a few terms common to the fulfillment industry.

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
      xmlns=http://www.w3.org/2005/01/pronunciation-lexicon
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
       http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
      alphabet="ipa" xml:lang="en-US">
  <lexeme>
    <grapheme>andon</grapheme>
    <phoneme>'ændɔn</phoneme>
  </lexeme>
  <lexeme>
    <grapheme>bin</grapheme>
    <phoneme>'bɪn</phoneme>
  </lexeme>
  <lexeme>
    <grapheme>tote</grapheme>
    <phoneme>toʊt</phoneme> </lexeme>
  <lexeme>
    <grapheme>ASIN</grapheme>
    <grapheme>asin</grapheme>
    <phoneme>eɪsɪn</phoneme>
  </lexeme>
  <lexeme>
   <grapheme>rebin</grapheme>
    <phoneme>ɹi: 'bɪn</phoneme>
  </lexeme>
  <lexeme>
    <grapheme>TAKT</grapheme>
    <grapheme>takt</grapheme>
    <phoneme>tækt</phoneme>
  </lexeme>
</lexicon>
```

CITATIONS

1. *AWS JavaScript SDK*. Amazon. Retrieved June 18, 2023, from https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/Polly.html#synthesizeSpeech-property