Amazon Polly

Developer Guide

Amazon Polly: Developer Guide

Table of Contents

What Is Amazon Polly?

Amazon Polly is a cloud service that converts text into lifelike speech. You can use Amazon Polly to develop applications that increase engagement and accessibility. Amazon Polly supports multiple languages and includes a variety of lifelike voices, so you can build speech-enabled applications that work in multiple locations and use the ideal voice for your customers. With Amazon Polly, you only pay for the text you synthesize. You can also cache and replay Amazon Polly's generated speech at no additional cost.

Additionally, Amazon Polly includes a number of Neural Text-to-Speech (NTTS) voices, delivering ground-breaking improvements in speech quality through a new machine learning approach, thereby offering to customers the most natural and human-like text-to-speech voices possible. Neural TTS technology also supports a Newscaster speaking style that is tailored to news narration use cases.

Common use cases for Amazon Polly include, but are not limited to, mobile applications such as newsreaders, games, eLearning platforms, accessibility applications for visually impaired people, and the rapidly growing segment of Internet of Things (IoT).

Amazon Polly is certified for use with regulated workloads for HIPAA (the Health Insurance Portability and Accountability Act of 1996), and Payment Card Industry Data Security Standard (PCI DSS).

Some of the benefits of using Amazon Polly include:

- **High quality** – Amazon Polly offers both new neural TTS and best-in-class standard TTS technology to synthesize the superior natural speech with high pronunciation accuracy (including abbreviations, acronym expansions, date/time interpretations, and homograph disambiguation).

- **Low latency** – Amazon Polly ensures fast responses, which make it a viable option for low-latency use cases such as dialog systems.

- **Support for a large portfolio of languages and voices** – Amazon Polly supports dozens of voices languages, offering male and female voice options for most languages. Neural TTS currently supports three British English voices and eight US English voices. This number will continue to increase as we bring more neural voices online. US English voices Matthew and Joanna can also use the Neural Newscaster speaking style, similar to what you might hear from a professional news anchor.

- **Cost-effective** – Amazon Polly's pay-per-use model means there are no setup costs. You can start small and scale up as your application grows.

- **Cloud-based solution** – On-device TTS solutions require significant computing resources, notably CPU power, RAM, and disk space. These can result in higher development costs and higher power consumption on devices such as tablets, smart phones, and so on. In contrast, TTS conversion done in the AWS Cloud dramatically reduces local resource requirements. This enables support of all the available languages and voices at the best possible quality. Moreover, speech improvements are instantly available to all end-users and do not require additional updates for devices.

Are You a First-time User of Amazon Polly?

If you are a first-time user of Amazon Polly, we recommend that you read the following sections in the listed order:

1. **How Amazon Polly Works (p. 3)** – This section introduces various Amazon Polly inputs and options that you can work with in order to create an end-to-end experience.
2. **Getting Started with Amazon Polly (p. 4)** – In this section, you set up your account and test Amazon Polly speech synthesis.
3. **Example Applications (p. 155)** – This section provides additional examples that you can use to explore Amazon Polly.

How Amazon Polly Works

Amazon Polly converts input text into life-like speech. You call one of the speech synthesis methods, provide the text that you want to synthesize, choose one of the Neural Text-to-Speech (NTTS) or Standard Text-to-Speech (TTS) voices, and specify an audio output format. Amazon Polly then synthesizes the provided text into a high-quality speech audio stream.

- **Input text** – Provide the text that you want to synthesize, and Amazon Polly returns an audio stream. You can provide the input as plain text or in Speech Synthesis Markup Language (SSML) format. With SSML you can control various aspects of speech, such as pronunciation, volume, pitch, and speech rate. For more information, see Generating Speech from SSML Documents (p. 100).

- **Available voices** – Amazon Polly provides a portfolio of languages and a variety of voices, including a bilingual voice (for both English and Hindi). For most languages you can choose from several voices, both male and female. When launching a speech synthesis task, you specify the voice ID, and then Amazon Polly uses this voice to convert the text to speech. Amazon Polly is not a translation service —the synthesized speech is in the same language as the text. However, if the text is in a different language than designated for the voice, numbers represented as digits (for example, *53*, not *fifty-three*) are synthesized in the language of the voice and not the text. For more information, see Voices in Amazon Polly.

- **Output format** – Amazon Polly can deliver the synthesized speech in multiple formats. You can select the audio format that suits your needs. For example, you might request the speech in the MP3 or Ogg Vorbis format for consumption by web and mobile applications. Or, you might request the PCM output format for consumption by AWS IoT devices and telephony solutions.

What's Next?

If you are new to Amazon Polly, we recommend that you to read the following topics in order:

- Getting Started with Amazon Polly (p. 4)
- Example Applications (p. 155)
- Limits in Amazon Polly (p. 186)

Getting Started with Amazon Polly

Amazon Polly provides simple API operations that you can easily integrate with your existing applications. For a list of supported operations, see Actions (p. 209). You can use either of the following options:

- AWS SDKs – When using the SDKs, your requests to Amazon Polly are automatically signed and authenticated using the credentials you provide. This is the recommended choice for building your applications.
- AWS CLI – You can use the AWS CLI to access any of Amazon Polly functionality without having to write any code.

The following sections describe how to get set up and provide an introductory exercise.

Topics

Step 1: Set Up an AWS Account and Create a User

Before you use Amazon Polly for the first time, complete the following tasks:

1. Step 1.1: Sign up for AWS (p. 4)
2. Step 1.2: Create an IAM User (p. 5)

Step 1.1: Sign up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Amazon Polly. You are charged only for the services that you use.

With Amazon Polly, you pay only for the resources you use. If you are a new AWS customer, you can get started with Amazon Polly for free. For more information, see AWS Free Usage Tier.

If you already have an AWS account, skip to the next step. If you don't have an AWS account, perform the steps in the following procedure to create one.

To create an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Note your AWS account ID because you'll need it for the next step.

Step 1.2: Create an IAM User

Services in AWS, such as Amazon Polly, require that you provide credentials when you access them so that the service can determine whether you have permissions to access the resources owned by that service. The console requires your password. You can create access keys for your AWS account to access the AWS CLI or API. However, we don't recommend that you access AWS using the credentials for your AWS account. Instead, we recommend that you use AWS Identity and Access Management (IAM). Create an IAM user, add the user to an IAM group with administrative permissions, and then grant administrative permissions to the IAM user that you created. You can then access AWS using a special URL and that IAM user's credentials.

If you signed up for AWS, but you haven't created an IAM user for yourself, you can create one using the IAM console.

The exercises in this guide assume that you have a user (`adminuser`) with administrator privileges. Follow the procedure to create `adminuser` in your account.

To create an administrator user and sign in to the console

1. Create an administrator user called `adminuser` in your AWS account. For instructions, see Creating Your First IAM User and Administrators Group in the *IAM User Guide*.
2. A user can sign in to the AWS Management Console using a special URL. For more information, How Users Sign In to Your Account in the *IAM User Guide*.

Important
The Getting Started exercises use the adminuser credentials. For added security, when building and testing production application we recommend you create a service-specific administrator user who has permissions for only the Amazon Polly actions. For an example policy that grants Amazon Polly specific permissions, see Example 1: Allow All Amazon Polly Actions (p. 204).

For more information about IAM, see the following:

- AWS Identity and Access Management (IAM)
- Getting Started
- IAM User Guide

Next Step

Step 2: Getting Started (Console) (p. 5)

Step 2: Getting Started (Console)

The Amazon Polly console is the easiest way to get started testing and using Amazon Polly's speech synthesizing. The Amazon Polly console supports synthesizing speech from either plain text or SSML input.

Topics

- Next Step (p. 7)

Exercise 1: Synthesizing Speech Quick Start (Console)

The Quick Start walks you through the fastest way to test the Amazon Polly speech synthesis for speech quality. When you select the **Text-to-Speech** tab, the text field for entering your text is pre-loaded with example text so you can quickly try out Amazon Polly.

To quickly test Amazon Polly (Console)

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Choose the **Text-to-Speech** tab.
3. Choose **Plain text**.
4. Under **Engine**, choose `Standard` of `Neural`.
5. Choose a language and AWS Region, then choose a voice. If you choose `Neural` for **Engine**, only English US is available for language, and only the Regions where NTTS is available regions are displayed. All Standard voices are disabled.
6. Choose **Listen to speech**.

For more in-depth testing, see the following topics:

- Exercise 2: Synthesizing Speech with Plain Text Input (Console) (p. 6)
- Using SSML (Console) (p. 102)
- Applying Lexicons Using the Console (Synthesize Speech) (p. 130)

Exercise 2: Synthesizing Speech with Plain Text Input (Console)

The following procedure synthesizes speech using plain text input. Note how "W3C" and the date "10/3" (October 3rd) are synthesized.

To synthesize speech using plain text input (console)

1. After logging on to the Amazon Polly console, choose **Get started**, and then choose the **Text-to-Speech** tab.
2. Choose the **Plain text** tab.
3. Type or paste this text into the input box.

```
He was caught up in the game.
In the middle of the 10/3/2014 W3C meeting
he shouted, "Score!" quite loudly.
```

4. For **Engine**, choose `Standard` or `Neural`.
5. Choose a language and AWS Region, then choose a voice. If you choose `Neural` for **Engine**, only English US is available for language, and only Regions where NTTS is available regions are displayed. All Standard voices are disabled.
6. To listen to the speech immediately, choose **Listen to speech**.
7. To save the speech to a file, do one of the following:

a. Choose **Save speech to MP3**.
b. To change to a different file format, choose **Change file format**, choose the file format that you want, and then choose **Change**.

For more in-depth examples, see the following topics:

- Applying Lexicons Using the Console (Synthesize Speech) (p. 130)
- Using SSML (Console) (p. 102)

Next Step

Step 3: Getting Started (AWS CLI) (p. 7)

Step 3: Getting Started (AWS CLI)

You can perform almost all of the Amazon Polly operations that you can perform using the Amazon Polly console using the AWS Command Line Interface (AWS CLI). You can't listen to synthesized speech using the AWS CLI. Instead, you must save it to a file and then open the file in an application that can play it.

Topics

Step 3.1: Set Up the AWS Command Line Interface (AWS CLI)

Follow the steps to download and configure the AWS CLI.

Important
You don't need the AWS CLI to perform the steps in this exercise. However, some of the exercises in this guide use the AWS CLI. You can skip this step and go to Step 3.2: Getting Started Exercise Using the AWS CLI (p. 9), and then set up the AWS CLI later when you need it.

To set up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   - Getting Set Up with the AWS Command Line Interface
   - Configuring the AWS Command Line Interface
2. Add a named profile for the administrator user in the AWS CLI config file. You use this profile when running the AWS CLI commands. For more information about named profiles, see Named Profiles in the *AWS Command Line Interface User Guide*.

```
[profile adminuser]
    aws_access_key_id = adminuser access key ID
    aws_secret_access_key = adminuser secret access key
    region = aws-region
```

For a list of available AWS Regions and those supported by Amazon Polly, see Regions and Endpoints in the *Amazon Web Services General Reference*.

> **Note**
> If you're using the Region supported by Amazon Polly that you specified when you configured the AWS CLI, omit the following line from the AWS CLI code examples.
>
> ```
> --region aws-region
> ```

3. Verify the setup by typing the following help command at the command prompt.

```
aws help
```

A list of valid AWS commands should appear in the AWS CLI window.

To enable Amazon Polly in the AWS CLI (optional)

If you have previously downloaded and configured the AWS CLI, Amazon Polly might not be available unless you reconfigure the AWS CLI. This procedure checks to see if this is necessary and provides instructions if Amazon Polly is not automatically available.

1. Verify the availability of Amazon Polly by typing the following help command at the AWS CLI command prompt.

   ```
   aws polly help
   ```

   If a description of Amazon Polly and a list of valid commands appears in the AWS CLI window, Amazon Polly is available in the AWS CLI and can be used immediately. In this case, you can skip the rest of this procedure. If this is not displayed, continue with Step 2.
2. Use one of the two following options to enable Amazon Polly:

   a. Uninstall and reinstall the AWS CLI.

      For instructions, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

      or
   b. Download the file service-2.json.

      At the command prompt, run the following command.

      ```
      aws configure add-model --service-model file://service-2.json --service-name polly
      ```
3. Reverify the availability of Amazon Polly.

   ```
   aws polly help
   ```

   The description of Amazon Polly should be visible.

Next Step

Step 3.2: Getting Started Exercise Using the AWS CLI (p. 9)

Step 3.2: Getting Started Exercise Using the AWS CLI

Now you can test the speech synthesis offered by Amazon Polly. In this exercise, you call the `SynthesizeSpeech` operation by passing in sample text. You can save the resulting audio as a file and verify its content.

1. Run the `synthesize-speech` AWS CLI command to synthesize sample text to an audio file (`hello.mp3`).

   The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

   ```
   aws polly synthesize-speech \
       --output-format mp3 \
       --voice-id Joanna \
       --text 'Hello, my name is Joanna. I learned about the W3C on 10/3 of last year.' \
       hello.mp3
   ```

   In the call to `synthesize-speech`, you provide sample text for the synthesis, the voice to use (by providing a voice ID, explained in the following step 3), and the output format. The command saves the resulting audio to the `hello.mp3` file.

   In addition to the MP3 file, the operation sends the following output to the console.

   ```
   {
           "ContentType": "audio/mpeg",
           "RequestCharacters": "71"
   }
   ```

2. Play the resulting `hello.mp3` file to verify the synthesized speech.
3. Get the list of available voices by using the `DescribeVoices` operation. Run the following `describe-voices` AWS CLI command.

   ```
   aws polly describe-voices
   ```

   In response, Amazon Polly returns the list of all available voices. For each voice, the response provides the following metadata: voice ID, language code, language name, and the gender of the voice. The following is a sample response.

   ```
   {
       "Voices": [
           {
               "Gender": "Female",
               "Name": "Salli",
               "LanguageName": "US English",
               "Id": "Salli",
               "LanguageCode": "en-US"
           },
           {
               "Gender": "Female",
               "Name": "Joanna",
               "LanguageName": "US English",
               "Id": "Joanna",
               "LanguageCode": "en-US"
           }
       ]
   }
   ```

Optionally, you can specify the language code to find the available voices for a specific language. Amazon Polly supports dozens of voices. The following example lists all the voices for Brazilian Portuguese.

```
aws polly describe-voices \
    --language-code pt-BR
```

For a list of language codes, see Languages Supported by Amazon Polly (p. 17). These language codes are W3C language identification tags (*ISO 639 code for the language name*-*ISO 3166 country code*). For example, en-US (US English), en-GB (British English), and es-ES (Spanish), etc.

You can also use the `help` option in the AWS CLI to get the list of language codes:

```
aws polly describe-voices help
```

Python Examples

This guide provides additional examples, some of which are Python code examples that use AWS SDK for Python (Boto) to make API calls to Amazon Polly. We recommend that you set up Python and test the example code provided in the following section. For additional examples, see Example Applications (p. 155).

Set Up Python and Test an Example (SDK)

To test the Python example code, you need the AWS SDK for Python (Boto). For instruction, see AWS SDK for Python (Boto3).

To test the example Python code

The following Python code example performs the following actions:

- Uses the AWS SDK for Python (Boto) to send a `SynthesizeSpeech` request to Amazon Polly (by providing simple text as input).
- Accesses the resulting audio stream in the response and saves the audio to a file (`speech.mp3`) on your local disk.
- Plays the audio file with the default audio player for your local system.

Save the code to a file (example.py) and run it.

```
"""Getting Started Example for Python 2.7+/3.3+"""
from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError
from contextlib import closing
import os
import sys
import subprocess
from tempfile import gettempdir

# Create a client using the credentials and region defined in the [adminuser]
# section of the AWS credentials file (~/.aws/credentials).
session = Session(profile_name="adminuser")
polly = session.client("polly")
```

```
try:
    # Request speech synthesis
    response = polly.synthesize_speech(Text="Hello world!", OutputFormat="mp3",
                                        VoiceId="Joanna")
except (BotoCoreError, ClientError) as error:
    # The service returned an error, exit gracefully
    print(error)
    sys.exit(-1)

# Access the audio stream from the response
if "AudioStream" in response:
    # Note: Closing the stream is important because the service throttles on the
    # number of parallel connections. Here we are using contextlib.closing to
    # ensure the close method of the stream object will be called automatically
    # at the end of the with statement's scope.
    # with closing(response["AudioStream"]) as stream:
        output = os.path.join(gettempdir(), "speech.mp3")

        try:
            # Open a file for writing the output as a binary stream
            # with open(output, "wb") as file:
                file.write(stream.read())
        except IOError as error:
            # Could not write to file, exit gracefully
            print(error)
            sys.exit(-1)

else:
    # The response didn't contain audio data, exit gracefully
    print("Could not stream audio")
    sys.exit(-1)

# Play the audio using the platform's default player
if sys.platform == "win32":
    os.startfile(output)
else:
    # The following works on macOS and Linux. (Darwin = mac, xdg-open = linux).
    opener = "open" if sys.platform == "darwin" else "xdg-open"
    subprocess.call([opener, output])
```

For additional examples including an example application, see Example Applications (p. 155).

Voices in Amazon Polly

Available Voices

Amazon Polly provides a variety of different voices in multiple languages for synthesizing speech from text.

Language	Name/ID	Gender	Standard Voice	Neural Voice
Arabic (arb)	Zeina	Female	Yes	No
Chinese, Mandarin (cmn-CN)	Zhiyu	Female	Yes	No
Danish (da-DK)	Naja	Female	Yes	No
	Mads	Male	Yes	No
Dutch (nl-NL)	Lotte	Female	Yes	No
	Ruben	Male	Yes	No
English (Australian) (en-AU)	Nicole	Female	Yes	No
	Russell	Male	Yes	No
English (British) (en-GB)	Amy	Female	Yes	Yes
	Emma	Female	Yes	Yes
	Brian	Male	Yes	Yes
English (Indian) (en-IN)	Aditi*	Female	Yes	No
	Raveena	Female	Yes	No
English (US) (en-US)	Ivy	Female (child)	Yes	Yes
	Joanna**	Female	Yes	Yes
	Kendra	Female	Yes	Yes
	Kimberly	Female	Yes	Yes
	Salli	Female	Yes	Yes
	Joey	Male	Yes	Yes
	Justin	Male (child)	Yes	Yes
	Matthew**	Male	Yes	Yes
English (Welsh) (en-GB-WLS)	Geraint	Male	Yes	No

Language	Name/ID	Gender	Standard Voice	Neural Voice
French (fr-FR)	Céline/Celine	Female	Yes	No
	Léa	Female	Yes	No
	Mathieu	Male	Yes	No
French (Canadian) (fr-CA)	Chantal	Female	Yes	No
German (de-DE)	Marlene	Female	Yes	No
	Vicki	Female	Yes	No
	Hans	Male	Yes	No
Hindi (hi-IN)	Aditi*	Female	Yes	No
Icelandic (is-IS)	Dóra/Dora	Female	Yes	No
	Karl	Male	Yes	No
Italian (it-IT)	Carla	Female	Yes	No
	Bianca	Female	Yes	No
	Giorgio	Male	Yes	No
Japanese (ja-JP)	Mizuki	Female	Yes	No
	Takumi	Male	Yes	No
Korean (ko-KR)	Seoyeon	Female	Yes	No
Norwegian (nb-NO)	Liv	Female	Yes	No
Polish (pl-PL)	Ewa	Female	Yes	No
	Maja	Female	Yes	No
	Jacek	Male	Yes	No
	Jan	Male	Yes	No
Portuguese (Brazilian) (pt-BR)	Camila	Female	Yes	Yes
	Vitória/Vitoria	Female	Yes	No
	Ricardo	Male	Yes	No
Portuguese (European) (pt-PT)	Inês/Ines	Female	Yes	No
	Cristiano	Male	Yes	No
Romanian (ro-RO)	Carmen	Female	Yes	No
Russian (ru-RU)	Tatyana	Female	Yes	No
	Maxim	Male	Yes	No

Language	Name/ID	Gender	Standard Voice	Neural Voice
Spanish (European) (es-ES)	Conchita	Female	Yes	No
	Lucia	Female	Yes	No
	Enrique	Male	Yes	No
Spanish (Mexican) (es-MX)	Mia	Female	Yes	No
Spanish (US) (es-US)	Lupe	Female	Yes	Yes
	Penélope/ Penelope	Female	Yes	No
	Miguel	Male	Yes	No
Swedish (sv-SE)	Astrid	Female	Yes	No
Turkish (tr-TR)	Filiz	Female	Yes	No
Welsh (cy-GB)	Gwyneth	Female	Yes	No

* This voice is bilingual and can speak both English and Hindi. For more information, see Bilingual Voices (p. 14).

** These voices can be used with both the Conversational and Newscaster speaking styles when used with the Neural format. For more information, see NTTS Speaking Styles (p. 91).

In addition to the above voices, Amazon Polly can build you a custom Brand Voice that reflects your brand persona, providing you the opportunity to offer unique and exclusive NTTS voices to your customers. To learn more about Amazon Polly Brand Voices, please see https://aws.amazon.com/polly/features/#Brand_Voice.

Bilingual Voices

A bilingual voice like Aditi (Indian English and Hindi) can speak two languages fluently. This gives you the ability to use words and phrases from both languages in a single text using the same voice.

Currently, Aditi is the only bilingual voice available.

Using a Bilingual Voice (Aditi)

Aditi speaks both Indian English (en-IN) and Hindi (hi-IN) fluently. You can synthesize speech in both English and Hindi, and the voice can switch between the two languages even within the same sentence.

Hindi can be used in two different forms:

- Devanagari: "उसेन कहाँ, खेल तोह अब शुर होगा"
- Romanagari (using the Latin alphabet): "Usne kahan, khel toh ab shuru hoga"

Additionally, it's possible to mix English and Hindi of either or both forms within a single sentence:

- Devanagari + English: "This is the song कभी कभी अदिति"
- Romanagari + English: "This is the song from the movie Jaane Tu Ya Jaane Na."

- Devanagari + Romanagari + English: "This is the song कभी कभी अदिति from the movie Jaane Tu Ya Jaane Na."

Because Aditi is a bilingual voice, text in all of these cases will be read correctly, as Amazon Polly can differentiate between the languages and scripts.

Amazon Polly also supports numbers, dates, times, and currency expansion in both English (Arabic numerals) and Hindi (Devanagari numerals). By default, Arabic numerals are read in Indian English. To make Amazon Polly read them in Hindi, you must use the `hi-IN` language code parameter.

Listening to the Voices

You can use the Amazon Polly console to hear a sample from any of the voices available in Amazon Polly

To listen to a voice in Amazon Polly

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Choose the **Text-to-Speech** tab.
3. For **Engine**, choose **Standard** or **Neural**.
4. Choose a language and a Region, then choose a voice.
5. Enter text for the voice to speak or use the default phrase, and then choose **Listen to speech**.

You can choose any of the languages offered by Amazon Polly and the console will display the voices available for that language. In most cases, there will be at least one male and one female voice, often more than one of each. A few only have a single voice. For a complete list, see Voices in Amazon Polly (p. 12)

> **Note**
> The inventory of voices and the number of languages included is continually being updated to include additional choices. To suggest a new language or voice, feel free to provide feedback on this page. Unfortunately, we are not able to comment on plans for specific new languages be they are released.

Each voice is created using native language speakers, so there are variations from voice to voice, even within the same language. When selecting a voice for your project, you should test each of the possible voices with a passage of text to see which best suits your needs.

Voice Speed

Because of the natural variation between voices, each available voice will speak the text at slightly different speeds. For instance, with US English voices, Ivy and Joanna are slightly faster than Matthew when saying "Mary had a little lamb," and considerably faster than Joey.

Since there is so much variation between voices, and the degree of that variation can depend on the text being spoken, no standard speed (words per minute) is available for Amazon Polly voices. However, you can find how long it takes for your voice to say the selected text using SpeechMarks. For more information on using speechmarks in Amazon Polly, see Using Speech Marks (p. 95)

To see approximately how long it takes to speak a text passage

1. Open the AWS CLI.

2. Run the following code, filling in as needed

```
    aws polly synthesize-speech \
         --language-code optional language code if needed
         --output-format json \
         --voice-id [name of desired voice] \
         --text '[desired text]' \
         --speech-mark-types='["viseme"]' \
         LengthOfText.txt
```

3. Open LengthOfText.txt

If the text were "Mary had a little lamb," the last few lines returned by Amazon Polly would be:

```
    {"time":882,"type":"viseme","value":"t"}
    {"time":964,"type":"viseme","value":"a"}
    {"time":1082,"type":"viseme","value":"p"}
```

The last viseme, essentially the sound for the final letters in "lamb" starts 1082 milliseconds after the beginning of the speech. While this is not exactly the length of the audio, it's close and can serve as the basis for comparison between voices.

Changing Your Voice Speed

For certain applications, you may find that you'd prefer the voice you like be slowed down, or speeded up. If the speed of the voice is a concern, Amazon Polly provides the ability to modify this using SSML tags.

For example:

Your organization is making an application that reads books to immigrant audiences. The audience speaks English, but their fluency is limited. In this case, you might consider slowing the rate of speech to give your audience a little more time for comprehension while the application is speaking.

Amazon Polly helps you slow down the rate of speech using the SSML <prosody> tag, as in:

```
<speak>
     In some cases, it might help your audience to <prosody rate="85%">slow
     the speaking rate slightly to aid in comprehension.</prosody>
<speak
```

or

```
<speak>
     In some cases, it might help your audience to <prosody rate="85%">slow
     the speaking rate slightly to aid in comprehension.</prosody>
<speak
```

Two speed options are available to you when using SSML with Amazon Polly:

- Preset speeds: `x-slow`, `slow`, `medium`, `fast`, and `x-fast`. In these cases, the speed of each option is approximate, depending on your preferred voice. The `medium` option is the normal speed of the voice.
- n% of speech rate: any percentage of the speech rate, between 20% and 200% can be used. In these cases, you can choose exactly the speed you want. However, the actual speed of the voice is approximate, depending on the voice you've chosen. 100% is considered to be the normal speed of the voice.

Because the speed of each option is approximate and depends on the voice you choose, we recommend that you test your selected voice at various speeds to see what exactly meets your needs.

For more information on using the `prosody` tag to best effect, see Controlling Volume, Speaking Rate, and Pitch (p. 112)

Languages Supported by Amazon Polly

The following languages are supported by Amazon Polly and can be used to synthesize speech. With each language is the language code. These language codes are W3C language identification tags (*ISO 639-3* for the language name and *ISO 3166* for the country code).

For in-depth tables showing the phonemes and visemes associated with each language, choose the link on each language in the table below.

Language	Language Code
Arabic (arb) (p. 19)	arb
Chinese, Mandarin (cmn-CN) (p. 21)	cmn-CN
Danish (da-DK) (p. 24)	da-DK
Dutch (nl-NL) (p. 26)	nl-NL
English, Australian (en-AU) (p. 29)	en-AU
English, British (en-GB) (p. 34)	en-GB
English, Indian (en-IN) (p. 31)	en-IN
English, Indian (en-IN) (p. 36)	en-US
English, Welsh (en-GB-WSL) (p. 39)	en-GB-WLS
French (fr-FR) (p. 41)	fr-FR
French, Canadian (fr-CA) (p. 43)	fr-CA
Hindi (hi-IN) (p. 48)	hi-IN
German (de-DE) (p. 45)	de-DE
Icelandic (is-IS) (p. 50)	is-IS
Italian (it-IT) (p. 53)	it-IT
Japanese (ja-JP) (p. 55)	ja-JP
Korean (ko-KR) (p. 57)	ko-KR
Norwegian (nb-NO) (p. 58)	nb-NO
Polish (pl-PL) (p. 61)	pl-PL
Portuguese, Brazilian (pt-BR) (p. 65)	pt-BR
Portuguese (pt-PT) (p. 63)	pt-PT
Romanian (ro-RO) (p. 67)	ro-RO

Language	Language Code
Russian (ru-RU) (p. 69)	ru-RU
Spanish (es-ES) (p. 71)	es-ES
Spanish, Mexican (es-MX) (p. 74)	es-MX
Spanish, US (es-US) (p. 76)	es-US
Swedish (sv-SE) (p. 78)	sv-SE
Turkish (tr-TR) (p. 81)	tr-TR
Welsh (cy-GB) (p. 83)	cy-GB

For more information, see Phoneme and Viseme Tables for Supported Languages (p. 18).

Phoneme and Viseme Tables for Supported Languages

The following tables list the phonemes for the languages supported by Amazon Polly, along with examples and the corresponding visemes.

Topics

Arabic (arb)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Arabic voice of Zeina that is supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
ʔ	?	glottal stop	أنا	
ʕ	?\	voiced pharyngeal fricative	عُمَر	k
b	b	voiced bilabial plosive	بَلَد	p
d	d	voiced alveolar plosive	داري	t
dˤ	d_?\	emphatic voiced alveolar plosive	ضَوء	t
d͡ʒ	dZ	voiced postalveolar affricate	جَميل	S
ð	D	voiced dental fricative	ذلِكَ	T
ðˤ	D_?\	emphatic voiced dental fricative	ظَلام	T
f	f	voiceless labiodental fricative	فَصل	f
g	g	voiced velar plosive	إنجلترا	k
ɣ	G	voiced velar fricative	غَرب	k
h	h	voiceless glottal fricative	هذا	k
j	j	palatal approximant	يَمشي	i
k	k	voiceless velar plosive	كَلب	k
l	l	alveolar lateral approximant	لاقى	t

IPA	X-SAMPA	Description	Example	Viseme
lˠ	l_G	emphatic alveolar lateral approximant	الله	t
m	m	bilabial nasal	ماذا	p
n	n	alveolar nasal	نور	t
p	p	voiceless bilabial plosive	حَبس	p
q	q	voiceless uvular plosive	قَريب	k
r	r	alveolar trill	رَمل	r
s	s	voiceless alveolar fricative	سُؤال	s
sˤ	s_?\	emphatic voiceless alveolar fricative	صاحِب	s
ʃ	S	voiceless postalveolar fricative	شُكر	S
t	t	voiceless alveolar plosive	تَمر	t
tˤ	t_?\	emphatic voiceless alveolar plosive	طالِب	t
θ	T	voiceless dental fricative	ثَلاث	T
v	v	voiced labiodental fricative	فيتامين	f
w	w	labio-velar approximant	وَلَد	u
x	x	voiceless velar fricative	خَوْف	k
ħ	X\	voiceless pharyngeal fricative	حَوْلَ	k
z	z	voiced alveolar fricative	زُهور	s
Vowels				
a	a	open front unrounded vowel	بَرد	a
aː	a:	long open front unrounded vowel	دار	a
ɑˤ	A_?\	emphatic open back unrounded vowel	طَبل	a

IPA	X-SAMPA	Description	Example	Viseme
ɑˤː	A_?\:	emphatic long open back unrounded vowel	ظالِم	a
u	u	close back rounded vowel	شُرب	u
uː	u:	long close back rounded vowel	سور	u
uˤ	u_?\	emphatic close back rounded vowel	بُدّ	u
uˤː	u_?\:	emphatic long close back rounded vowel	طول	u
i	i	close front unrounded vowel	بِنت	i
iː	i:	long close front unrounded vowel	حَزين	i
iˤ	i_?\	emphatic close front unrounded vowel	ضِدّ	i
iˤː	i_?\:	emphatic long close front unrounded vowel	ماضي	i
e	e	close-mid front unrounded vowel	ماركت	e
eː	e:	long close-mid front unrounded vowel	موديل	e
ɔ	O	open-mid back rounded vowel	تكنولوجي	O
ɔː	O:	long open-mid back rounded vowel	تليفزيون	O

Chinese, Mandarin (cmn-CN)

The following table lists the Pinyin and International Phonetic Alphabet (IPA) phonemes for the Mandarin Chinese voice that is supported by Amazon Polly. Pinyin is the international standard for Standard Chinese romanization. IPA and X-SAMPA are not commonly used but are available for English support. The IPA and X-SAMPA symbols in the table are for reference only and should not be used for Chinese transcription. Pinyin examples and the corresponding visemes are also shown.

To make Amazon Polly use phonetic pronunciation win Pinyin, use the `phoneme alphabet="x-amazon-`*`phonetic standard used`*`"` tag.

The following examples show this with each standard.

Pinyin:

```
<speak>
```

```
     ## <phoneme alphabet="x-amazon-pinyin" ph="bo2">#</phoneme>#
     ## <phoneme alphabet="x-amazon-pinyin" ph="bao2">#</phoneme>#
</speak>
```

IPA:

```
<speak>
     ## <phoneme alphabet="ipa" ph="p##k##n">pecan</phoneme>#
     ## <phoneme alphabet="ipa" ph="#pi.kæn">pecan</phoneme>#
</speak>
```

X-SAMPA:

```
<speak>
     ## <phoneme alphabet='x-sampa' ph='pI"kA:n'>pecan</phoneme>#
     ## <phoneme alphabet='x-sampa' ph='"pi.k{n'>pecan</phoneme>#
</speak>
```

Note
Amazon Polly accepts Mandarin Chinese input encoded in UTF-8 only. The GB 18030 encoding standard is not currently supported by Amazon Polly.

Phoneme/Viseme Table

Pinyin	IPA	X-SAMPA	Description	Pinyin Example	Viseme
Consonants					
f	f	f	voiceless labiodental fricative	发, **f**a1	f
h	h	h	voiceless glottal fricative	和, **h**e2	k
g	k	k	voiceless velar plosive	古, **g**u3	k
k	kʰ	k_h	aspirated voiceless velar plosive	苦, **k**u3	k
l	l	l	alveolar lateral approximant	拉, **l**a1	t
m	m	m	bilabial nasal	骂, **m**a4	p
n	n	n	alveolar nasal	那, **n**a4	t
ng	ŋ	N	velar nasal	正, zhe**ng**4	k
b	p	p	voiceless bilabial plosive	爸, **b**a4	p
p	pʰ	p_h	aspirated voiceless bilabial plosive	怕, **p**a4	p
s	s	s	voiceless alveolar fricative	四, **s**i4	s
x	ɕ	s\	voiceless alveolo-palatal fricative	西, **x**i1	J
sh	ʂ	s`	voiceless retroflex fricative	是, **sh**i4	S
d	t	t	voiceless alveolar plosive	打, **d**a3	t

Pinyin	IPA	X-SAMPA	Description	Pinyin Example	Viseme
t	tʰ	t_h	aspirated voiceless alveolar plosive	他, **t**a1	t
zh	͡ʈʂ	t`s`	voiceless retroflex affricate	之, **zh**i1	S
ch	͡ʈʂʰ	t`s`_h	aspirated voiceless retroflex affricate	吃, **ch**i1	S
s	͡ts	ts	voiceless alveolar affricate	字, **zi**4	s
j	͡tɕ	ts\	voiceless alveolo-palatal affricate	鸡, **j**i1	J
q	͡tɕʰ	ts_h	aspirated voiceless alveolo-palatal affricate	七, **qi**1	J
c	͡tsʰ	ts_h	aspirated voiceless alveolar affricate	次, **ci**4	s
w	w	w	labio-velar approximant	我, **w**o3	u
r	ʐ	z`	voiced retroflex fricative	日, **r**i4	S
"er" and "r" colored syllables					
er	ɚ	@`	r-coloured mid central vowel	二, **er**4	@
-r			r-colored syllable	馅儿, xian**r**4	@
Vowels					
e	ɤ	7	close-mid back unrounded vowel	恶, **e**4	e
e	ə	@	mid central vowel	恩, **e**n1	@
a	a	a	open front unrounded vowel	安, **a**n1	a
ai	aɪ	aI	diphthong	爱, **ai**4	a
ao	aʊ	aU	diphthong	奥, **ao**4	a
ei	eɪ	e	diphthong	诶, **ei**4	e
e	ɛ	E	open-mid front unrounded vowel	姐, ji**e**3	E
i	i	i	close front unrounded vowel	鸡, j**i**1	i
ou	oʊ	oU	diphthong	欧, **ou**1	o
o	ɔ	O	open-mid back rounded vowel	哦, **o**4	o
u	u	u	close back rounded vowel	主, zh**u**3	u

Pinyin	IPA	X-SAMPA	Description	Pinyin Example	Viseme
yu	y	y	close front rounded vowel	于, **yu**2	u
Tone marks and Additional Symbols					
1			high level tone	淤, yu1	
2			rising tone	鱼, yu2	
3			low (falling-rising) tone	语, yu3	
4			falling tone	育, yu4	
0			neutral tone	的, de0	
-	.	.	syllable boundary	语音 yu3-yin1	

Danish (da-DK)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Danish voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**at	p
d	d	voiced alveolar plosive	**d**a	t
ð	D	voiced dental fricative	ma**d**, **th**riller	T
f	f	voiceless labiodental fricative	**f**at	f
g	g	voiced velar plosive	**g**at	k
h	h	voiceless glottal fricative	**h**at	k
j	j	palatal approximant	**j**o	i
k	k	voiceless velar plosive	**k**at	k
l	l	alveolar lateral approximant	**l**adt	t
m	m	bilabial nasal	**m**at	p
n	n	alveolar nasal	**n**ay	t

IPA	X-SAMPA	Description	Example	Viseme
ŋ	N	velar nasal	la**ng**	k
p	p	voiceless bilabial plosive	**p**ande	p
r	r	alveolar trill	**th**riller, sto**r**y	r
ʁ	R	voiced uvular fricative	**r**at	k
s	s	voiceless alveolar fricative	**s**at	s
t	t	voiceless alveolar plosive	**t**al	t
v	v	voiced labiodental fricative	**v**at	f
w	w	labial-velar approximant	ha**v**, **w**eekend	u
Vowels				
ø	2	close-mid front rounded vowel	**ø**st	o
ø:	2:	long close-mid front rounded vowel	**ø**se	o
ɐ	6	near-open central vowel	m**or**	a
œ	9	open-mid front rounded vowel	sk**ø**n, gr**ø**nt	O
œ:	9:	long open-mid front rounded vowel	h**ø**ne, g**ø**re	O
ə	@	mid central vowel	an**e**	@
æ:	{:	long near-open front unrounded vowel	m**a**le	a
a	a	open front unrounded vowel	m**a**n	a
æ	{	near-open front unrounded vowel	adr**e**sse	a
ɑ	A	open back unrounded vowel	l**a**k, t**a**k	a
ɑ:	A:	long open back unrounded vowel	r**a**se	a
e	e	close-mid front unrounded vowel	m**i**dt	e

IPA	X-SAMPA	Description	Example	Viseme
eː	e:	long close-mid front unrounded vowel	m**e**le	e
ɛ	E	open-mid front unrounded vowel	m**æ**t	E
ɛː	E:	long open-mid front unrounded vowel	m**æ**le	E
i	i	close front unrounded vowel	m**i**t	i
iː	i:	long close front unrounded vowel	m**i**le	i
o	o	close-mid back rounded vowel	f**oto**	o
oː	o:	long close-mid back rounded vowel	m**o**le	o
ɔ	O	open-mid back rounded vowel	m**u**nd	O
ɔː	O:	long open-mid back rounded vowel	m**å**le	O
ɒː	Q:	long open back rounded vowel	m**o**rse	O
u	u	close back rounded vowel	l**u**sk	u
uː	u:	long close back rounded vowel	m**u**le	u
ʌ	V	open-mid back unrounded	kør**er**	E
y	y	close front rounded vowel	**y**t	u
yː	y:	long close front rounded vowel	h**y**le	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Dutch (nl-NL)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Dutch voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**ak	p
d	d	voiced alveolar plosive	**d**ak	t
d͡ʒ	dZ	voiced postalveolar affricate	mana**g**er	S
f	f	voiceless labiodental fricative	**f**el	f
g	g	voiced velar plosive	**g**oal	k
ɣ	G	voiced velar fricative	**h**oed	k
ɦ	h\	voiced glottal fricative	**h**and	k
j	j	palatal approximant	**j**a	i
k	k	voiceless velar plosive	**k**ap	k
l	l	alveolar lateral approximant	**l**and	t
m	m	bilabial nasal	**m**et	p
n	n	alveolar nasal	**n**et	t
ŋ	N	velar nasal	ba**ng**	k
p	p	voiceless bilabial plosive	**p**ak	p
r	r	alveolar trill	**r**and	r
s	s	voiceless alveolar fricative	**s**ein	s
ʃ	S	voiceless postalveolar fricative	**sh**ow	S
t	t	voiceless alveolar plosive	**t**ak	t
v	v	voiced labiodental fricative	**v**el	f
ʋ	v\	labiodental approximant	**w**it	f
x	x	voiceless velar fricative	to**ch**	k

IPA	X-SAMPA	Description	Example	Viseme
z	z	voiced alveolar fricative	**z**iin	s
ʒ	Z	voiced postalveolar fricative	baga**g**e	S
Vowels				
øː	2:	long close-mid front rounded vowel	n**eu**s	o
œy	9y	diphthong	b**ui**t	O
ə	@	mid central vowel	d**e**	@
aː	a:	long open front unrounded vowel	b**aa**d	a
ɑː	A	open back unrounded vowel	b**a**d	a
eː	e:	long close-mid front unrounded vowel	b**ee**t	e
ɜː	3:	long open-mid central unrounded vowel	barri**è**re	E
ɛ	E	open-mid front unrounded vowel	b**e**d	E
ɛi	Ei	diphthong	b**ee**t	E
i	i	close front unrounded vowel	v**ie**r	i
ɪ	I	near-close near-front unrounded vowel	p**i**t	i
oː	o:	long close-mid back rounded vowel	b**oo**t	o
ɔ	O	open-mid back rounded vowel	p**o**t	O
u	u	close back rounded vowel	h**oe**d	u
ʌu	Vu	diphthong	f**ou**t	E
yː	y:	long close front rounded vowel	f**uu**t	u
ʏ	Y	near-close near-front rounded vowel	h**u**t	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	

IPA	X-SAMPA	Description	Example	Viseme
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

English, Australian (en-AU)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Australian English voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**ed	p
d	d	voiced alveolar plosive	**d**ig	t
d͡ʒ	dZ	voiced postalveolar affricate	**j**ump	S
ð	D	voiced dental fricative	**th**en	T
f	f	voiceless labiodental fricative	**f**ive	f
g	g	voiced velar plosive	**g**ame	k
h	h	voiceless glottal fricative	**h**ouse	k
j	j	palatal approximant	**y**es	i
k	k	voiceless velar plosive	**c**at	k
l	l	alveolar lateral approximant	**l**ay	t
l̩	l=	syllabic alveolar lateral approximant	batt**le**	t
m	m	bilabial nasal	**m**ouse	p
m̩	m=	syllabic bilabial nasal	anth**em**	p
n	n	alveolar nasal	**n**ap	t
n̩	n=	syllabic alveolar nasal	**n**ap	t
ŋ	N	velar nasal	thi**ng**	k

IPA	X-SAMPA	Description	Example	Viseme
p	p	voiceless bilabial plosive	**p**in	p
ɹ	r\	alveolar approximant	**r**ed	r
s	s	voiceless alveolar fricative	**s**eem	s
ʃ	S	voiceless postalveolar fricative	**sh**ip	S
t	t	voiceless alveolar plosive	**t**ask	t
t͡ʃ	tS	voiceless postalveolar affricate	**ch**art	S
Θ	T	voiceless dental fricative	**th**in	T
v	v	voiced labiodental fricative	**v**est	f
w	w	labial-velar approximant	**w**est	u
z	z	voiced alveolar fricative	**z**ero	s
ʒ	Z	voiced postalveolar fricative	vi**s**ion	S
Vowels				
ə	@	mid central vowel	**a**ren**a**	@
əʊ	@U	diphthong	g**oa**t	@
æ	{	near open-front unrounded vowel	tr**a**p	a
aɪ	aI	diphthong	pr**i**ce	a
aʊ	aU	diphthong	m**ou**th	a
ɑː	A:	long open-back unrounded vowel	f**a**ther	a
eɪ	eI	diphthong	f**a**ce	e
ɜː	3:	long open mid-central unrounded vowel	n**ur**se	E
ɛ	E	open mid-front unrounded vowel	dr**e**ss	E
ɛə	E@	diphthong	squ**are**	E

IPA	X-SAMPA	Description	Example	Viseme
iː	i	long close front unrounded vowel	fl**ee**ce	i
ɪ	I	near-close near-front unrounded vowel	k**i**t	i
ɪə	I@	diphthong	**near**	i
ɔː	OI	long open-mid back rounded vowel	th**ou**ght	O
ɔɪ	OI	Diphthong	ch**oi**ce	O
ɒ	Q	open back rounded vowel	l**o**t	O
uː	u:	long close-back rounded vowel	g**oo**se	u
ʊ	U	near-close near-back rounded vowel	f**oo**t	u
ʊə	U@	diphthong	c**ure**	u
ʌ	V	Open-mid-back unrounded vowel	str**u**t	E
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

English, Indian (en-IN)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Indian English voice supported by Amazon Polly.

For additional phonemes used in conjunction with Indian English, see Hindi (hi-IN) (p. 48).

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**ed	p
d	d	voiced alveolar plosive	**d**ig	t
d͡ʒ	dZ	voiced postalveolar affricate	**j**ump	S

IPA	X-SAMPA	Description	Example	Viseme
ð	D	voiced dental fricative	**th**en	T
f	f	voiceless labiodental fricative	**f**ive	f
g	g	voiced velar plosive	**g**ame	k
h	h	voiceless glottal fricative	**h**ouse	k
j	j	palatal approximant	**y**es	i
k	k	voiceless velar plosive	**c**at	k
l	l	alveolar lateral approximant	**l**ay	t
l̩	l=	syllabic alveolar lateral approximant	batt**le**	t
m	m	bilabial nasal	**m**ouse	p
m̩	m=	syllabic bilabial nasal	anth**em**	p
n	n	alveolar nasal	**n**ap	t
n̩	n=	syllabic alveolar nasal	**n**ap	t
ŋ	N	velar nasal	thi**ng**	k
p	p	voiceless bilabial plosive	**p**in	p
ɹ	r\	alveolar approximant	**r**ed	r
s	s	voiceless alveolar fricative	**s**eem	s
ʃ	S	voiceless postalveolar fricative	**sh**ip	S
t	t	voiceless alveolar plosive	**t**ask	t
t͡ʃ	tS	voiceless postalveolar affricate	**ch**art	S
Θ	T	voiceless dental fricative	**th**in	T
v	v	voiced labiodental fricative	**v**est	f
w	w	labial-velar approximant	**w**est	u

IPA	X-SAMPA	Description	Example	Viseme
z	z	voiced alveolar fricative	**z**ero	s
ʒ	Z	voiced postalveolar fricative	vi**s**ion	S
Vowels				
ə	@	mid central vowel	**a**ren**a**	@
əʊ	@U	diphthong	g**oa**t	@
æ	{	near open-front unrounded vowel	tr**a**p	a
aɪ	aI	diphthong	pr**i**ce	a
aʊ	aU	diphthong	m**ou**th	a
ɑː	A:	long open-back unrounded vowel	f**a**ther	a
eɪ	eI	diphthong	f**a**ce	e
ɜː	3:	long open mid-central unrounded vowel	n**ur**se	E
ɛ	E	open mid-front unrounded vowel	dr**e**ss	E
ɛə	E@	diphthong	squ**are**	E
iː	i	long close front unrounded vowel	fl**ee**ce	i
ɪ	I	near-close near-front unrounded vowel	k**i**t	i
ɪə	I@	diphthong	n**ear**	i
ɔː	OI	long open-mid back rounded vowel	th**ou**ght	O
ɔɪ	OI	Diphthong	ch**oi**ce	O
ɒ	Q	open back rounded vowel	l**o**t	O
uː	u:	long close-back rounded vowel	g**oo**se	u
ʊ	U	near-close near-back rounded vowel	f**oo**t	u
ʊə	U@	diphthong	c**u**re	u
ʌ	V	Open-mid-back unrounded vowel	str**u**t	E

IPA	X-SAMPA	Description	Example	Viseme
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

English, British (en-GB)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the British English voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**ed	p
d	d	voiced alveolar plosive	**d**ig	t
d͡ʒ	dZ	voiced postalveolar affricate	**j**ump	S
ð	D	voiced dental fricative	**th**en	T
f	f	voiceless labiodental fricative	**f**ive	f
g	g	voiced velar plosive	**g**ame	k
h	h	voiceless glottal fricative	**h**ouse	k
j	j	palatal approximant	**y**es	i
k	k	voiceless velar plosive	**c**at	k
l	l	alveolar lateral approximant	**l**ay	t
l̩	l=	syllabic alveolar lateral approximant	batt**le**	t
m	m	bilabial nasal	**m**ouse	p
m̩	m=	syllabic bilabial nasal	anth**em**	p
n	n	alveolar nasal	**n**ap	t

IPA	X-SAMPA	Description	Example	Viseme
n̩	n=	syllabic alveolar nasal	butto**n**	t
ŋ	N	velar nasal	thi**ng**	k
p	p	voiceless bilabial plosive	**p**in	p
ɹ	r\	alveolar approximant	**r**ed	r
s	s	voiceless alveolar fricative	**s**eem	s
ʃ	S	voiceless postalveolar fricative	**sh**ip	S
t	t	voiceless alveolar plosive	**t**ask	t
t͡ʃ	tS	voiceless postalveolar affricate	**ch**art	S
Θ	T	voiceless dental fricative	**th**in	T
v	v	voiced labiodental fricative	**v**est	f
w	w	labial-velar approximant	**w**est	u
z	z	voiced alveolar fricative	**z**ero	s
ʒ	Z	voiced postalveolar fricative	vi**s**ion	S
Vowels				
ə	@	mid central vowel	**a**ren**a**	@
əʊ	@U	diphthong	g**oa**t	@
æ	{	near open-front unrounded vowel	tr**a**p	a
aɪ	aI	diphthong	pr**i**ce	a
aʊ	aU	diphthong	m**ou**th	a
ɑː	A:	long open-back unrounded vowel	f**a**ther	a
eɪ	eI	diphthong	f**a**ce	e
ɜː	3:	long open mid-central unrounded vowel	n**ur**se	E

IPA	X-SAMPA	Description	Example	Viseme
ɛ	E	open mid-front unrounded vowel	dr**e**ss	E
ɛə	E@	diphthong	squ**are**	E
iː	i	long close front unrounded vowel	fl**ee**ce	i
ɪ	I	near-close near-front unrounded vowel	k**i**t	i
ɪə	I@	diphthong	n**ear**	i
ɔː	O:	long open-mid back rounded vowel	th**ough**t	O
ɔɪ	OI	Diphthong	ch**oi**ce	O
ɒ	Q	open back rounded vowel	l**o**t	O
uː	u:	long close-back rounded vowel	g**oo**se	u
ʊ	U	near-close near-back rounded vowel	f**oo**t	u
ʊə	U@	diphthong	c**ure**	u
ʌ	V	Open-mid-back unrounded vowel	str**u**t	E
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

English, American (en-US)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the American English voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**ed	p
d	d	voiced alveolar plosive	**d**ig	t

IPA	X-SAMPA	Description	Example	Viseme
d͡ʒ	dZ	voiced postalveolar affricate	**j**ump	S
ð	D	voiced dental fricative	**th**en	T
f	f	voiceless labiodental fricative	**f**ive	f
g	g	voiced velar plosive	**g**ame	k
h	h	voiceless glottal fricative	**h**ouse	k
j	j	palatal approximant	**y**es	i
k	k	voiceless velar plosive	**c**at	k
l	l	alveolar lateral approximant	**l**ay	t
m	m	bilabial nasal	**m**ouse	p
n	n	alveolar nasal	**n**ap	t
ŋ	N	velar nasal	thi**ng**	k
p	p	voiceless bilabial plosive	s**p**eak	p
ɹ	r\	alveolar approximant	**r**ed	r
s	s	voiceless alveolar fricative	**s**eem	s
ʃ	S	voiceless postalveolar fricative	**sh**ip	S
t	t	voiceless alveolar plosive	**t**rap	t
t͡ʃ	tS	voiceless postalveolar affricate	**ch**art	S
θ	T	voiceless dental fricative	**th**in	T
v	v	voiced labiodental fricative	**v**est	f
w	w	labial-velar approximant	**w**est	u
z	z	voiced alveolar fricative	**z**ero	s
ʒ	Z	voiced postalveolar fricative	vi**s**ion	S

IPA	X-SAMPA	Description	Example	Viseme
Vowels				
ə	@	mid-central vowel	**a**ren**a**	@
ɚ	@`	mid-central r-colored vowel	read**er**	@
æ	{	near open-front unrounded vowel	tr**a**p	a
aɪ	aI	diphthong	pr**i**ce	a
aʊ	aU	diphthong	m**ou**th	a
ɑ	A	long open-back unrounded vowel	f**a**ther	a
eɪ	eI	diphthong	f**a**ce	e
ɝ	3`	open mid-central unrounded r-colored vowel	n**ur**se	E
ɛ	E	open mid-front unrounded vowel	dr**e**ss	E
i	i	long close front unrounded vowel	fl**ee**ce	i
ɪ	I	near-close near-front unrounded vowel	k**i**t	i
oʊ	oU	diphthong	g**oa**t	o
ɔ	O	long open mid-back rounded vowel	th**ou**ght	O
ɔɪ	OI	diphthong	ch**oi**ce	O
u	u	long close-back rounded vowel	g**oo**se	u
ʊ	U	near-close near-back rounded vowel	f**oo**t	u
ʌ	V	open-mid-back unrounded vowel	str**u**t	E
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

English, Welsh (en-GB-WSL)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Welsh English voice supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**ed	p
d	d	voiced alveolar plosive	**d**ig	t
d͡ʒ	dZ	voiced postalveolar affricate	**j**ump	S
ð	D	voiced dental fricative	**th**en	T
f	f	voiceless labiodental fricative	**f**ive	f
g	g	voiced velar plosive	**g**ame	k
h	h	voiceless glottal fricative	**h**ouse	k
j	j	palatal approximant	**y**es	i
k	k	voiceless velar plosive	**c**at	k
l	l	alveolar lateral approximant	**l**ay	t
l̩	l=	syllabic alveolar lateral approximant	batt**le**	t
m	m	bilabial nasal	**m**ouse	p
m̩	m=	syllabic bilabial nasal	anth**em**	p
n	n	alveolar nasal	**n**ap	t
n̩	n=	syllabic alveolar nasal	**n**ap	t
ŋ	N	velar nasal	thi**ng**	k
p	p	voiceless bilabial plosive	**p**in	p
ɹ	r\	alveolar approximant	**r**ed	r
s	s	voiceless alveolar fricative	**s**eem	s

IPA	X-SAMPA	Description	Example	Viseme
ʃ	S	voiceless postalveolar fricative	**sh**ip	S
t	t	voiceless alveolar plosive	**t**ask	t
t͡ʃ	tS	voiceless postalveolar affricate	**ch**art	S
Θ	T	voiceless dental fricative	**th**in	T
v	v	voiced labiodental fricative	**v**est	f
w	w	labial-velar approximant	**w**est	u
z	z	voiced alveolar fricative	**z**ero	s
ʒ	Z	voiced postalveolar fricative	vi**s**ion	S
Vowels				
ə	@	mid central vowel	**a**ren**a**	@
əʊ	@U	diphthong	g**oa**t	@
æ	{	near open-front unrounded vowel	tr**a**p	a
aɪ	aI	diphthong	pr**i**ce	a
aʊ	aU	diphthong	m**ou**th	a
ɑː	A:	long open-back unrounded vowel	f**a**ther	a
eɪ	eI	diphthong	f**a**ce	e
ɜː	3:	long open mid-central unrounded vowel	n**ur**se	E
ɛ	E	open mid-front unrounded vowel	dr**e**ss	E
ɛə	E@	diphthong	squ**are**	E
iː	i	long close front unrounded vowel	fl**ee**ce	i
ɪ	I	near-close near-front unrounded vowel	k**i**t	i
ɪə	I@	diphthong	n**ear**	i

IPA	X-SAMPA	Description	Example	Viseme
ɔ:	OI	long open-mid back rounded vowel	th**ou**ght	O
ɔɪ	OI	Diphthong	ch**oi**ce	O
ɒ	Q	open back rounded vowel	l**o**t	O
u:	u:	long close-back rounded vowel	g**oo**se	u
ʊ	U	near-close near-back rounded vowel	f**oo**t	u
ʊə	U@	diphthong	c**u**re	u
ʌ	V	Open-mid-back unrounded vowel	str**u**t	E
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

French (fr-FR)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the French voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**oire	p
d	d	voiced alveolar plosive	ma**d**ame	t
f	f	voiceless labiodental fricative	**f**emme	f
g	g	voiced velar plosive	**g**rand	k
ɥ	H	labial-palatal approximant	br**u**it	u
j	j	palatal approximant	mei**ll**eur	i
k	k	voiceless velar plosive	**q**uatre	k

IPA	X-SAMPA	Description	Example	Viseme
l	l	alveolar lateral approximant	ma**l**ade	t
m	m	bilabial nasal	**m**aison	p
n	n	alveolar nasal	astro**n**ome	t
ɲ	J	palatal nasal	bai**gn**er	J
ŋ	N	velar nasal	park**ing**	k
p	p	voiceless bilabial plosive	**p**omme	p
ʁ	R	voiced uvular fricative	amou**r**eux	k
s	s	voiceless alveolar fricative	**s**anté	s
ʃ	S	voiceless postalveolar fricative	**ch**at	S
t	t	voiceless alveolar plosive	**t**éléphone	t
v	v	voiced labiodental fricative	**v**rai	f
w	w	labial-velar approximant	s**o**ir	u
z	z	voiced alveolar fricative	rai**s**on	s
ʒ	Z	voiced postalveolar fricative	auber**g**ine	S
Vowels				
ø	2	close-mid front rounded vowel	d**eu**x	o
œ	9	open-mid front rounded vowel	n**eu**f	O
œ̃	9~	nasal open-mid front rounded vowel	br**un**	O
ə	@	mid central vowel	**je**	@
a	a	open front unrounded vowel	t**a**ble	a
ɑ̃	A~	nasal open back unrounded vowel	cam**em**bert	a
e	e	close-mid front unrounded vowel	march**é**	e

IPA	X-SAMPA	Description	Example	Viseme
ɛ	E	open-mid front unrounded vowel	n**ei**ge	E
ɛ̃	E~	nasal open-mid front unrounded vowel	sap**in**	E
i	i	close front unrounded vowel	m**i**lle	i
o	o	close-mid back rounded vowel	h**ô**pital	o
ɔ	O	open-mid back rounded vowel	h**o**mme	O
ɔ̃	O~	nasal open-mid back rounded vowel	b**on**	O
u	u	close back rounded vowel	s**ou**s	u
y	y	close front rounded vowel	d**u**r	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

French, Canadian (fr-CA)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the French Canadian voice supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**oire	p
d	d	voiced alveolar plosive	ma**d**ame	t
f	f	voiceless labiodental fricative	**f**emme	f
g	g	voiced velar plosive	**g**rand	k
ɥ	H	labial-palatal approximant	br**u**it	u

IPA	X-SAMPA	Description	Example	Viseme
j	j	palatal approximant	mei**ll**eur	i
k	k	voiceless velar plosive	**q**uatre	k
l	l	alveolar lateral approximant	ma**l**ade	t
m	m	bilabial nasal	**m**aison	p
n	n	alveolar nasal	astro**n**ome	t
ɲ	J	palatal nasal	bai**gn**er	J
ŋ	N	velar nasal	park**ing**	k
p	p	voiceless bilabial plosive	**p**omme	p
ʁ	R	voiced uvular fricative	amou**r**eux	k
s	s	voiceless alveolar fricative	**s**anté	s
ʃ	S	voiceless postalveolar fricative	**ch**at	S
t	t	voiceless alveolar plosive	**t**éléphone	t
v	v	voiced labiodental fricative	**v**rai	f
w	w	labial-velar approximant	s**o**ir	u
z	z	voiced alveolar fricative	rai**s**on	s
ʒ	Z	voiced postalveolar fricative	auber**g**ine	S
Vowels				
ø	2	close-mid front rounded vowel	d**eu**x	o
œ	9	open-mid front rounded vowel	n**eu**f	O
œ̃	9~	nasal open-mid front rounded vowel	br**un**	O
ə	@	mid central vowel	**je**	@
a	a	open front unrounded vowel	t**a**ble	a

IPA	X-SAMPA	Description	Example	Viseme
ɑ̃	A~	nasal open back unrounded vowel	cam**em**bert	a
e	e	close-mid front unrounded vowel	march**é**	e
ɛ	E	open-mid front unrounded vowel	n**ei**ge	E
ɛ̃	E~	nasal open-mid front unrounded vowel	sap**in**	E
i	i	close front unrounded vowel	m**i**lle	i
o	o	close-mid back rounded vowel	h**ô**pital	o
ɔ	O	open-mid back rounded vowel	h**o**mme	O
ɔ̃	O~	nasal open-mid back rounded vowel	b**on**	O
u	u	close back rounded vowel	s**ou**s	u
y	y	close front rounded vowel	d**u**r	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

German (de-DE)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the German voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
ʔ	?	glottal stop		
b	b	voiced bilabial plosive	**B**ier	p
d	d	voiced alveolar plosive	**D**ach	t

IPA	X-SAMPA	Description	Example	Viseme
ç	C	voiceless palatal fricative	i**ch**	k
d͡ʒ	dZ	voiced postalveolar affricate	**Dsch**ungel	S
f	f	Voiceless labiodental fricative	**V**ogel	f
g	g	Voiced velar plosive	**G**abel	k
h	h	Voiceless glottal fricative	**H**aus	k
j	j	Voiceless glottal fricative	**j**emand	i
k	k	Voiceless velar plosive	**K**leid	k
l	l	Alveolar lateral approximant	**L**och	t
m	m	Bilabial nasal	**M**ilch	p
n	n	Alveolar nasal	**N**atur	t
ŋ	N	Velar nasal	kli**ng**en	k
p	p	Voiceless bilabial plosive	**P**ark	p
p͡f	pf	Voiceless labiodental affricate	A**pf**el	
ʀ	R	Uvular trill	**R**egen	
s	s	voiceless alveolar fricative	Me**ss**er	s
ʃ	S	Voiceless postalveolar fricative	Fi**sch**er	S
t	t	Voiceless alveolar plosive	**T**opf	T
t͡s	Ts	Voiceless alveolar affricate	**Z**ahl	
t͡ʃ	tS	Voiceless postalveolar affricate	deu**tsch**	S
v	v	Voiced labiodental fricative	**W**asser	f
x	x	Voiceless velar fricative	ko**ch**en	k

IPA	X-SAMPA	Description	Example	Viseme
z	z	Voiced alveolar fricative	**S**ee	s
ʒ	Z	Voiced postalveolar fricative	Oran**g**e	S
Vowels				
øː	2:	long close-mid front rounded vowel	b**ö**se	o
ɐ	6	near-open central vowel	bess**er**	a
ɐ̯	6_^	non-syllabic near-open central vowel	Kl**ar**	a
œ	9	open-mid front rounded vowel	k**ö**nnen	O
ə	@	mid central vowel	Red**e**	@
a	a	open front unrounded vowel	S**a**lz	a
aː	a:	long open front unrounded vowel	S**ah**ne	a
aɪ	aI	diphthong	n**ei**n	a
aʊ	aU	diphthong	**Au**gen	a
ɑ̃	A~	nasal open back unrounded vowel	Restaur**ant**	a
eː	e:	long close-mid front unrounded vowel	R**e**de	e
ɛ	E	open-mid front unrounded vowel	K**e**ller	E
ɛ̃	E~	nasal open-mid front unrounded vowel	Terr**ain**	E
iː	i:	long close front unrounded vowel	L**ie**d	i
ɪ	I	near-close near-front unrounded vowel	b**i**tte	i
oː	o:	long close-mid back rounded vowel	K**oh**l	o
ɔ	O	open-mid back rounded vowel	K**o**ffer	O
ɔ̃	O~	nasal open-mid back rounded vowel	Ann**on**ce	O

IPA	X-SAMPA	Description	Example	Viseme
ɔʏ	OY	diphthong	n**eu**	O
u:	u:	long close back rounded vowel	Br**u**der	u
ʊ	U	near-close near-back rounded vowel	W**u**nder	u
y:	y:	long close front rounded vowel	k**üh**l	u
ʏ	Y	near-close near-front rounded vowel	K**ü**che	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Hindi (hi-IN)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the phoneme's sound type for the Hindi voices that are supported by Amazon Polly.

For additional phonemes used in conjunction with Hindi, see English, Indian (en-IN) (p. 31).

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example
Consonants			
pʰ	p_h	voiceless aspirated bilabial plosive	फूल (phool)
bʱ	b_h	voiced aspirated bilabial plosive	भारी (bhaari)
t̪	t_d	voiceless dental plosive	तापमान (taapmaan)
t̪ʰ	t_d_h	voiceless aspirated dental plosive	थोड़ा (thoda)
d̪	d_d	voiced dental plosive	दल्ली (dilli)
d̪ʱ	d_d_h	voiced aspirated dental plosive	धोबी (dhobi)
ʈ	t`	voiceless retroflex plosive	कटोरा (katora)
ʈʰ	t`_h	voiceless aspirated retroflex plosive	ठंड (thand)
ɖ	d`	voiced retroflex plosive	डर (darr)

IPA	X-SAMPA	Description	Example
ɖʱ	d`_h	voiced aspirated retroflex plosive	ढाल (dhal)
tʃʱ	tS_h	voiceless aspirated palatal affricate	छाल (chaal)
dʒʱ	dZ_h	voiced aspirated palatal affricate	झाल (jhaal)
kʱ	k_h	voiceless aspirated velar plosive	खान (khan)
gʱ	g_h	voiced aspirated velar plosive	घान (ghaan)
ɳ	n`	retroflex nasal	कृषण (kshan)
ɾ	4	alveolar flap	राम (ram)
ɽ	r`	plain retroflex flap	बड़ा (bada)
ɽʱ	r`_h	voiced aspirated retroflex flap	बढ़ी (barhi)
ʋ	v\	bilabial approximant	वसूल (wasool)
Vowels			
ə	@_o	mid central vowel	अच्छा (achhaa)
ə̃	@~	nasalised mid central vowel	हँसना (hansnaa)
a	A_o	open front unrounded vowel	आग (aag)
a˜	A~	nasalised open front unrounded vowel	घ़डयिाँ (ghariyaan)
ɪ	I_o	near-close near-front unrounded vowel	इक्कीस (ikkees)
ɪ̃	I~	nasalised near-close near front unrounded vowel	संचिाई (sinchai)
i	i_o	close front unrounded vowel	बल्ल्ली (billee)
ı˜	i~	nasalised close front unrounded vowel	नहीं (nahin)
ʊ	U_o	near-close near-back rounded vowel	उलूल (ullu)
ʊ̃	U~	nasalised near-close near-back rounded vowel	मुँह (munh)
u	u_o	close back rounded vowel	फूल (phool)

IPA	X-SAMPA	Description	Example
ũ	u~	nasalised close back rounded vowel	ऊँट (oont)
ɔ	O_o	open-mid back rounded vowel	कौन (kaun)
ɔ̃	O~	nasalised open-mid back rounded vowel	भौ (bhaun)
o	o	close-mid back rounded vowel	सोना (sona)
õ	o~	nasalised close-mid back rounded vowel	क्यों (kyon)
ɛ	E_o	open-mid front unrounded vowel	पैसा (paisa)
ɛ̃	E~	nasalised open-mid front unrounded vowel	मैं (main)
e	e	close-mid front unrounded vowel	एक (ek)
ẽ	e~	nasalised close-mid front unrounded vowel	किताबें (kitabein)

Icelandic (is-IS)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Icelandic voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	gras**b**akkanum	0
c	c	voiceless palatal plosive	pak**k**in	k
cʰ	c_h	aspirated voiceless palatal plosive	anar**k**istai	k
ç	C	voiceless palatal fricative	**h**éðan	k
d	d	voiced alveolar plosive	bón**d**i	t
ð	D	voiced dental fricative	bor**ð**	T

IPA	X-SAMPA	Description	Example	Viseme
f	f	voiceless labiodental fricative	du**f**t	f
g	g	voiced velar plosive	hol**g**óma	k
ɣ	G	voiced velar fricative	hu**g**ur	k
h	h	voiceless glottal fricative	**h**eili	k
j	j	palatal approximant	**j**ökull	i
kʰ	k_h	aspirated voiceless velar plosive	ós**k**öpunum	k
l	l	alveolar lateral approximant	gó**l**f	t
l̥	l_0	voiceless alveolar lateral approximant	fó**l**k	t
m	m	bilabial nasal	septe**m**ber	p
m̥	m_0	voiceless bilabial nasal	ko**m**pa	p
n	n	alveolar nasal	**n**úmer	t
n̥	n_0	voiceless alveolar nasal	pö**n**tun	t
ɲ	J	palatal nasal	pæli**n**gar	J
ŋ	N	velar nasal	sö**n**gvarann	k
ŋ̊	N_0	voiceless velar nasal	fræ**n**ka	k
pʰ	p_h	aspirated voiceless bilabial plosive	af**p**lánun	p
r	r	alveolar trill	afsk**r**ifta	r
r̥	r_0	voiceless alveolar trill	andvö**r**pum	r
s	s	voiceless alveolar fricative	baðhú**s**	s
tʰ	t_h	aspirated voiceless alveolar plosive	**t**anki	t
θ	T	voiceless dental fricative	**þ**eldökki	T
v	v	voiced labiodental fricative	sil**f**ur	f
w	w	labial-velar approximant		u

IPA	X-SAMPA	Description	Example	Viseme
x	x	voiceless velar fricative	samféla**g**s	k
Vowels				
œ	9	open-mid front rounded vowel	þr**ö**skuldinum	O
œː	9:	long open-mid front rounded vowel	tv**ö**	O
a	a	open front unrounded vowel	nefn**a**	a
aː	a:	long open front unrounded vowel	f**a**ra	a
au	au	diphthong	**á**tta	a
auː	au:	diphthong	**á**tján	a
ɛ	E	open-mid front unrounded vowel	k**e**nnari	E
ɛː	E:	long open-mid front unrounded vowel	dr**e**ka	E
i	i	close front unrounded vowel	Gúl**í**ver	i
iː	i:	long close front unrounded vowel	þr**í**r	i
ɪ	I	near-close near-front unrounded vowel	samsp**i**l	i
ɪː	I:	long near-close near-front unrounded vowel	st**i**g	i
ɔ	O	open-mid back rounded vowel	regndr**o**par	O
ɔː	O:	long open-mid back rounded vowel	ullarb**o**lur	O
ɔu	Ou	diphthong	t**ó**lf	O
ɔuː	Ou:	diphthong	fj**ó**rir	O
u	u	close back rounded vowel	st**ú**lkan	u
uː	u:	long close back rounded vowel	fr**ú**	u
ʏ	Y	near-close near-front rounded vowel	tíu	u

IPA	X-SAMPA	Description	Example	Viseme
ʏː	Y	long near-close near-front rounded vowel	gr**u**ninn	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Italian (it-IT)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Italian voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**acca	p
d	d	voiced alveolar plosive	**d**ama	t
d͡z	dz	voiced alveolar affricate	**z**ero	s
d͡ʒ	dZ	voiced postalveolar affricate	**g**iro	S
f	f	voiceless labiodental fricative	**f**amiglia	f
g	g	voiced velar plosive	**g**atto	k
h	h	voiceless glottal fricative	**h**orror	k
j	j	palatal approximant	d**i**eci	i
k	k	voiceless velar plosive	**c**ampo	k
l	l	alveolar lateral approximant	**l**ido	t
ʎ	L	palatal lateral approximant	a**gli**o	J
m	m	bilabial nasal	**m**ille	p
n	n	alveolar nasal	**n**ove	t

IPA	X-SAMPA	Description	Example	Viseme
ɲ	J	palatal nasal	lasa**gn**e	J
p	p	voiceless bilabial plosive	**p**izza	p
r	r	alveolar trill	**r**isata	r
s	s	voiceless alveolar fricative	**s**ei	s
ʃ	S	voiceless postalveolar fricative	**sci**enza	S
t	t	voiceless alveolar plosive	**t**avola	t
t͡s	ts	voiceless alveolar affricate	for**z**a	s
t͡ʃ	tS	voiceless postalveolar affricate	**ci**elo	S
v	v	voiced labiodental fricative	**v**enti	f
w	w	labial-velar approximant	q**u**attro	u
z	z	voiced alveolar fricative	bi**s**ogno	s
ʒ	Z	voiced postalveolar fricative	bi**j**ou	S
Vowels				
a	a	open front unrounded vowel	**a**rco	a
e	e	close-mid front unrounded vowel	tr**e**	e
ɛ	E	open-mid front unrounded vowel	**e**ttaro	E
i	i	close front unrounded vowel	**i**mpero	i
o	o	close-mid back rounded vowel	cent**o**	o
ɔ	O	open-mid back rounded vowel	**o**tto	O
u	u	close back rounded vowel	**u**no	u
Additional Symbols				

IPA	X-SAMPA	Description	Example	Viseme
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Japanese (ja-JP)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Japanese voice supported by Amazon Polly.

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
ɾ	4	alveolar flap	練習, **r**enshuu	t
ʔ	?	glottal stop	あつっ, atsu**'**	
b	b	voiced bilabial plosive	舞踊, **b**uyou	p
β	B	voiced bilabial fricative	ヴィンテージ, **v**inteeji	B
c	c	voiceless palatal plosive	ききょう, **kiky**ou	k
ç	C	voiceless palatal fricative	人, **h**ito	k
d	d	voiced alveolar plosive	濁点, **d**akuten	t
d͡ʑ	dz\	voiced alveolo-palatal affricate	純, **j**un	J
g	g	voiced velar plosive	ご飯, **g**ohan	k
h	h	voiceless glottal fricative	本, **h**on	k
j	j	palatal approximant	屋根, **y**ane	i
ɟ	J\	voiced palatal plosive	行儀, **gy**ou**g**i	J
k	k	voiceless velar plosive	漢字, **k**anji	k
ɺ	l\	alveolar lateral flap	釣り, tsu**r**i	r
ɺj	l\j	alveolar lateral flap, palatal approximant	流行, **ry**uukou	r
m	m	bilabial nasal	飯, **m**eshi	p

IPA	X-SAMPA	Description	Example	Viseme
n	n	alveolar nasal	猫, **n**eko	t
ɲ	J	palatal nasal	日本, **n**ippon	J
ɴ	N\	uvular nasal	缶, ka**n**	k
p	p	voiceless bilabial plosive	パン, **p**an	p
ɸ	p\	voiceless bilabial fricative	福, **h**uku	f
s	s	voiceless alveolar fricative	層, **s**ou	s
ɕ	s\	voiceless alveolo-palatal fricative	書簡, **sh**okan	J
t	t	voiceless alveolar plosive	手紙, **t**egami	t
t͡s	ts	voiceless alveolar affricate	釣り, **ts**uri	s
t͡ɕ	ts\	voiceless alveolo-palatal affricate	吉, ki**chi**	J
w	w	labial-velar approximant	電話, den**wa**	u
z	z	voiced alveolar fricative	座敷, **z**ashiki	s
Vowels				
äː	a:_"	long open central unrounded vowel	羽蟻, h**aa**ri	a
ä	a_"	open central unrounded vowel	仮名, k**ana**	a
eː	e:_o	long mid front unrounded vowel	学生, gakus**ei**	@
e	e_o	mid front unrounded vowel	歴, r**e**ki	@
i	i	close front unrounded vowel	気, k**i**	i
iː	i:	long close front unrounded vowel	詩歌, shi**i**ka	i
ɯ	M	close back unrounded vowel	運, **u**n	i
ɯː	M:	long close back unrounded vowel	宗教, sh**uu**kyou	i

IPA	X-SAMPA	Description	Example	Viseme
oː	o:_o	long mid back rounded vowel	購読, k**oo**doku	o
o	o_o	mid back rounded vowel	読者, d**o**kusha	o

Korean (ko-KR)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA)symbols, and the corresponding visemes for the Korean voice supported by Amazon Polly.

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
k	k	voiceless velar plosive	강, [g]ang	k
k#	k_t	strong voiceless velar plosive	깨, [kk]e	k
n	n	alveolar nasal	남, [n]am	t
t	t	voiceless alveolar plosive	도, [d]o	t
t#	t_t	strong voiceless alveolar plosive	때, [tt]e	t
ɾ	4	alveolar flap	사랑, sa[r]ang	t
l	l	alveolar lateral approximant	돌, do[l]	t
m	m	bilabial nasal	무, [m]u	p
p	p	voiceless bilabial plosive	봄, [b]om	p
p#	p_t	strong voiceless bilabial plosive	뻘, [pp]eol	p
s	s	voiceless alveolar fricative	새, [s]e	s
s#	s_t	strong voiceless alveolar fricative	씨, [ss]i	s
ŋ	N	velar nasal	방, ba[ng]	k
t͡ɕ	ts\	voiceless alveolo-palatal affricate	조, [j]o	J

IPA	X-SAMPA	Description	Example	Viseme
t͡#ɕ	ts_t	strong voiceless alveolo-palatal affricate	찌, [jj]i	J
t͡ɕʰ	ts_h	aspirated voiceless alveolo-palatal affricate	차, [ch]a	J
kʰ	k_h	aspirated voiceless velar plosive	코, [k]o	k
tʰ	t_h	aspirated voiceless alveolar plosive	통, [t]ong	t
pʰ	p_h	aspirated voiceless bilabial plosive	패, [p]e	p
h	h	voiceless glottal fricative	힘, [h]im	k
j	j	palatal approximant	양, [y]ang	i
w	w	labial-velar approximant	왕, [w]ang	u
ɰ	M\	velar approximant>	의, [wj]i	i
Vowels				
a	a	open front unrounded vowel	밥, b[a]b	a
ʌ	V	open-mid back unrounded vowel	정, j[eo]ng	E
ɛ	E	open-mid front unrounded vowel	배, b[e]	E
o	o	close-mid back rounded vowel	노, n[o]	o
u	u	close back rounded vowel	둘, d[u]l	u
ɯ	M	close back unrounded vowel	은, [eu]n	i
i	i	close front unrounded vowel	김, k[i]m	i

Norwegian (nb-NO)

The following chart lists the full set of International Phonetic Alphabet (IPA) phonemes and the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols as well as the corresponding visemes as supported by Amazon Polly for Norwegian language voices.

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
ɾ	4	alveolar flap	p**r**øv	t
b	b	voiced bilabial plosive	la**bb**	p
ç	C	voiceless palatal fricative	**k**ino	k
d	d	voiced alveolar plosive	la**dd**	t
ɖ	d`	voiced retroflex plosive	ve**rd**i	t
f	f	voiceless labiodental fricative	**f**ot	f
g	g	voiced velar plosive	ta**gg**	k
h	h	voiceless glottal fricative	**h**a	k
j	j	palatal approximant	**g**i	i
k	k	voiceless velar plosive	ta**kk**	k
l	l	alveolar lateral approximant	fa**ll**, ba**ll**	t
ɭ	l`	retroflex lateral approximant	æ**rl**ig	t
m	m	bilabial nasal	la**m**	p
n	n	alveolar nasal	va**nn**	t
ɳ	n`	retroflex nasal	ga**rn**	t
ŋ	N	velar nasal	sa**ng**	k
p	p	voiceless bilabial plosive	ho**pp**	p
s	s	voiceless alveolar fricative	la**ss**	s
ʂ	s`	voiceless retroflex fricative	å**rs**	S
ʃ	S	voiceless postalveolar fricative	**s**kyt	S
t	t	voiceless alveolar plosive	la**t**	t

IPA	X-SAMPA	Description	Example	Viseme
ʈ	t`	voiceless retroflex plosive	ha**rdt**	t
ʋ	v\	labiodental approximant	**v**in	f
w	w	labial-velar approximant	**w**ill	x
Vowels				
øː	2:	long close-mid front rounded vowel	s**ø**t	o
œ	9	open-mid front rounded vowel	s**ø**tt	O
ə	@	mid central vowel	ap**e**	@
æː	{:	long near-open front unrounded vowel	v**æ**r	a
ʉ	}	close central rounded vowel	l**u**nd	u
ʉː	}:	long close central rounded vowel	l**u**n	u
æ	{	near-open front unrounded vowel	v**æ**rt	a
ɑ	A	open back unrounded vowel	h**a**tt	a
ɑː	A:	long open back unrounded vowel	h**a**t	a
eː	e:	long close-mid front unrounded vowel	s**e**n	e
ɛ	E	open-mid front unrounded vowel	s**e**nd	E
iː	i:	long close front unrounded vowel	v**i**n	i
ɪ	I	near-close near-front unrounded vowel	v**i**nd	i
oː	oː	long close-mid back rounded vowel	v**å**t	o
ɔ	O	open-mid back rounded vowel	v**å**tt	O
uː	uː	long close back rounded vowel	b**o**k	u

IPA	X-SAMPA	Description	Example	Viseme
ʊ	U	near-close near-back rounded vowel	b**u**kk	u
yː	y:	long close front rounded vowel	l**y**n	u
ʏ	Y	near-close near-front rounded vowel	l**y**nne	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Polish (pl-PL)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Polish voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**obas, **b**elka	p
d	d	voiced alveolar plosive	**d**ar, **d**o	t
d͡z	dz	voiced alveolar affricate	**dz**won, wi**dz**owie	s
d͡ʑ	dz\	voiced alveolo-palatal affricate	**dź**więk	J
d͡ʐ	dz`	voiced retroflex affricate	**dż**em, **dż**ungla	S
f	f	voiceless labiodental fricative	**f**urtka, **f**ilm	f
g	g	voiced velar plosive	**g**azeta, wa**g**a	k
h	h	voiceless glottal fricative	**ch**leb, **h**andel	k
j	j	palatal approximant	**j**ak, ma**j**a	i
k	k	voiceless velar plosive	**k**ura, mare**k**	k

IPA	X-SAMPA	Description	Example	Viseme
l	l	alveolar lateral approximant	**l**ipa, a**l**icja	t
m	m	bilabial nasal	**m**atka, **m**olo	p
n	n	alveolar nasal	**n**orka	t
ɲ	J	palatal nasal	ko**ń**, toru**ń**	J
p	p	voiceless bilabial plosive	**p**ora, sto**p**	p
r	r	alveolar trill	**r**ok, pa**r**k	r
s	s	voiceless alveolar fricative	**s**um, pa**s**	s
ɕ	s\	voiceless alveolo-palatal fricative	**ś**ruba, **ś**nieg	J
ʂ	s`	voiceless retroflex fricative	**sz**um, ma**sz**	S
t	t	voiceless alveolar plosive	**t**ok, s**t**ół	t
t͡s	ts	voiceless alveolar affricate	**c**ar, **c**o	s
t͡ɕ	ts\	voiceless alveolo-palatal affricate	**ć**ma, mie**ć**	J
t͡ʂ	ts`	voiceless retroflex affricate	**cz**as, ra**cz**ej	S
v	v	voiced labiodental fricative	**w**orek, me**w**a	f
w	w	labial-velar approximant	**ł**aska, ma**ł**o	u
z	z	voiced alveolar fricative	**z**ero	s
ʑ	z\	voiced alveolo-palatal fricative	**ź**rebię, bieli**ź**nie	J
ʐ	z`	voiced retroflex fricative	**ż**ar, **ż**ona	S
Vowels				
a	a	open front unrounded vowel	**ja**	a
ɛ	E	open-mid front unrounded vowel	**e**cho	E

IPA	X-SAMPA	Description	Example	Viseme
ɛ̃	E~	nasal open-mid front unrounded vowel	w**ę**że	E
i	i	close front unrounded vowel	**i**le	i
ɔ	O	open-mid back rounded vowel	**o**czy	O
ɔ̃	O~	nasal open-mid back rounded vowel	w**ą**ż	O
u	u	close back rounded vowel	**u**czta	u
ɨ	1	close central unrounded vowel	b**y**k	i
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Portuguese (pt-PT)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Portuguese voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
ɾ	4	alveolar flap	pi**r**a	t
b	b	voiced bilabial plosive	**d**ato	p
d	d	voiced alveolar plosive	**d**ato	t
f	f	voiceless labiodental fricative	**f**acto	f
g	g	voiced velar plosive	**g**ato	k
j	j	palatal approximant	paragua**y**	i
k	k	voiceless velar plosive	**c**acto	k
l	l	alveolar lateral approximant	ga**l**o	t

IPA	X-SAMPA	Description	Example	Viseme
ʎ	L	palatal lateral approximant	ga**lh**o	J
m	m	bilabial nasal	**m**ato	p
n	n	alveolar nasal	**n**ato	t
ɲ	J	palatal nasal	pi**nh**a	J
p	p	voiceless bilabial plosive	**p**ato	p
ʀ	R\	uvular trill	ba**rr**oso	k
s	s	voiceless alveolar fricative	**s**aca	s
ʃ	S	voiceless postalveolar fricative	**ch**ato	S
t	t	voiceless alveolar plosive	**t**acto	t
v	v	voiced labiodental fricative	**v**aca	f
w	w	labial-velar approximant	ma**u**	u
z	z	voiced alveolar fricative	**z**aca	s
ʒ	Z	voiced postalveolar fricative	**j**acto	S
Vowels				
a	a	open front unrounded vowel	p**a**rto	a
a˜	a~	nasal open front unrounded vowel	p**e**ga	a
e	e	close-mid front unrounded vowel	p**e**ga	e
e˜	e~	nasal close-mid front unrounded vowel	mov**e**m	e
ɛ	E	open-mid front unrounded vowel	caf**é**	E
i	i	close front unrounded vowel	l**i**ngueta	i
ĩ	i~	nasal close front unrounded vowel	c**i**nto	i

IPA	X-SAMPA	Description	Example	Viseme
o	o	close-mid back rounded vowel	p**o**der	o
o˜	o~	nasal close-mid back rounded vowel	c**o**mpra	o
ɔ	O	open-mid back rounded vowel	cot**ó**	O
u	u	close back rounded vowel	f**u**i	u
u˜	u~	nasal close back rounded vowel	s**u**nto	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Portuguese, Brazilian (pt-BR)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Brazilian Portuguese voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
ɾ	4	alveolar flap	pi**r**a	t
b	b	voiced bilabial plosive	**b**ato	p
d	d	voiced alveolar plosive	**d**ato	t
d͡ʒ	dZ	voiced postalveolar affricate	ida**de**	S
f	f	voiceless labiodental fricative	**f**acto	f
g	g	voiced velar plosive	**g**ato	k
j	j	palatal approximant	paragua**y**	i
k	k	voiceless velar plosive	**c**acto	k
l	l	alveolar lateral approximant	ga**l**o	t

IPA	X-SAMPA	Description	Example	Viseme
ʎ	L	palatal lateral approximant	ga**lh**o	J
m	m	bilabial nasal	**m**ato	p
n	n	alveolar nasal	**n**ato	t
ɲ	J	palatal nasal	pi**nh**a	J
p	p	voiceless bilabial plosive	**p**ato	p
s	s	voiceless alveolar fricative	**s**aca	s
ʃ	S	voiceless postalveolar fricative	**ch**ato	S
t	t	voiceless alveolar plosive	**t**acto	t
t͡ʃ	tS	voiceless postalveolar affricate	noi**te**	S
v	v	voiced labiodental fricative	**v**aca	f
w	w	labial-velar approximant	ma**u**	u
χ	X	voiceless uvular fricative	ca**rr**o	k
z	z	voiced alveolar fricative	**z**aca	s
ʒ	Z	voiced postalveolar fricative	**j**acto	S
Vowels				
a	a	open front unrounded vowel	p**a**rto	a
a˜	a~	nasal open front unrounded vowel	pens**a**mos	a
e	e	close-mid front unrounded vowel	p**e**ga	e
e˜	e~	nasal close-mid front unrounded vowel	mov**e**m	e
ɛ	E	open-mid front unrounded vowel	caf**é**	E
i	i	close front unrounded vowel	l**i**ngueta	i

IPA	X-SAMPA	Description	Example	Viseme
ĩ	i~	nasal close front unrounded vowel	c**i**nto	i
o	o	close-mid back rounded vowel	p**o**der	o
õ	o~	nasal close-mid back rounded vowel	c**o**mpra	o
ɔ	O	open-mid back rounded vowel	cot**ó**	O
u	u	close back rounded vowel	f**u**i	u
ũ	u~	nasal close back rounded vowel	s**u**nto	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Romanian (ro-RO)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Romanian voice supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**ubă	p
d	d	voiced alveolar plosive	**d**upă	t
d͡ʒ	dZ	voiced postalveolar affricate	**ge**orge	S
f	f	voiceless labiodental fricative	a**f**acere	f
g	g	voiced velar plosive	a**g**riș	k
h	h	voiceless glottal fricative	**h**arpă	k
j	j	palatal approximant	ba**i**e	i

IPA	X-SAMPA	Description	Example	Viseme
k	k	voiceless velar plosive	**c**oș	k
l	l	alveolar lateral approximant	**l**ampa	t
m	m	bilabial nasal	**m**ama	p
n	n	alveolar nasal	**n**or	t
p	p	voiceless bilabial plosive	**p**ilă	p
r	r	alveolar trill	**r**ampă	r
s	s	voiceless alveolar fricative	**s**oare	s
ʃ	S	voiceless postalveolar fricative	ma**ș**ină	S
t	t	voiceless alveolar plosive	**t**ata	t
t͡s	ts	voiceless alveolar affricate	**ț**ară	s
t͡ʃ	tS	voiceless postalveolar affricate	**ce**ai	S
v	v	voiced labiodental fricative	**v**iață	f
w	w	labial-velar approximant	bea**u**	u
z	z	voiced alveolar fricative	mo**z**ol	s
ʒ	Z	voiced postalveolar fricative	**j**oacă	S
Vowels				
ə	@	mid central vowel	bab**ă**	@
a	a	open front unrounded vowel	c**a**sa	a
e	e	close-mid front unrounded vowel	**e**lan	e
e̯	e_^	non-syllabic close-mid front unrounded vowel	b**e**au	e
i	i	close front unrounded vowel	m**i**e	i

IPA	X-SAMPA	Description	Example	Viseme
o	o	close-mid back rounded vo	**o**ră	o
oa	o_^a	diphthong	**oa**re	o
u	u	close back rounded vowel	**u**nde	u
ɨ	1	close central unrounded vowel	Rom**â**nia	i
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Russian (ru-RU)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Russian voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**б**орт	p
b^j	b'	palatalized voiced bilabial plosive	**б**юро	p
d	d	voiced alveolar plosive	**д**ом	t
d^j	d'	palatalized voiced alveolar plosive	**д**я**д**я	t
f	f	voiceless labiodental fricative	**ф**лаг	f
f^j	f'	palatalized voiceless labiodental fricative	**ф**евраль	f
g	g	voiced velar plosive	но**г**а	k
g^j	g'	palatalized voiced velar plosive	**г**ерой	k
j	j	palatal approximant	диза**й**н, **я**щик	i
k	k	voiceless velar plosive	**к**от	k

IPA	X-SAMPA	Description	Example	Viseme
k^j	k'	palatalized voiceless velar plosive	**к**ино	k
l	l	alveolar lateral approximant	**л**ампа	t
l^j	l'	palatalized alveolar lateral approximant	**л**ес	t
m	m	bilabial nasal	**м**ама	p
m^j	m'	palatalized bilabial nasal	**м**яч	p
n	n	alveolar nasal	**н**ос	t
n^j	n'	palatalized alveolar nasal	**н**я**н**я	t
p	p	voiceless bilabial plosive	**п**апа	p
p^j	p'	palatalized voiceless bilabial plosive	**п**еро	p
r	r	alveolar trill	**р**оза	r
r^j	r'	palatalized alveolar trill	**р**юмка	r
s	s	voiceless alveolar fricative	**с**ыр	s
s^j	s'	palatalized voiceless alveolar fricative	**с**ердце, ру**сь**	s
ɕː	s\:	long voiceless alveolo-palatal fricative	**щ**ека	J
ʂ	s`	voiceless retroflex fricative	**ш**ум	S
t	t	voiceless alveolar plosive	**т**очка	t
t^j	t'	palatalized voiceless alveolar plosive	**т**ё**т**я	t
t͡s	ts	voiceless alveolar affricate	**ц**арь	s
t͡ɕ	ts\	voiceless alveolo-palatal affricate	**ч**ас	J
v	v	voiced labiodental fricative	**в**ор	f

IPA	X-SAMPA	Description	Example	Viseme
vʲ	v'	palatalized voiced labiodental fricative	**в**ерфь	f
x	x	voiceless velar fricative	**х**ор	k
xʲ	x'	palatalized voiceless velar fricative	**х**имия	k
z	z	voiced alveolar fricative	**з**уб	s
zʲ	z'	palatalized voiced alveolar fricative	**з**има	s
ʑː	z\:	long voiced alveolo-palatal fricative	уе**зж**ать	J
ʐ	z`	voiced retroflex fricative	**ж**ена	S
Vowels				
ə	@	mid central vowel	канарейк**а**	@
a	a	open front unrounded vowel	дв**а**, **я**блоко	a
e	e	close-mid front unrounded vowel	п**е**чь	e
ɛ	E	open-mid front unrounded vowel	**э**то	E
i	i	close front unrounded vowel	од**и**н, ч**е**тыре	i
o	o	close-mid back rounded vowel	к**о**т	o
u	u	close back rounded vowel	м**у**ж, вь**ю**га	u
ɨ	1	close central unrounded vowel	м**ы**шь	i

Spanish (es-ES)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Spanish voices that are supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				

IPA	X-SAMPA	Description	Example	Viseme
ɾ	4	alveolar flap	pe**r**o, b**r**avo, amo**r**, ete**r**no	t
b	b	voiced bilabial plosive	**b**estia	p
β	B	voiced bilabial fricative	be**b**é	B
d	d	voiced alveolar plosive	cuan**d**o	t
ð	D	voiced dental fricative	ar**d**er	T
f	f	voiceless labiodental fricative	**f**ase, ca**f**é	f
g	g	voiced velar plosive	**g**ato, len**g**ua, **g**uerra	k
ɣ	G	voiced velar fricative	tri**g**o, Ar**g**os	k
j	j	palatal approximant	hac**i**a, t**i**erra, rad**i**o, v**i**uda	i
ʝ	j\	voiced palatal fricative	enh**i**elar, sa**y**o, in**y**ectado, des**y**erba	J
k	k	voiceless velar plosive	**c**aña, la**c**a, **q**uisimos	k
l	l	alveolar lateral approximant	**l**ino, ca**l**or, principa**l**	t
ʎ	L	palatal lateral approximant	**ll**ave, po**ll**o	J
m	m	bilabial nasal	**m**adre, co**m**er, a**n**fibio	p
n	n	alveolar nasal	**n**ido, a**n**illo, si**n**	t
ɲ	J	palatal nasal	caba**ñ**a, **ñ**oquis	J
ŋ	N	velar nasal	ci**n**co, ve**n**ga	k
p	p	voiceless bilabial plosive	**p**ozo, to**p**o	p
r	r	alveolar trill	pe**rr**o, en**r**achado	r
s	s	voiceless alveolar fricative	**s**aco, ca**s**a, puerta**s**	s
t	t	voiceless alveolar plosive	**t**amiz, á**t**omo	t
t͡ʃ	tS	voiceless postalveolar affricate	**ch**ubasco	S

IPA	X-SAMPA	Description	Example	Viseme
θ	T	voiceless dental fricative	cere**z**a, **z**orro, la**c**ero, pa**z**	T
w	w	labial-velar approximant	f**u**ego, f**u**imos, c**u**ota, c**u**adro	u
x	x	voiceless velar fricative	**j**amón, **g**eneral, su**j**e, relo**j**	k
z	z	voiced alveolar fricative	ra**s**go, mi**s**mo	s
Vowels				
a	a	open front unrounded vowel	t**a**nque	a
e	e	close-mid front unrounded vowel	p**e**so	e
i	i	close front unrounded vowel	c**i**nco	i
o	o	close-mid back rounded vowel	b**o**sque	o
u	u	close-mid front unrounded vowel	p**u**blicar	u
e	e	close-mid front unrounded vowel	k**e**çi	e
ɛ	E	open-mid front unrounded vowel	ded**e**	e
i	i	close front unrounded vowel	b**i**r	i
i:	i:	long close front unrounded vowel	**i**zah	i
ɪ	I	near-close near-front unrounded vowel	keç**i**	i
ɯ	M	close back unrounded vowel	kıl	i
o	o	long close-mid back rounded vowel	k**o**l	o
o:	o:	long close-mid back rounded vowel	d**o**lar	o
u	u	close back rounded vowel	d**u**r**u**m	u
u:	u:	long close back rounded vowel	r**u**hum	u

IPA	X-SAMPA	Description	Example	Viseme
ʊ	U	near-close near-back rounded vowel	dol**u**	u
Y	y	close front rounded vowel	g**ü**venlik	u
ʏ	Y	near-close near-front rounded vowel	aş**ı**	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Spanish, Mexican (es-MX)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Mexican Spanish voice that is supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
ɾ	4	alveolar flap	pe**r**o, b**r**avo, amo**r**, ete**r**no	t
b	b	voiced bilabial plosive	**b**estia	p
β	B	voiced bilabial fricative	be**b**é	B
d	d	voiced alveolar plosive	cuan**d**o	t
ð	D	voiced dental fricative	ar**d**er	T
f	f	voiceless labiodental fricative	**f**ase, ca**f**é	f
g	g	voiced velar plosive	**g**ato, len**g**ua, **g**uerra	k
ɣ	G	voiced velar fricative	tri**g**o, Ar**g**os	k
j	j	palatal approximant	hac**i**a, t**i**erra, rad**i**o, v**i**uda	i
ʝ	j\	voiced palatal fricative	enh**i**elar, sa**y**o, in**y**ectado, des**y**erba	J

IPA	X-SAMPA	Description	Example	Viseme
k	k	voiceless velar plosive	**c**aña, la**c**a, **q**uisimos	k
l	l	lateral alveolar approximant	**l**ino, ca**l**or, principa**l**	t
m	m	bilabial nasal	**m**adre, co**m**er, a**n**fibio	p
n	n	alveolar nasal	**n**ido, a**n**illo, si**n**	t
ɲ	J	palatal nasal	caba**ñ**a, **ñ**oquis	J
ŋ	N	velar nasal	a**n**gosto, i**n**creíble	k
p	p	voiceless bilabial plosive	**p**ozo, to**p**o	p
r	r	alveolar trill	pe**rr**o, en**r**achado	r
s	s	voiceless alveolar fricative	**s**aco, ca**s**a, puerta**s**	s
ʃ	S	voiceless postalveolar fricative	**sh**ow, fla**sh**	S
t	t	voiceless alveolar plosive	**t**amiz, á**t**omo	t
t͡ʃ	tS	voiceless postalveolar affricate	**ch**ubasco	S
w	w	labial-velar approximant	f**u**ego, f**u**imos, c**u**ota, c**u**adro	u
x	x	voiceless velar fricative	**j**amón, **g**eneral, pea**j**e, relo**j**	k
z	z	voiced alveolar fricative	ra**s**go, mi**s**mo	s
h	h	voiceless glottal fricative	**H**arrison	k
ɹ	r\	postalveolar approximant	**Br**ian	r
v	v	voiced labiodental fricative	**V**ancouver	f
Vowels				
a	a	central open unrounded vowel	t**a**nque	a
e	e	close-mid front unrounded vowel	p**e**so	e

IPA	X-SAMPA	Description	Example	Viseme
i	i	close front unrounded vowel	c**i**nco	i
o	o	close-mid back rounded vowel	b**o**sque	o
u	u	close back rounded vowel	p**u**blicar	u
ɛ	E	open mid-front unrounded vowel	dr**e**ss	E
ɔ	O	open-mid back rounded vowel	F**o**rt	O
ə	@	mid central vowel	Laud**e**rdale	@
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Spanish, US (es-US)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the US Spanish voices that are supported by Amazon Polly.

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	Voiced bilabial plosive	**b**ed	p
d	d	Voiced alveolar plosive	**d**ig	t
d͡ʒ	dZ	Voiced postalveolar affricate	**j**ump	S
ð	D	Voiced dental fricative	**th**en	T
f	f	Voiceless labiodental fricative	**f**ive	f
g	g	Voiced velar plosive	**g**ame	k
h	h	Voiceless glottal fricative	**h**ouse	k
j	j	Palatal approximant	**y**es	i

IPA	X-SAMPA	Description	Example	Viseme
k	k	Voiceless velar plosive	**c**at	k
l	l	Alveolar lateral approximant	**l**ay	t
m	m	Bilabial nasal	**m**ouse	p
n	n	Alveolar nasal	**n**ap	t
ŋ	N	Velar nasal	thi**ng**	k
p	p	Voiceless bilabial plosive	s**p**eak	p
ɹ	r\	Alveolar approximant	**r**ed	r
s	s	Voiceless alveolar fricative	**s**eem	s
ʃ	S	Voiceless postalveolar fricative	**sh**ip	S
t	t	Voiceless alveolar plosive	**t**rap	t
t͡ʃ	tS	Voiceless postalveolar affricate	**ch**art	S
Θ	T	Voiceless dental fricative	**th**in	T
v	v	Voiced labiodental fricative	**v**est	f
w	w	Labial-velar approximant	**w**est	u
z	z	Voiced alveolar fricative	**z**ero	s
ʒ	Z	Voiced postalveolar fricative	vi**s**ion	S
Vowels				
ə	@	Mid central vowel	**a**ren**a**	@
ɚ	@`	Mid central r-colored vowel	read**er**	@
æ	{	Near open-front unrounded vowel	tr**a**p	a
aɪ	aI	Diphthong	pr**i**ce	a
aʊ	aU	Diphthong	m**ou**th	a

IPA	X-SAMPA	Description	Example	Viseme
ɑ	A	Long open-back unrounded vowel	f**a**ther	a
eɪ	eI	Diphthong	f**a**ce	e
ɝ	3`	Open mid-central unrounded r-colored vowel	n**ur**se	E
ɛ	E	Open mid-front unrounded vowel	dr**e**ss	E
i:	i	Long close front unrounded vowel	fl**ee**ce	i
ɪ	I	Near-close near-front unrounded vowel	k**i**t	i
oʊ	oU	Diphthong	g**oa**t	o
ɔ	O	Long open mid-back rounded vowel	th**ou**ght	O
ɔɪ	OI	Diphthong	ch**oi**ce	O
u	u	Long close-back rounded vowel	g**oo**se	u
ʊ	U	Near-close near-back rounded vowel	f**oo**t	u
ʌ	V	Open-mid-back unrounded vowel	str**u**t	E
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Swedish (sv-SE)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Swedish voice supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**il	p

IPA	X-SAMPA	Description	Example	Viseme
d	d	voiced alveolar plosive	**d**al	t
ɖ	d`	voiced retroflex plosive	bo**rd**	t
f	f	voiceless labiodental fricative	**f**il	f
g	g	voiced velar plosive	**g**ås	k
h	h	voiceless glottal fricative	**h**al	k
j	j	palatal approximant	**j**ag	i
k	k	voiceless velar plosive	**k**al	k
l	l	alveolar lateral approximant	**l**ös	t
ɭ	l`	retroflex lateral approximant	hä**rl**ig	t
m	m	bilabial nasal	**m**il	p
n	n	alveolar nasal	**n**ålar	t
ɳ	n`	retroflex nasal	ba**rn**	t
ŋ	N	velar nasal	ri**ng**	k
p	p	voiceless bilabial plosive	**p**il	p
r	r	alveolar trill	**r**is	r
s	s	voiceless alveolar fricative	**s**il	s
ɕ	s\	voiceless alveolo-palatal fricative	**tj**ock	J
ʂ	s`	voiceless retroflex fricative	fo**rs**, **sch**lager	S
t	t	voiceless alveolar plosive	**t**al	t
ʈ	t`	voiceless retroflex plosive	hjo**rt**	t
v	v	voiced labiodental fricative	**v**år	f
w	w	labial-velar approximant	a**u**la, air**w**ays	u

IPA	X-SAMPA	Description	Example	Viseme
ɧ	x\	voiceless palatal-velar fricative	**sj**uk	k
Vowels				
ø	2	close-mid front rounded vowel	f**ö**ll, f**ö**rr	o
ø	2:	long close-mid front rounded vowel	f**ö**l, n**ö**t, f**ö**r	o
ɵ	8	close-mid central rounded vowel	b**u**ss, f**u**ll	o
ə	@	mid central vowel	pojk**e**n	@
ʉː	}:	long close central rounded vowel	h**u**s, f**u**l	u
a	a	open front unrounded vowel	h**a**ll, m**a**tt	a
æ	{	near-open front unrounded vowel	h**e**rr	a
ɑː	A:	long open back unrounded vowel	h**a**l, m**a**t	a
eː	e:	long close-mid front unrounded vowel	v**e**t, h**e**l	e
ɛ	E	open-mid front unrounded vowel	v**e**tt, r**ä**tt, h**e**tta, h**ä**ll	E
ɛː	E:	long open-mid front unrounded vowel	s**ä**l, h**ä**l, h**ä**r	E:
iː	i:	long close front unrounded vowel	v**i**t, s**i**l	i:
ɪ	I	near-close near-front unrounded vowel	v**i**tt, s**i**ll	i
oː	o:	long close-mid back rounded vowel	h**å**l, m**å**l	o
ɔ	O	open-mid back rounded vowel	h**å**ll, m**o**ll	O
uː	u:	long close back rounded vowel	s**o**l, b**o**t	u
ʊ	U	near-close near-back rounded vowel	b**o**tt	u
y	y	close front rounded vowel	b**y**tt	u

IPA	X-SAMPA	Description	Example	Viseme
yː	y:	long close front rounded vowel	s**y**l, s**y**l	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Turkish (tr-TR)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Turkish voice supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
ɾ	4	alveolar flap	du**r**um	t
ɾ̝̊	4_0_r	voiceless fricated alveolar flap	bi**r**	t
ɾ̝	4_r	fricated alveolar flap	**r**af	t
b	b	voiced bilabial plosive	**r**af	p
c	c	voiceless palatal plosive	**k**edi	k
d	d	voiced alveolar plosive	**d**e**d**e	t
d͡ʒ	dZ	voiced postalveolar affricate	**c**am	S
f	f	voiceless labiodental fricative	**f**are	f
g	g	voiced velar plosiv	**g**alibi	k
h	h	voiceless glottal fricative	**h**asta	k
j	j	palatal approximant	**y**at	i
ɟ	J\	voiced palatal plosive	**g**enç	J
k	k	voiceless velar plosive	a**k**ıl	k

IPA	X-SAMPA	Description	Example	Viseme
l	l	alveolar lateral approximant	**l**a**l**e	t
ɫ	5	velarized alveolar lateral approximant	**l**abirent	t
m	m	bilabial nasal	**m**aaş	p
n	n	alveolar nasal	a**n**ı	t
p	p	voiceless bilabial plosive	i**p**	p
s	s	voiceless alveolar fricative	**s**es	s
ʃ	S	voiceless postalveolar fricative	a**ş**ı	S
t	t	voiceless alveolar plosive	ü**t**ü	t
t͡ʃ	tS	voiceless postalveolar affricate	**ç**aba	S
v	v	voiced labiodental fricative	ek**v**ator, kah**v**eci, ak**v**aryum, is**v**eçli, teş**v**iki, cet**v**el	f
z	z	voiced alveolar fricative	**v**er	s
ʒ	Z	voiced postalveolar fricative	a**z**ık	S
Vowels				
ø	2	close-mid front rounded vowel	g**ö**l	0
œ	9	open-mid front rounded vowel	banliy**ö**	O
a	a	open front unrounded vowel	k**a**l	a
a:	a:	long open front unrounded vowel	d**a**v**a**cı	a
æ	{	near-open front unrounded vowel	özl**e**m, güv**e**nlik, gür**e**l, som**e**rsault	a
e	e	close-mid front unrounded vowel	k**e**çi	e
ɛ	E	open-mid front unrounded vowel	ded**e**	E

IPA	X-SAMPA	Description	Example	Viseme
i	i	close front unrounded vowel	b**i**r	i
iː	i:	long close front unrounded vowel	**i**zah	i
ɪ	I	near-close near-front unrounded vowel	keç**i**	i
ɯ	M	close back unrounded vowel	k**ı**l	i
o	o	close-mid back rounded vowel	k**o**l	o
oː	o:	long close-mid back rounded vowel	d**o**lar	o
u	u	close back rounded vowel	d**u**r**u**m	u
uː	u:	long close back rounded vowel	r**u**hum	u
ʊ	U	near-close near-back rounded vowel	dol**u**	u
y	y	close front rounded vowel	g**ü**venlik	u
ʏ	Y	near-close near-front rounded vowel	aş**ı**	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Welsh (cy-GB)

The following table lists the International Phonetic Alphabet (IPA) phonemes, the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols, and the corresponding visemes for the Welsh voice supported by Amazon Polly.

Phoneme/Viseme Table

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	voiced bilabial plosive	**b**aban	p

IPA	X-SAMPA	Description	Example	Viseme
d	d	voiced alveolar plosive	**d**eg	t
d͡ʒ	dZ	voiced postalveolar affricate	gare**j**	S
ð	D	voiced dental fricative	deu**dd**eg	T
f	f	voiceless labiodental fricative	**ff**acs	f
g	g	voiced velar plosive	**g**adael	k
h	h	voiceless glottal fricative	**h**aearn	k
j	j	palatal approximant	astud**i**o	i
k	k	voiceless velar plosive	**c**ant	k
l	l	alveolar lateral approximant	**l**an	t
ɬ	K	voiceless alveolar lateral fricative	**ll**an	t
m	m	bilabial nasal	**m**ae	p
m̥	m_0	voiceless bilabial nasal	y**mh**en	p
n	n	alveolar nasal	**n**aw	t
n̥	n_0	voiceless alveolar nasal	a**nh**awster	t
ŋ	N	velar nasal	argyfw**ng**	k
ŋ̊	N_0	voiceless velar nasal	a**ngh**enion	k
p	p	voiceless bilabial plosive	**p**ump	p
r	r	alveolar trill	**rh**oi	r
r̥	r_0	voiceless alveolar trill	ga**rw**	r
s	s	voiceless alveolar fricative	**s**aith	s
ʃ	S	voiceless postalveolar fricative	**si**awns	S
t	t	voiceless alveolar plosive	**t**egan	t

IPA	X-SAMPA	Description	Example	Viseme
t͡ʃ	tS	voiceless postalveolar affricate	cy**ts**ain	S
θ	T	voiceless dental fricative	aber**th**	T
v	v	voiced labiodental fricative	praw**f**	f
w	w	labial-velar approximant	rhag**w**eld	u
χ	X	voiceless uvular fricative	**ch**we**ch**	k
z	z	voiced alveolar fricative	aid**s**	s
ʒ	Z	voiced postalveolar fricative	rou**ge**	S
Vowels				
ə	@	mid central vowel	**y**chwaneg**a**	@
a	a	open front unrounded vowel	**a**cen	a
ai	ai	diphthong	d**au**	a
au	au	diphthong	**aw**dur	a
ɑː	A:	long open back unrounded vowel	m**a**b	a
ɑːɨ	A:1	diphthong	**ae**lod	a
eː	e:	long close-mid front unrounded vowel	p**e**th	e
ɛ	E	open-mid front unrounded vowel	p**e**dwar	E
ɛi	Ei	diphthong	b**ei**c	E
iː	i:	long close front unrounded vowel	tr**i**	i
ɪ	I	near-close near-front unrounded vowel	m**i**liwn	i
ɨu	1u	diphthong	unigr**yw**	i
oː	o:	long close-mid back rounded vowel	**o**ddi	o
ɔ	O	open-mid back rounded vowel	**o**ddieithr	O

IPA	X-SAMPA	Description	Example	Viseme
ɔi	Oi	diphthong	tr**oi**	O
ɔu	Ou	diphthong	r**ow**nd	O
u:	u:	long close back rounded vowel	c**w**ch	u
ʊ	U	near-close near-back rounded vowel	ac**w**stig	u
ʊi	Ui	diphthong	**wy**th	u
Additional Symbols				
ˈ	"	primary stress	Ala**ba**ma	
ˌ	%	secondary stress	**A**labama	
.	.	syllable boundary	A.la.ba.ma	

Neural TTS

Amazon Polly has a *Neural TTS (NTTS)* system that can produce even higher quality voices than its standard voices. The NTTS system produces the most natural and human-like text-to-speech voices possible.

Standard TTS voices use concatenative synthesis. This method strings together (concatenates) the phonemes of recorded speech, producing very natural-sounding synthesized speech. However, the inevitable variations in speech and the techniques used to segment the waveforms limits the quality of speech.

The Amazon Polly Neural TTS system doesn't use standard concatenative synthesis to produce speech. It has two parts:

- A neural network that converts a sequence of phonemes—the most basic units of language—into a sequence of *spectrograms*, which are snapshots of the energy levels in different frequency bands
- A vocoder, which converts the spectrograms into a continuous audio signal.

The first component of the neural TTS system is a sequence-to-sequence model. This model doesn’t create its results solely from the corresponding input but also considers how the sequence of the elements of the input work together. The model chooses the spectrograms that it outputs so that their frequency bands emphasize acoustic features that the human brain uses when processing speech.

The output of this model then passes to a neural vocoder. This converts the spectrograms into speech waveforms. When trained on the large data sets used to build general-purpose concatenative-synthesis systems, this sequence-to-sequence approach will yield higher-quality, more natural-sounding voices.

Topics

Feature and Region Compatibility

Neural voices aren't available in all AWS Regions, nor do they support all Amazon Polly features.

Neural voices are supported in the following Regions:

- US East (N. Virginia): us-east-1
- US West (Oregon): us-west-2
- EU (Ireland): eu-west-1
- Asia Pacific (Sydney): ap-southeast-2

Endpoints and protocols for these Regions are identical to those used for standard voices. For more information, see Regions and Endpoints.

The following features are supported for neural voices:

- Real-time and asynchronous speech synthesis operations.
- Newscaster and Conversational speaking styles. (For more information about the apeaking styles, see NTTS Speaking Styles (p. 91).)
- All speechmarks.
- Many (but not all) of the SSML tags that are supported by Amazon Polly. (For more information about NTTS-supported SSML tags, see Supported SSML Tags (p. 107).)

As with standard voices, you can choose from various sampling rates to optimize the bandwidth and audio quality for your application. Valid sampling rates for standard and neural voices are 8 kHz, 16 kHz, 22 kHz, or 24 kHz. The default for standard voices is 22 kHz. The default for neural voices is 24 kHz. Amazon Polly supports MP3, OGG (Vorbis), and raw PCM audio stream formats.

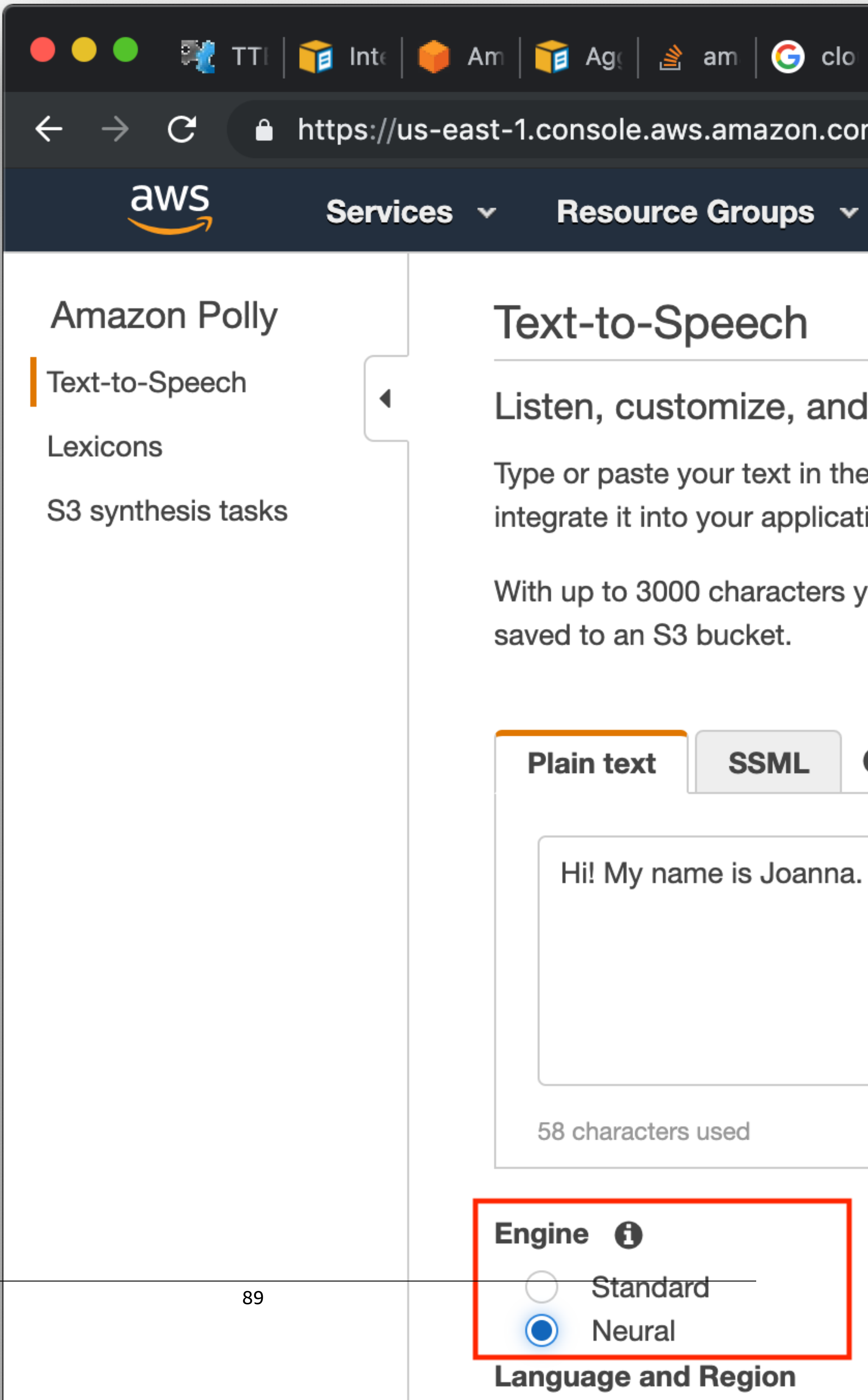
The Voice Engine

Amazon Polly enables you to use either neural or standard voice with the `engine` property. It has two possible values: **Standard** or **Neural**, indicating whether to use a standard or neural voice. **Standard** is the default value.

Choosing the Voice Engine (Console)

To choose a voice engine (console)

1. Open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. On the Text-to-Speech page, for **Engine**, choose **Standard** or **Neural**.



Text-to-Speech

Listen, customize, and

Type or paste your text in the
integrate it into your applicati

With up to 3000 characters y
saved to an S3 bucket.

Hi! My name is Joanna.

58 characters used

Engine

Standard [✓ Neural]

Language and Region

If you choose **Neural**, only neural voices are available and standard-only voices are disabled.

Choosing the Voice Engine (CLI)

To choose a voice engine (CLI)

The `engine` parameter is optional, with two possible values: `standard` or `Neural`. Use this property when creating a `SynthesisSynthesisTask` operation.

For example, you can use the following code to run the `start-speech-synthesis-task` AWS CLI command in the US West-2 (Oregon) region

The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

```
aws polly start-speech-synthesis-task \
  --engine neural
  --region us-west-2 \
  --endpoint-url "https://polly.us-west-1.amazonaws.com/" \
  --output-format mp3 \
  --output-s3-bucket-name your-bucket-name \
  --output-s3-key-prefix optional/prefix/path/file \
  --voice-id Joanna \
  --text file://text_file.txt
```

This will result in a response that looks similar to this:

```
"SynthesisTask":
{
     "CreationTime": [..],
     "Engine": "neural",
     "OutputFormat": "mp3",
     "OutputUri": "https://s3.us-west-1.amazonaws.com/your-bucket-name/optional/prefix/
path/file.<task_id>.mp3",
     "TextType": "text",
     "RequestCharacters": [..],
     "TaskStatus": "scheduled",
     "TaskId": [task_id],
     "VoiceId": "Joanna"
 }
```

Neural Voices

Neural voices are available in US English (es-US), and British English (es-GB), Brazilian Portuguese (pt-BR), and US Spanish (es-US). The following table lists the voices.

Language	Name/ID	Gender
English (British) (en-GB)	Amy	Female
	Emma	Female

Language	Name/ID	Gender
	Brian	Male
English (US) (en-US)	Ivy	Female (child)
	Joanna	Female
	Kendra	Female
	Kimberly	Female
	Salli	Female
	Joey	Male
	Justin	Male (child)
	Matthew	Male
Portuguese (Brazil) (pt-BR)	Camila	Female
Spanish (US) (es-US)	Lupe	Female

The **bold** voices can be used with the Newscaster and Conversational speaking styles. For more information, see NTTS Speaking Styles (p. 91).

NTTS Speaking Styles

People use different speaking styles, depending on context. Casual conversation, for example, sounds very different from a TV or radio newscast. When Amazon Polly synthesizes speech using standard voices, it uses the concatenative method. The concatenative method strings together short speech snippets stored in an audio database to produce the optimal, most natural sounding speech possible. However, because of the way these voices are made, they can't produce different speaking styles.

In addition to the standard concatenative synthesis, Amazon Polly can use neural technology to produce speech. Amazon Polly generates neural voices using a sequence-to-sequence model. This model produces results that uses the audio data input to form the voice, and also considers its position in the sequence of outputs. It can then be used as a very natural voice as it is, or it can be trained for a specific speaking style, with the variations and emphasis on certain parts of speech inherent in that style.

Amazon Polly provides two speaking styles that you can use: Newscaster and Conversational.

The Newscaster style uses the neural system to generate speech in the style of a TV or radio newscaster. The Newscaster style is available only for the Matthew and Joanna voices, which are available only in American English (en-US).

The Conversational style uses the neural system to generate speech in a more friendly and expressive conversational style that can be used in many use cases. The Conversational style is available only for the Matthew and Joanna voices, which are available only in American English (en-US).

Topics

Using the Newscaster Style

The Newscaster style is available only for the Matthew and Joanna voices, using the neural engine. This style is available only in American English (en-US). To use the Newscaster style, first choose the neural engine and then use the syntax described in the following steps in your input text.

> **Note**
> To use any neural speaking style, you must use one of the AWS Regions that support neural voices. This option is not available in all Regions. For more information, see Feature and Region Compatibility (p. 87).

To apply the Newscaster style (console)

1. Open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Ensure that you are using an AWS Region where neural voices are supported.
3. On the Text-to-Speech page, for **Engine**, choose **Neural**.
4. Choose the **SSML** tab.
5. Add input text to your text-to-speech request using the Newscaster style SSML syntax.

To apply the Newscaster style (CLI)

1. In your API request, include the engine parameter with the `neural` value:

   ```
   --engine neural
   ```
2. Add input text to your API request using the Newscaster style SSML syntax.

Use the Newscaster style SSML syntax in your input file.

```
<amazon:domain name="news">text</amazon:domain>
```

For example, you might use the newscaster tag with the Matthew or Joanna voice as follows:

```
<speak>
<amazon:domain name="news">
From the Tuesday, April 16th, 1912 edition of The Guardian newspaper:

The maiden voyage of the White Star liner Titanic, the largest ship ever launched
ended in disaster.

The Titanic started her trip from Southampton for New York on Wednesday. Late on
Sunday night she struck an iceberg off the Grand Banks of Newfoundland. By
wireless telegraphy she sent out signals of distress, and several liners were
near enough to catch and respond to the call.
</amazon:domain>
</speak>
```

For more information about SSML, see Supported SSML Tags (p. 107).

Using the Conversational Style

The Conversational style is available only for the Matthew and Joanna voices, using the neural engine. This style is available only in American English (en-US). To use the Conversational style, first choose the neural engine and then use the syntax described in the following steps in your input text.

Note
To use any neural speaking style, you must use one of the AWS Regions that support neural voices. This option is not available in all Regions. For more information, see Feature and Region Compatibility (p. 87).

To apply the Conversational style (console)

1. Open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Ensure that you are using an AWS Region where neural voices are supported.
3. On the Text-to-Speech page, for **Engine**, choose **Neural**.
4. Choose the **SSML** tab.
5. Add input text to your text-to-speech request using the Conversational style SSML syntax.

To apply the Conversational style (CLI)

1. In your API request, include the engine parameter with the `neural` value:

   ```
   --engine neural
   ```
2. Add input text to your API request using the Conversational style SSML syntax.

Use the following Conversational style SSML in your input file.

```
<amazon:domain name="conversational">text</amazon:domain>
```

For example, you might use the conversational tag with the Matthew or Joanna voice in the following quotation from H.G. Wells's novel *War of the Worlds*:

```
<speak>
<amazon:domain name="conversational">
No one would have believed in the last years of the nineteenth century that this world was
 being watched
keenly and closely by intelligences greater than man's and yet as mortal as his own; that
 as men busied
themselves about their various concerns they were scrutinised and studied perhaps almost as
 narrowly as a
man with a microscope might scritinise the transient creatures that swarm and multiply in a
 drop of water.
</amazon:domain>
</speak>
```

For more information about SSML, see Supported SSML Tags (p. 107).

Speech Marks

Speech marks are metadata that describe the speech that you synthesize, such as where a sentence or word starts and ends in the audio stream. When you request speech marks for your text, Amazon Polly returns this metadata instead of synthesized speech. By using speech marks in conjunction with the synthesized speech audio stream, you can provide your applications with an enhanced visual experience.

For example, combining the metadata with the audio stream from your text can enable to you synchronize speech with facial animation (lip-syncing) or to highlight written words as they're spoken.

Speechmarks are available when using either neural or standard text-to-speech formats.

Topics

Speech Mark Types

You request speech marks using the SpeechMarkTypes option for either the SynthesizeSpeech or StartSpeechSynthesisTask commands. You specify the metadata elements that you want to return from your input text. You can request as many as four types of metadata but you must specify at least one per request. No audio output is generated with the request.

In the AWS CLI, for example:

```
--speech-mark-types='["sentence", "word", "viseme", "ssml"]'
```

Amazon Polly generates speech marks using the following elements:

- **sentence** – Indicates a sentence element in the input text.
- **word** – Indicates a word element in the text.
- **viseme** – Describes the face and mouth movements corresponding to each phoneme being spoken. For more information, see Visemes and Amazon Polly (p. 94).
- **ssml** – Describes a <mark> element from the SSML input text. For more information, see Generating Speech from SSML Documents (p. 100).

Visemes and Amazon Polly

A *viseme* represents the position of the face and mouth when saying a word. It is the visual equivalent of a phoneme, which is the basic acoustic unit from which a word is formed. Visemes are the basic visual building blocks of speech.

Each language has a set of viseme that correspond to their specific phonemes.. In a language, each phoneme has a corresponding viseme that represents the shape that the mouth makes when forming the sound. However, not all visemes can be mapped to a particular phoneme because numerous phonemes appear the same when spoken, even though they sound different. For example, in English, the words "pet" and "bet" are acoustically different. However, when observed visually (without sound), they look exactly the same.

The following chart shows a partial list of International Phonetic Alphabet (IPA) phonemes and Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols as well as their corresponding visemes for US English voices.

For the complete table and tables for all available languages, see Phoneme and Viseme Tables for Supported Languages (p. 18).

IPA	X-SAMPA	Description	Example	Viseme
Consonants				
b	b	Voiced bilabial plosive	**b**ed	p
d	d	Voiced alveolar plosive	**d**ig	t
d͡ʒ	dZ	Voiced postalveolar affricate	**j**ump	S
ð	D	Voiced dental fricative	**th**en	T
f	f	Voiceless labiodental fricative	**f**ive	f
g	g	Voiced velar plosive	**g**ame	k
h	h	Voiceless glottal fricative	**h**ouse	k
...	...	...	...	...

Using Speech Marks

Requesting Speech Marks

To request speech marks for input text, use the `synthesize-speech` command. Besides the input text, the following elements are required to return this metadata:

- `output-format`

  Amazon Polly supports only the JSON format when returning speech marks.

```
--output-format json
```

  If you use an unsupported output format, Amazon Polly throws an exception.

- `voice-id`

  To ensure that the metadata matches the associated audio stream, specify the same voice that is used to generate the synthesized speech audio stream. The available voices don't have identical speech rates. If you use a voice other than the one used to generate the speech, the metadata will not match the audio stream.

```
--voice-id Joanna
```

- `speech-mark-types`

  Specify the type or types of speech marks you want. You can request any or all of the speech mark types, but must specify at least one type.

```
--speech-mark-types='["sentence", "word", "viseme", "ssml"]'
```

- `text-type`

  Plain text is the default input text for Amazon Polly, so you must use `text-type ssml` if you want to return SSML speech marks.
- `outfile`

  Specify the output file to which the metadata is written.

```
MaryLamb.txt
```

The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

```
aws polly synthesize-speech \
  --output-format json \
  --voice-id Voice ID \
  --text 'Input text' \
  --speech-mark-types='["sentence", "word", "viseme"]' \
  outfile
```

Speech Mark Output

Amazon Polly returns speech mark objects in a line-delimited JSON stream. A speech mark object contains the following fields:

- **time** – the timestamp in milliseconds from the beginning of the corresponding audio stream
- **type** – the type of speech mark (sentence, word, viseme, or ssml)
- **start** – the offset in bytes (not characters) of the start of the object in the input text (not including viseme marks)
- **end** – the offset in bytes (not characters) of the object's end in the input text (not including viseme marks)
- **value** – this varies depending on the type of speech mark
  - **SSML**: <mark> SSML tag
  - **viseme**: the viseme name
  - **word** or **sentence**: a substring of the input text, as delimited by the start and end fields

For example, Amazon Polly generates the following `word` speech mark object from the text "Mary had a little lamb":

```
{"time":373,"type":"word","start":5,"end":8,"value":"had"}
```

The described word ("had") begins 373 milliseconds after the audio stream begins, and starts at byte 5 and ends at byte 8 of the input text.

> **Note**
> This metadata is for the `Joanna` voice-id. If you use another voice with the same input text, the metadata might differ.

Speech Mark Examples

The following examples of speech mark requests show how to make common requests and the output that they generate.

Example 1: Speech Marks Without SSML

The following example shows you what requested metadata looks like on your screen for the simple sentence: "Mary had a little lamb." For simplicity, we don't include SSML speech marks in this example.

The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

```
aws polly synthesize-speech \
  --output-format json \
  --voice-id Joanna \
  --text 'Mary had a little lamb.' \
  --speech-mark-types='["viseme", "word", "sentence"]' \
  MaryLamb.txt
```

When you make this request, Amazon Polly returns the following in the .txt file:

```
{"time":0,"type":"sentence","start":0,"end":23,"value":"Mary had a little lamb."}
{"time":6,"type":"word","start":0,"end":4,"value":"Mary"}
{"time":6,"type":"viseme","value":"p"}
{"time":73,"type":"viseme","value":"E"}
{"time":180,"type":"viseme","value":"r"}
{"time":292,"type":"viseme","value":"i"}
{"time":373,"type":"word","start":5,"end":8,"value":"had"}
{"time":373,"type":"viseme","value":"k"}
{"time":460,"type":"viseme","value":"a"}
{"time":521,"type":"viseme","value":"t"}
{"time":604,"type":"word","start":9,"end":10,"value":"a"}
{"time":604,"type":"viseme","value":"@"}
{"time":643,"type":"word","start":11,"end":17,"value":"little"}
{"time":643,"type":"viseme","value":"t"}
{"time":739,"type":"viseme","value":"i"}
{"time":769,"type":"viseme","value":"t"}
{"time":799,"type":"viseme","value":"t"}
{"time":882,"type":"word","start":18,"end":22,"value":"lamb"}
{"time":882,"type":"viseme","value":"t"}
{"time":964,"type":"viseme","value":"a"}
{"time":1082,"type":"viseme","value":"p"}
```

In this output, each part of the text is broken out in terms of speech marks:

- The sentence "Mary had a little lamb."

- Each word in the text: "Mary", "had", "a", "little", and "lamb."
- The viseme for each sound in the corresponding audio stream: "p", "E", "r", "i", and so on. For more information on visemes see Visemes and Amazon Polly (p. 94).

Example 2: Speech Marks with SSML

The process of generating speech marks from SSML-enhanced text is similar to the process when SSML is not present. Use the `synthesize-speech` command, and specify the SSML-enhanced text and the type of speech marks that you want, as shown in the following example. To make the example easier to read, we do not include viseme speech marks, but these could be included as well.

The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

```
aws polly synthesize-speech \
  --output-format json \
  --voice-id Joanna \
  --text-type ssml \
  --text '<speak><prosody volume="+20dB">Mary had <break time="300ms"/>a little <mark
 name="animal"/>lamb</prosody></speak>' \
  --speech-mark-types='["sentence", "word", "ssml"]' \
  output.txt
```

When you make this request, Amazon Polly returns the following in the .txt file:

```
{"time":0,"type":"sentence","start":31,"end":95,"value":"Mary had <break time=\"300ms\"\/>a
 little <mark name=\"animal\"\/>lamb"}
{"time":6,"type":"word","start":31,"end":35,"value":"Mary"}
{"time":325,"type":"word","start":36,"end":39,"value":"had"}
{"time":897,"type":"word","start":40,"end":61,"value":"<break time=\"300ms\"\/>"}
{"time":1291,"type":"word","start":61,"end":62,"value":"a"}
{"time":1373,"type":"word","start":63,"end":69,"value":"little"}
{"time":1635,"type":"ssml","start":70,"end":91,"value":"animal"}
{"time":1635,"type":"word","start":91,"end":95,"value":"lamb"}
```

Requesting Speech Marks (Console)

You can use the console to request speech marks from Amazon Polly. You can then view the metadata or save it to a file.

To generate speech marks (console)

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Choose the **Text-to-Speech** tab.
3. Continue using the **Plain Text** tab or choose the **SSML** tab.
4. Type or paste your text into the input box.
5. For **Language and region**, choose the language for your text.
6. For **Voice**, choose the voice you want to use for the text.
7. To change text pronunciation, choose **Customize Pronunciation**, and for **Apply Lexicon** choose the desired lexicon.
8. To verify that the speech is in its final form, choose **Listen to speech**.

9. Choose **Change File Format**.

    Note
    Downloading MP3, OGG, or PCM formats will not generate speech marks.
10. For **File Format**, choose **Speech Marks**.
11. For **Speech Mark Types**, choose the types of speech marks to generate. The option to choose **SSML** metadata is only available on the **SSML** tab. For more information on using SSML with Amazon Polly see Generating Speech from SSML Documents (p. 100).
12. Choose **Change**.
13. Choose **Download Speech Marks**.

Generating Speech from SSML Documents

You can use Amazon Polly to generate speech from either plain text or from documents marked up with Speech Synthesis Markup Language (SSML). Using SSML-enhanced text gives you additional control over how Amazon Polly generates speech from the text you provide.

For example, you can include a long pause within your text, or change the speech rate or pitch. Other options include:

- emphasizing specific words or phrases
- using phonetic pronunciation
- including breathing sounds
- whispering
- using the Newscaster speaking style.

For complete details on the SSML tags supported by Amazon Polly and how to use them, see Supported SSML Tags (p. 107)

When using SSML, there are several reserved characters that require special treatment. This is because SSML uses these characters as part of its code. In order to use them, you use a specific entity to *escape* them. For more information, see Reserved Characters in SSML (p. 100)

Amazon Polly provides these types of control with a subset of the SSML markup tags that are defined by Speech Synthesis Markup Language (SSML) Version 1.1, W3C Recommendation.

You can use SSML within the Amazon Polly console or by using the AWS CLI. The following topics show you how you can use SSML to generate speech and control the output so that it precisely fits your needs.

Topics

Reserved Characters in SSML

There are five predefined characters that can't normally be used within an SSML statement. These entities are reserved by the language specification. These characters are

Name	Character	Escape code
quotation mark (double quotation mark)	"	&quot;
ampersand	&	&amp;

Name	Character	Escape code
apostrophe or single quotation mark	'	&apos;
less than sign	<	&lt;
greater than sign	>	&gt;

Because SSML uses these characters as part of its code, to use these symbols in SSML, you must *escape* the character when you use it. You use the escape code instead of the actual character so it displays properly while still creating a valid SSML document. For example, the following sentence

```
We're using the lawyer at Peabody & Chambers, attorneys-at-law.
```

would be rendered in SSML as

```
<speak>
We&apos;re using the lawyer at Peabody &amp; Chambers, attorneys-at-law.
</speak>
```

In this case, the special characters for the apostrophe and ampersand are escaped so the SSML document remains valid.

For the **&**, **<**, and **>** symbols, escape codes are always necessary when you use SSML. Additionallty, when you use the apostrophe/single quotation mark (**'**) as an apostrophe, you must also use the escape code.

However, when you use the double quotation mark (**"**), or the apostrophe/single quotation mark (**'**) as a quotation mark, then whether or not you use the escape code is dependent on context.

Double quotation marks

- Must be escaped when in a attribute value delimited by double quotes. For example, in the following AWS CLI code

  ```
  --text "Pete &quot;Maverick&quot; Mitchell"
  ```

- Do not need to be escaped when in textual context. For example, in the following

  ```
  He said, "Turn right at the corner."
  ```

- Do not need to be escaped when in a attribute value delimited by single quotes. For example, in the following AWS CLI code

  ```
  --text 'Pete "Maverick" Mitchell'
  ```

Single quotation marks

- Must be escaped when used as an apostrophe. For example, in the following

```
We&apos;ve got to leave quickly.
```

- Do not need to be escaped when in textual context. For example, in the following

```
"And then I said, 'Don't quote me.'"
```

- Do not need to be escaped when in a code attribute delimited by double quotes. For example, in the following AWS CLI code

```
--text "Pete 'Maverick' Mitchell"
```

Using SSML (Console)

With SSML tags, you can customize and control aspects of speech such as pronunciation, volume, and speech rate. In the AWS Console, the SSML-enhanced text that you want to convert to audio is entered on the SSML tab of the Text-to-Speech page. Although text entered in plain text relies on default settings for the language and voice you've chosen, text enhanced with SSML tells Amazon Polly not only what you want to say, but how you want to say it. Except for the added SSML tags, Amazon Polly synthesizes SSML-enhanced text in the same way as it synthesizes plain text. See Exercise 2: Synthesizing Speech with Plain Text Input (Console) (p. 6) for more information.

When using SSML, you enclose the entire text in a `<speak>` tag to let Amazon Polly know that you're using SSML. For example:

```
<speak>Hi! My name is Joanna. I will read any text you type here.</speak>
```

You then use specific SSML tags on the text inside the `<speak>` tags to customize the way you want the text to sound. You can add a pause, change the pace of the speech, lower or raise the volume of the voice, or add many other customizations so that the text sounds right for you. For a full list of the SSML tags that you can use, see Supported SSML Tags (p. 107).

In the following example, you use an SSML tag to tell Amazon Polly to substitute "World Wide Web Consortium" for "W3C" when it speaks a short paragraph. You also use tags to introduce a pause and whisper a word. Compare the results of this exercise with that of Applying Lexicons Using the Console (Synthesize Speech) (p. 130) .

For more information on SSML, with examples, see Supported SSML Tags (p. 107).

To synthesize speech from SSML-enhanced text (console)

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. If it isn't already displayed, choose the **Text-to-Speech** tab.
3. Choose the **SSML** tab.
4. Type or paste the following text in the text box:

```
<speak>
     He was caught up in the game.<break time="1s"/> In the middle of the
     10/3/2014 <sub alias="World Wide Web Consortium">W3C</sub> meeting,
     he shouted, "Nice job!" quite loudly. When his boss stared at him, he repeated
     <amazon:effect name="whispered">"Nice job,"</amazon:effect> in a
     whisper.
</speak>
```

The SSML tags tell Amazon Polly how to render the text:

- `<break time="1s"/>` tells Amazon Polly to pause 1 second between the first two sentences.
- `<sub alias="World Wide Web Consortium">W3C</sub>` tells Amazon Polly to substitute World Wide Web Consortium for the acronym W3C.
- `<amazon:effect name="whispered">Nice job</amazon:effect>` tells Amazon Polly to whisper the second instance of "Nice job." .

  Note
  When you use the AWS CLI, you enclose the input text in quotation marks to differentiate it from the surrounding code. The Amazon Polly console doesn't show you code, so you don't enclose input text in quotation marks when you use it.

5. For **Choose a language and region**, choose **English US**, then choose a voice.
6. To listen to the speech, choose **Listen to speech**.
7. To save the speech file, choose **Download [format]**. If you want to save it in a different format, choose **Change file format**, and choose the format that you want. Then choose **Change** and **Download [format]**.

Using SSML (AWS CLI)

You can use the AWS CLI to synthesize SSML input text. The following examples show how to perform common tasks using the AWS CLI.

Topics

Using SSML With the Synthesize-Speech Command

This example shows how to use the `synthesize-speech` command with an SSML string. When you use the `synthesize-speech` command, you typically provide the following:

- The input text (required)
- Opening and closing tags (required)
- The output format
- A voice

In this example, you specify a simple text string in quotation marks along with the required opening and closing `<speak></speak>` tags.

Important
Although you don't use quotation marks around input text in the Amazon Polly console, you must use them in use the AWS CLI It's also important that you differentiate between the quotation marks around input text and quotations required for individual tags.
For example, you can use standard quotation marks (") to enclose the input text, and single quotation marks (') for interior tags, or vice versa. Either option works for Unix, Linux, and macOS. However, with Windows you must enclose the input text in standard quotations marks and use single quotation marks for the tags.
For all operating systems, you can use standard quotation marks (") to enclose the input text, and single quotation marks (') for interior tags). For example:

```
--text "<speak>Hello <break time='300ms'/> World</speak>"
```

For Unix, Linux, and macOS, you can also use the reverse, with single quotation marks (') enclosing the input text and standard quotation marks (") for interior tags:

```
--text '<speak>Hello <break time="300ms"/> World</speak>'
```

The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

```
aws polly synthesize-speech \
--text-type ssml \
--text '<speak>Hello world</speak>' \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

To hear the synthesized speech, play the resulting `speech.mp3` file using any audio player.

Synthesizing an SSML-enhanced Document

For longer input text, you may find it easier to save your SSML content to a file and simply specify the file name in the `synthesize-speech` command. For example you could save the following to a file called `example.xml`:

```
<?xml version="1.0"?>
<speak version="1.1"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.w3.org/2001/10/synthesis http://www.w3.org/TR/speech-
synthesis11/synthesis.xsd"
       xml:lang="en-US">Hello World</speak>
```

The `xml:lang` attribute specifies `en-US` (US English) as the language of the input text. For information about how the language of the input text and the language of the chosen voice affect the `SynthesizeSpeech` operation, see Improving the Pronunciation of Foreign Words (p. 106).

To run an SSML-enhanced file

1. Save the SSML to a file (for example, `example.xml`).
2. Run the following `synthesize-speech` command from the path where the XML file is stored and specify the SSML file as input by substituting `file:\\example.xml` for the input text. Because this command points to a file instead of containing the actual input text, you don't use quotation marks.

   Note
   The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

   ```
   aws polly synthesize-speech \
   --text-type ssml \
   --text file://example.xml \
   --output-format mp3 \
   --voice-id Joanna \
   speech.mp3
   ```
3. To hear the synthesized speech, play the resulting `speech.mp3` file using any audio player.

Using SSML for Common Amazon Polly Tasks

The following examples show how to use SSML tags to complete common Amazon Polly tasks. For more SSML tags, see Supported SSML Tags (p. 107).

To test the following examples, use the following `synthesize-speech` command with the appropriate SSML-enhanced text:

The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

```
aws polly synthesize-speech \
--text-type ssml \
--text '<speak>Hello <break time="300ms"/> World</speak>' \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

Adding a Pause

To add a pause between words, use the <break> element. The following SSML `synthesize-speech`command uses the `<break>` element to add a 300-millisecond delay between the words "Hello" and "World."

```
<speak>
     Hello <break time="300ms"/> World.
</speak>
```

Controlling Volume, Pitch, and Speed

To control pitch, speaking rate, and speech volume, use the <prosody> element.

- The following synthesize-speech command uses the `<prosody>` element to control volume:

  ```
  <speak>
       <prosody volume="+20dB">Hello world</prosody>
  </speak>
  ```

- The following `synthesize-speech` command uses the `<prosody>` element to control pitch:

  ```
  <speak>
       <prosody pitch="x-high">Hello world.</prosody>
  </speak>
  ```

- The following `synthesize-speech` command uses the `<prosody>` element to specify the speech rate (speaking speed):

  ```
  <speak>
       <prosody rate="x-fast">Hello world.</prosody>
  </speak>
  ```

- You can specify multiple attributes in a `<prosody>` element, as shown in the following examples:

  ```
  <speak>
       <prosody volume="x-loud" pitch="x-high" rate="x-fast">Hello world.</prosody>
  </speak>
  ```

Whispering

To whisper words, use the `<amazon:effect name="whispered">` element. In the following example, the `<amazon:effect name="whispered">` element tells Amazon Polly to whisper "little lamb":

```
<speak>
     Mary has a <amazon:effect name="whispered">little lamb.</amazon:effect>
</speak>
```

To enhance this effect, use the <prosody> element to slightly slow down the whispered speech.

Emphasizing Words

To stress a word or phrase, use the <emphasis> element.

```
<speak>
     <emphasis level="strong">Hello</emphasis> world how are you?
</speak>
```

Specifying How to Say Certain Words

To provide information about the type of text to be spoken, use the <say-as> element.

For instance, in the following SSML, `<say-as>` indicates that the text 4/6 should be interpreted as a date. The attribute `interpret-as="date" format="dm"` indicates that it should be spoken as a date with the format month/day.

You can also use the <say-as> element to tell Amazon Polly to say numbers as fractions, telephone numbers, measurement units, and more.

```
<speak>
     Today is <say-as interpret-as="date" format="md" >4/6</say-as>
</speak>
```

The resulting speech is "Today is June 4th." The `<say-as>` tag describes how the text should be interpreted by providing additional context with the `interpret-as` attribute.

To verify the accuracy of the synthesized speech, play the resulting `speech.mp3` file.

For more information on this element, see Controlling How Special Types of Words Are Spoken (p. 116).

Improving the Pronunciation of Foreign Words

Amazon Polly assumes that the input text is in the same language as the language spoken by the voice you choose. To improve the pronunciation of foreign words within input text, in the `synthesize-speech` call. Specify the target language with the `xml:lang` attribute. This tells Amazon Polly to apply different pronunciation rules for the foreign words that you tag.

The following examples show how to use different combinations of languages in the input text, and how to specify voices and the pronunciation of foreign words. For a complete list of available languages, see Languages Supported by Amazon Polly (p. 17).

In the following example, the voice (Joanna) is a US English voice. By default, Amazon Polly assumes that the input text is in the same language as the voice (in this case, US English). When you use the `xml:lang` tag, Amazon Polly interprets the text as Spanish and the text is spoken as the selected voice would pronounce Spanish words, according to the pronunciation rules of the foreign language. Without this tag, the text is spoken using the pronunciation rules of the selected voice.

```
<speak>
     That restaurant is terrific. <lang xml:lang="es-ES">Mucho gusto.</lang>
</speak>
```

Because the language of the input text is English, Amazon Polly maps the Spanish phonemes to the closest English phonemes. As a result, Joanna speaks the text as a native US speaker who pronounces the works correctly in Spanish, but with a US English accent.

Note
Some languages are more similar than others, and so some language combinations work better than others.

Supported SSML Tags

Amazon Polly supports the following SSML tags:

Action	SSML Tag	Availability with Neural Voices
Adding a Pause (p. 108)	<break>	Full availability
Emphasizing Words (p. 109)	<emphasis>	Not available
Specifying Another Language for Specific Words (p. 109)	<lang>	Full availability
Placing a Custom Tag in Your Text (p. 110)	<mark>	Full availability
Adding a Pause Between Paragraphs (p. 111)	<p>	Full availability
Using Phonetic Pronunciation (p. 111)	<phoneme>	Full availability
Controlling Volume, Speaking Rate, and Pitch (p. 112)	<prosody>	Partial availability
Setting a Maximum Duration for Synthesized Speech (p. 114)	<prosody amazon:max-duration>	Not available
Adding a Pause Between Sentences (p. 116)	<s>	Full availability
Controlling How Special Types of Words Are Spoken (p. 116)	<say-as>	Partial availability
Identifying SSML-Enhanced Text (p. 108)	<speak>	Full availability
Pronouncing Acronyms and Abbreviations (p. 119)	<sub>	Full availability
Improving Pronunciation by Specifying Parts of Speech (p. 119)	<w>	Full availability

Action	SSML Tag	Availability with Neural Voices
Adding the Sound of Breathing (p. 120)	<amazon:auto-breaths>	Not available
Conversational speaking style (p. 122)	<amazon:domain name="conversational">	Select neural voices only
Newscaster speaking style (p. 123)	<amazon:domain name="news">	Select neural voices only
Adding Dynamic Range Compression (p. 123)	<amazon:effect name="drc">	Full availability
Speaking Softly (p. 125)	<amazon:effect phonation="soft">	Not available
Controlling Timbre (p. 125)	<amazon:effect vocal-tract-length>	Not available
Whispering (p. 126)	<amazon: effect name="whispered">	Not available

If you use unsupported SSML tags in either neural or standard format, you will get an error.

Identifying SSML-Enhanced Text

The `<speak>` tag is the root element of all Amazon Polly SSML text. All SSML-enhanced text must be enclosed within a pair of <speak> tags.

```
<speak>Mary had a little lamb.</speak>
```

This tag is supported by both neural and standard TTS formats.

Adding a Pause

<break>

To add a pause to your text, use the <break> tag. You can set a pause based on strength (equivalent to the pause after a comma, a sentence, or a paragraph), or you can set it to a specific length of time in seconds or milliseconds. If you don't specify an attribute to determine the pause length, Amazon Polly uses the default, which is `<break strength="medium">`, which adds a pause the length of a pause after a comma.

`strength` attribute values:

- `none`: No pause. Use `none` to remove a normally occurring pause, such as after a period.
- `x-weak`: Has the same strength as `none`, no pause.
- `weak`: Sets a pause of the same duration as the pause after a comma.
- `medium`: Has the same strength as `weak`.
- `strong`: Sets a pause of the same duration as the pause after a sentence.
- `x-strong`: Sets a pause of the same duration as the pause after a paragraph.

`time` attribute values:

- *`[number]`*`s`: The duration of the pause, in seconds. The maximum duration is `10s`.

- *[number]*ms: The duration of the pause, in milliseconds. The maximum duration is 10000ms.

For example:

```
<speak>
     Mary had a little lamb <break time="3s"/>Whose fleece was white as snow.
</speak>
```

If you don't use an attribute with the break tag, the result varies depending on text:

- If there is no other punctuation next to the break tag, it creates a <break strength="medium"> (comma-length pause).
- If the tag is next to a comma, it upgrades the tag to a <break strength="strong"> (sentence-length pause).
- If the tag is next to a period, it upgrades the tag to <break strength="x-strong"> (paragraph-length pause).

This tag is supported by both neural and standard TTS formats.

Emphasizing Words

<emphasis>

To emphasize words, use the <emphasis> tag. Emphasizing words changes the speaking rate and volume. More emphasis makes Amazon Polly speak the text louder and slower. Less emphasis makes it speak quieter and faster. To specify the degree of emphasis, use the level attribute.

level attribute values:

- Strong: Increases the volume and slows the speaking rate so that the speech is louder and slower.
- Moderate: Increases the volume and slows the speaking rate, but less than strong. Moderate is the default.
- Reduced: Decreases the volume and speeds up the speaking rate. Speech is softer and faster.

Note
The normal speaking rate and volume for a voice falls between the moderate and reduced levels.

For example:

```
<speak>
     I already told you I <emphasis level="strong">really like</emphasis> that person.
</speak>
```

This tag is currently supported only by the standard TTS format.

Specifying Another Language for Specific Words

<lang>

Specify another language for a specific word, phrase, or sentence with the <lang> tag. Foreign language words and phrases are generally spoken better when they are enclosed within a pair of <lang> tags. To specify the language, use the xml:lang attribute. For a complete list of available languages, see Languages Supported by Amazon Polly (p. 17).

Unless you apply the `<lang>` tag, all of the words in the input text are spoken in the language of the voice specified in the `voice-id`. If you apply the `<lang>` tag, the words are spoken in that language.

For example, if the `voice-id` is Joanna (who speaks US English), Amazon Polly speaks the following in the Joanna voice without a French accent:

```
<speak>
     Je ne parle pas français.
</speak>
```

If you use the Joanna voice with the `<lang>` tag, Amazon Polly speaks the sentence in the Joanna voice in American-accented French:

```
<speak>
     <lang xml:lang="fr-FR">Je ne parle pas français.</lang>.
</speak>
```

Because Joanna is not a native French voice, pronunciation is based on her native language, US English. For example, although perfect French pronunciation features an uvual trill /R/ in the word *français*, Joanna's US English voice pronounces this phoneme as the corresponding sound /r/.

If you use the `voice-id` of Giorgio, who speaks Italian, with the following text, Amazon Polly speaks the sentence in Giorgio's voice with an Italian pronunciation:

```
<speak>
     Mi piace Bruce Springsteen.
</speak>
```

If you use the same voice with the following `<lang>` tag, Amazon Polly pronounces Bruce Springsteen in Italian-accented English:

```
<speak>
     Mi piace <lang xml:lang="en-US">Bruce Springsteen.</lang>
</speak>
```

This tag can also be used as a substitute for the optional DefaultLangCode option when synthesizing speech. However, doing so requires that you format your text using SSML.

This tag is supported by both neural and standard TTS formats.

Placing a Custom Tag in Your Text

<mark>

To put a custom tag within the text, use the <mark> tag. Amazon Polly takes no action on the tag, but returns the location of the tag in the SSML metadata. This tag can be anything you want to call out, as long as it maintains the following format:

```
<mark name="tag_name"/>
```

For example, suppose that the tag name is "animal" and the input text is:

```
<speak>
     Mary had a little <mark name="animal"/>lamb.
</speak>
```

Amazon Polly might return the following SSML metadata:

```
{"time":767,"type":"ssml","start":25,"end":46,"value":"animal"}
```

This tag is supported by both neural and standard TTS formats.

Adding a Pause Between Paragraphs

<p>

To add a pause between paragraphs in your text, use the <p> tag. Using this tag provides a longer pause than native speakers usually place at commas or the end of a sentence. Use the <p> tag to enclose the paragraph:

```
<speak>
     <p>This is the first paragraph. There should be a pause after this text is spoken.</
p>
     <p>This is the second paragraph.</p>
</speak>
```

This is equivalent to specifying a pause using <break strength="x-strong"/>.

This tag is supported by both neural and standard TTS formats.

Using Phonetic Pronunciation

<phoneme>

To make Amazon Polly use phonetic pronunciation for specific text, use the <phoneme> tag.

Two attributes are required with the `<phoneme>` tag. They indicate the phonetic alphabet Amazon Polly uses and the phonetic symbols of the corrected pronunciation:

- `alphabet`
  - `ipa`— Indicates that the International Phonetic Alphabet (IPA) will be used.
  - `x-sampa`— Indicates that the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) will be used.
- `ph`
  - Specifies the phonetic symbols for pronunciation. For more information, see Phoneme and Viseme Tables for Supported Languages (p. 18)

With the `<phoneme>` tag, Amazon Polly uses the pronunciation specified by the `ph` attribute instead of the standard pronunciation associated by default with the language used by the selected voice.

For instance, the word "pecan" can be pronounced two ways. In the following example, “pecan” is assigned a different pronunciation in each line. Amazon Polly pronounces pecan as specified in the `ph` attributes, instead of using the default pronunciation.

International Phonetic Alphabet (IPA)

```
<speak>
     You say, <phoneme alphabet="ipa" ph="p##k##n">pecan</phoneme>.
     I say, <phoneme alphabet="ipa" ph="#pi.kæn">pecan</phoneme>.
</speak>
```

Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA)

```
<speak>
     You say, <phoneme alphabet='x-sampa' ph='pI"kA:n'>pecan</phoneme>.
     I say, <phoneme alphabet='x-sampa' ph='"pi.k{n'>pecan</phoneme>.
</speak>
```

Additionally, Mandarin Chinese uses Pinyin for phonetic pronunciation.

Pinyin

```
<speak>
     ## <phoneme alphabet="x-amazon-pinyin" ph="bo2">#</phoneme>#
     ## <phoneme alphabet="x-amazon-pinyin" ph="bao2">#</phoneme>#
</speak>
```

This tag is supported by both neural and standard TTS formats.

Controlling Volume, Speaking Rate, and Pitch

<prosody>

To control the volume, rate, or pitch of your selected voice, use the `prosody` tag.

Volume, speech rate, and pitch are dependent on the specific voice selected. In addition to differences between voices for different languages, there are differences between individual voices speaking the same language. Because of this, while attributes are similar across all languages, there are clear variations from language to language and no absolute value is available.

The `prosody` tag has three attributes, each of which has several available values to set the attribute. Each attribute uses the same syntax:

```
<prosody attribute="value"></prosody>
```

- `volume`
  - `default`: Resets volume to the default level for the current voice.
  - `silent`, `x-soft`, `soft`, `medium`, `loud`, `x-loud`: Sets the volume to a predefined value for the current voice.
  - `+ndB`, `-ndB`: Changes volume relative to the current level. A value of `+0dB` means no change, `+6dB` means approximately twice the current volume, and `-6dB` means approximately half the current volume.

  For example, you could set the volume for a passage as follows:

  ```
  <speak>
       Sometimes it can be useful to <prosody volume="loud">increase the volume
       for a specific speech.</prosody>
  </speak>
  ```

  Or you could set it this way:

  ```
  <speak>
       And sometimes a lower volume <prosody volume="-6dB">is a more effective way of
       interacting with your audience.</prosody>
  </speak>
  ```
- `rate`
  - `x-slow`, `slow`, `medium`, `fast`,`x-fast`. Sets the pitch to a predefined value for the selected voice.

  - `n%`: A non-negative percentage change in the speaking rate. For example, a value of 100% means no change in speaking rate, a value of 200% means a speaking rate twice the default rate, and a value of 50% means a speaking rate of half the default rate. This value has a range of 20-200%.

  For example, you could set the speech rate for a passage as follows:

```
<speak>
     For dramatic purposes, you might wish to <prosody rate="slow">slow up the speaking
     rate of your text.</prosody>
</speak>
```

  Or you could set it this way:

```
<speak>
     Although in some cases, it might help your audience to <prosody rate="85%">slow
     the speaking rate slightly to aid in comprehension.</prosody>
</speak>
```

- `pitch`
  - `default`: Resets pitch to the default level for the current voice.
  - `x-low`, `low`, `medium`, `high`, `x-high`: Sets the pitch to a predefined value for the current voice.
  - `+n%` or `-n%`: Adjusts pitch by a relative percentage. For example, a value of `+0%` means no baseline pitch change, `+5%` gives a little higher baseline pitch, and `-5%` results in a little lower baseline pitch.

  For example, you could set the pitch for a passage as follows:

```
<speak>
     Do you like sythesized speech <prosody pitch="high">with a pitch that is higher
     than normal?</prosody>
</speak>
```

  Or you could set it this way:

```
<speak>
     Or do you prefer your speech <prosody pitch="-10%">with a somewhat lower pitch?</
prosody>
</speak>
```

The <prosody> tag must contain at least one attribute, but can include more within the same tag.

```
<speak>
     Each morning when I wake up, <prosody volume="loud" rate="x-slow">I speak
     quite slowly and deliberately until I have my coffee.</prosody>
</speak>
```

It can also be combined with nested tags, as follows:

```
<speak>
     <prosody rate="85%">Sometimes combining attributes <prosody pitch="-10%">can
     change the impression your audience has of a voice</prosody> as well.</prosody>

</speak>
```

This tag is fully supported by the standard TTS format. The `<prosody volume>` and `<prosody rate>` tags are fully supported by NTTS, but the `<prosody pitch>` tag is not supported.

Setting a Maximum Duration for Synthesized Speech

<prosody amazon:max-duration>

To control how long you want a speech to take when it is synthesized, use the `<prosody>` tag with the `amazon:max-duration` attribute.

The duration of synthesized speech varies slightly, depending on the voice you select. This can make it difficult to match synthesized speech with visuals or other activities that require precise timing. This issue is magnified for translation applications because the time it takes to say particular phrases can vary widely with different languages.

The `<prosody amazon:max-duration>` tag matches synthesized speech to the amount of time you want it to take (the duration).

This tag uses the following syntax:

```
<prosody amazon:max-duration="time duration">
```

With the `<prosody amazon:max-duration>` tag, you can specify duration in either seconds or milliseconds:

- *n*`s`: the maximum duration in seconds
- *n*`ms`: the maximum duration in milliseconds

For example, the following spoken text has a maximum duration of 2 seconds:

```
<speak>
     <prosody amazon:max-duration="2s">
          Human speech is a powerful way to communicate.
     </prosody>
</speak>
```

Text placed within the tag, it doesn't exceed the specified duration. If the chosen voice or language would normally take longer than that duration, Amazon Polly speeds up the speech so that it fits into the specified duration.

If the specified duration is longer than it takes to read the text at a normal rate, Amazon Polly reads the speech normally. It doesn't slow down the speech or add silence, so the resulting audio is shorter than requested.

> **Note**
> Amazon Polly increases the speed no more than 5 times the normal rate. If text is spoken faster than this, it usually doesn't make sense. If a speech cannot fit within your specified duration even when speeded up to the maximum, the audio will be speeded up but will last longer than the specified duration.

You can include a single sentence or multiple sentences within a `<prosody amazon:max-duration>` tag, and you can use multiple `<prosody amazon:max-duration>` tags within your text.

For example:

```
<speak>
     <prosody amazon:max-duration="2400ms">
        Human speech is a powerful way to communicate.
     </prosody>
     <break strength="strong"/>
     <prosody amazon:max-duration="5100ms">
```

```
       Even a simple ‘Hello’ can convey a lot of information depending on the pitch,
 intonation, and tempo.
    </prosody>
    <break strength="strong"/>
    <prosody amazon:max-duration="8900ms">
       We naturally understand this information, which is why speech is ideal for creating
 applications where
       a screen isn’t practical or possible, or simply isn’t convenient.
    </prosody>
</speak>
```

Using the `<prosody amazon:max-duration>` tag can increase latency when Amazon Polly is returns synthesized speech. The degree of latency depends on the passage and its length. We recommend using text comprised of relatively short text passages.

Limitations

There are limitations both in how you use `<prosody amazon:max-duration>` tag and in how it works with other SSML tags:

- The text inside a `<prosody amazon:max-duration>` tag can't be longer than 1500 characters.
- You can't nest `<prosody amazon:max-duration>` tags. If you put one `<prosody amazon:max-duration>` tag inside another, Amazon Polly ignores the inner tag.

  For example, in the following, the `<prosody amazon:max-duration="5s">` tag is ignored:

```
<speak>
     <prosody amazon:max-duration="16s">
          Human speech is a powerful way to communicate.

          <prosody amazon:max-duration="5s">
               Even a simple ‘Hello’ can convey a lot of information depending on the
 pitch, intonation, and tempo.
          </prosody>

          We naturally understand this information, which is why speech is ideal for
 creating applications where a screen isn’t practical or possible, or simply isn’t
 convenient.
     </prosody>
</speak>
```

- You can't use the `<prosody>` tags with the `rate` attribute within a `<prosody amazon:max-duration>` tag. This is because both affect the speed at which text is spoken.

  In the following example, Amazon Polly ignores the `<prosody rate="2">` tag:

```
<speak>
     <prosody amazon:max-duration="7500ms">
          Human speech is a powerful way to communicate.

          <prosody rate="2">
               Even a simple ‘Hello’ can convey a lot of information depending on the
 pitch, intonation, and tempo.
          </prosody>
     </prosody>
</speak>
```

Pauses and `max-duration`

When using `max-duration` tag, you can still insert pauses within your text. However, Amazon Polly includes the length of the pause when calculating the maximum duration for speech. Additionally, Amazon Polly preserves the short pauses that occur where commas and periods are placed within a passage and includes in the maximum duration.

For example, in the following block, the 600 millisecond break and the breaks caused by the commas and periods occur within the 8-second speech:

```
<speak>
     <prosody amazon:max-duration="8s">
          Human speech is a powerful way to communicate.
          <break time="600ms"/>
          Even a simple ‘Hello’ can convey a lot of information depending on the pitch,
 intonation, and tempo.
     </prosody>
</speak>
```

This tag is currently supported only by the standard TTS format.

Adding a Pause Between Sentences

<s>

To add a pause between lines or sentences in your text, use the `<s>` tag. Using this tag has the same effect as:

- Ending a sentence with a period (.)
- Specifying a pause with `<break strength="strong"/>`

Unlike the `<break>` tag, the <s> tag encloses the sentence. This is useful for synthesizing speech that is organized in lines, rather than sentence, such as poetry.

In the following example, the `<s>` tag creates a short pause after both the first and second sentences. The final sentence has no `<s>` tag, but it is also followed by a short pause because it ends with a period.

```
<speak>
     <s>Mary had a little lamb</s>
     <s>Whose fleece was white as snow</s>
     And everywhere that Mary went, the lamb was sure to go.
</speak>
```

This tag is supported by both neural and standard TTS formats.

Controlling How Special Types of Words Are Spoken

<say-as>

Use the `<say-as>` tag with the `interpret-as` attribute to tell Amazon Polly how to say certain characters, words, and numbers. This enables you to provide additional context to eliminate any ambiguity on how Amazon Polly should render the text.

The `say-as` tag uses one attribute, <interpret-as>, which uses a number of possible available values. Each uses the same syntax:

```
<say-as interpret-as="value">[text to be interpreted]</say-as>
```

The following values are available with `interpret-as`:

- `characters` or `spell-out`: Spells out each letter of the text, as in a-b-c.

  Note
  This option is not currently supported using neural voices. If this SSML code is encountered by Amazon Polly at run-time, the affected sentence will be synthesized using the related standard voice. Please note, however, that this sentence will still be billed as if it uses the neural voice.
- `cardinal` or `number`: Interprets the numerical text as a cardinal number, as in 1,234.
- `ordinal`: Interprets the numerical text as an ordinal number, as in 1,234th.
- `digits`: Spells out each digit individually, as in 1-2-3-4.
- `fraction`: Interprets the numerical text as a fraction. This works for both common fractions such as 3/20, and mixed fractions, such as 2 ½. See below for more information.
- `unit`: Interprets a numerical text as a measurement. The value should be either a number or a fraction followed by a unit with no space in between as in `1/2inch`, or by just a unit, as in `1meter`.
- `date`: Interprets the text as a date. The format of the date must be specified with the format attribute. See below for more information.
- `time`: Interprets the numerical text as duration, in minutes and seconds, as in `1'21"`.
- `address`: Interprets the text as part of a street address.
- `expletive`: "Beeps out" the content included within the tag.
- `telephone`: Interprets the numerical text as a 7-digit or 10-digit telephone number, as in `2025551212`. You can also use this value for handle telephone extensions, as in `2025551212x345`. See below for more information.

  Note
  Currently the `telephone` option is not available for all languages. However, it is available for voices speaking English language variants (en-AU, en-GB, en-IN, en-US, and en-GB-WLS), Spanish language variants (es-ES, es-MX, and es-US), French language variants (fr-FR and fr-CA), and Portuguese variants (pt-BR and pt-PT), as well as German (de-DE), Italian (it-IT), Japanese (ja-JP), and Russian (ru-RU). It should also be noted that in some cases, languages such as Arabic (arb) automatically handle the number set as a telephone number and so do not actually implement the `telephone` SSML tag.

Fractions

Amazon Polly interprets values within the `say-as` tag that have the `interpret-as="fraction"` attribute as common fractions. The following is the syntax for fractions:

- *Fraction*

  Syntax: *`cardinal number`*/*`cardinal number`*, such as 2/9.

  For example: `<say-as interpret-as="fraction">2/9</say-as>` is pronounced "two ninths."
- *Non-negative Mixed Number*

  Syntax: *`cardinal number`*+*`cardinal number`*/*`cardinal number`*, such as 3+1/2.

  For example, `<say-as interpret-as="fraction">3+1/2</say-as>` is pronounced "three and a half."

  Note
  There must be a `+` between the "3" and the "1/2". Amazon Polly doesn't support a mixed number without the `+`, such as "3 1/2".

Dates

When `interpret-as` is set to `date`, you also need to indicate the format of the date.

This uses the following syntax:

```
<say-as interpret-as="date" format="format">[date]</say-as>
```

For example:

```
<speak>
     I was born on <say-as interpret-as="date" format="mdy">12-31-1900</say-as>.
</speak>
```

The following formats can be used with the `date` attribute.

- `mdy`: Month-day-year.
- `dmy`: Day-month-year.
- `ymd`: Year-month-day.
- `md`: Month-day.
- `dm`: Day-month.
- `ym`: Year-month.
- `my`: Month-year.
- `d`: Day.
- `m`: Month.
- `y`: Year.
- `yyyymmdd`: Year-month-day. If you use this format, you can make Amazon Polly skip parts of the date using question marks.

  For example, Amazon Polly renders the following as "September 22nd":

  ```
  <say-as interpret-as="date">????0922</say-as>
  ```

  `Format` is not needed.

Telephone

Amazon Polly attempts to interpret the text you provide correctly based on the text’s formatting even without the `<say-as>` tag. For example, if your text includes "202-555-1212," Amazon Polly interprets it as a 10-digit telephone number and says each digit individually, with a brief pause for each dash. In this case, you don't need to use `<say-as interpret-as="telephone">`. However, if you provide the text “2025551212” and want Amazon Polly to say it as a phone number, you would specify `<say-as interpret-as="telephone">`.

The logic for interpreting each element is language-specific. For example, US and UK English differ in how phone numbers are pronounced (in UK English, sequences of the same digit are grouped together, as in "double five" or "triple four"). To see the difference, test the following example with a US voice and with a UK voice:

```
<speak>
     Richard's number is <say-as interpret-as="telephone">2122241555</say-as>
</speak>
```

Except for the "characters" or "spell-out" feature, this tag is supported by both neural and standard TTS formats. If this SSML code is encountered by Amazon Polly at run-time, the affected sentence will be synthesized using the related standard voice. Please note, however, that this sentence will still be billed as if it uses the neural voice.

Pronouncing Acronyms and Abbreviations

<sub>

Use the `<sub>` tag with the `alias` attribute to substitute a different word (or pronunciation) for selected text such as an acronym or abbreviation.

This uses the syntax:

```
<sub alias="new word">abbreviation</sub>
```

In the following example, the name "Mercury" is substituted for the element's chemical symbol to make the audio content clearer.

```
<speak>
     My favorite chemical element is <sub alias="Mercury">Hg</sub>, because it looks so
 shiny.
</speak>
```

This tag is supported by both neural and standard TTS formats.

Improving Pronunciation by Specifying Parts of Speech

<w>

You can use the <w> tag to customize the pronunciation of words by specifying the word’s part of speech or alternate meaning. This is done using the `role` attribute.

This tag uses the following syntax:

```
<w role="attribute">text</w>
```

The following values can be used for the `role` attribute:

To specify the part of speech:

- `amazon:VB`: interprets the word as a verb (present simple).
- `amazon:VBD`: interprets the word as past tense or as a past participle.

For example, depending on its part of speech, the US English pronunciation of the word "read" varies based on the tag:

```
<speak>
     The word <say-as interpret-as="characters">read</say-as> may be interpreted
     as either the present simple form <w role="amazon:VB">read</w>, or the past
     participle form <w role="amazon:VBD">read</w>.
</speak>
```

To specify an alternate meaning:

- `amazon:SENSE_1`: uses the non-default sense of the word when present. For example, the noun "bass" is pronounced differently depending on its meaning. The default meaning is the lowest part of the musical range. The alternate meaning is a species of freshwater fish, also called "bass" but pronounced differently. Using `<w role="amazon:SENSE_1">bass</w>` renders the non-default pronunciation (freshwater fish) for the audio text.

  This difference can be heard if you synthesize the following:

```
<speak>
    Depending on your meaning, the word <say-as interpret-as="characters">bass</say-as>
    may be interpreted as either a musical element: read, or as its alternative meaning,
    a fresh waterfish <w role="amazon:SENSE_1">bass</w>.
</speak>
```

> **Note**
> Some languages may have a different selection of supported parts of speech.

This tag is supported by both neural and standard TTS formats.

Adding the Sound of Breathing

<amazon:breath> and <amazon:auto-breaths>

Natural-sounding speech includes both correctly spoken words and breathing sounds. By adding breathing sounds to synthesized speech, you can make it sound more natural. The `<amazon:breath>` and `<amazon:auto-breaths>` tags provide breaths. You have the following options:

- Manual mode: you set the location, length, and volume of a breath sound within the text
- Automated mode: Amazon Polly automatically inserts breathing sounds into the speech output
- Mixed mode: both you and Amazon Polly add breathing sounds

Manual Mode

In manual mode, you place the `<amazon:breath/>` tag in the input text where you want to locate a breath. You can customize the length and volume of breaths with the `duration` and `volume` attributes, respectively:

- `duration`: Controls the length of the breath. Valid values are: `default`, `x-short`, `short`, `medium`, `long`, `x-long`. The default value is `medium`.
- `volume`: Controls how loud breathing sounds. Valid values are: `default`, `x-soft`, `soft`, `medium`, `loud`, `x-loud`. The default value is `medium`.

> **Note**
> The exact length and volume of each attribute value is dependent on the specific Amazon Polly voice used.

To set a breath sound using the defaults, use `<amazon:breath/>` without attributes.

For example, to use attributes to set the duration and volume for a breath to medium, you would set the attributes as follows:

```
<speak>
     Sometimes you want to insert only <amazon:breath duration="medium" volume="x-loud"/>a
 single breath.
</speak>
```

To use the defaults, you would just use the tag:

```
<speak>
     Sometimes you need <amazon:breath/>to insert one or more average breathes
 <amazon:breath/> so that the
     text sounds correct.
</speak>
```

You can add individual breathing sounds within a passage, as follows:

```
<speak>
     <amazon:breath duration="long" volume="x-loud"/> <prosody rate="120%"> <prosody
 volume="loud">
     Wow! <amazon:breath duration="long" volume="loud"/> </prosody> That was quite fast
 <amazon:breath
     duration="medium" volume="x-loud"/>. I almost beat my personal best time on this
 track. </prosody>
</speak>
```

Automated Mode

In automated mode, you use the `<amazon:auto-breaths>` tag to tell Amazon Polly to automatically create breathing noises at appropriate intervals. You can set the frequency of the intervals, their volume, and their duration. Place the `</amazon:auto-breaths>` tag at the beginning of the text that you want to apply automated breathing to and the close the tag at the end.

> **Note**
> Unlike the manual mode tag, `<amazon:breath/>`, the `<amazon:auto-breaths>` tag requires a closing tag (`</amazon:auto-breaths>`).

You can use the following optional attributes with the `<amazon:auto-breaths>` tag:

- `volume`: Controls how loud the breathing sounds. Valid values are: `default`, `x-soft`, `soft`, `medium`, `loud`, `x-loud`. The default value is `medium`.
- `frequency`: Controls how often breathing sounds occur in the text. Valid values are: `default`, `x-low`, `low`, `medium`, `high`, `x-high`. The default value is `medium`.
- `duration`: Controls the length of the breath. Valid values are: `default`, `x-short`, `short`, `medium`, `long`, `x-long`. The default value is `medium`.

By default, the frequency of breathing sounds depends on the input text. However, breathing sounds often occur after commas and periods.

The following examples show how to use the `<amazon:auto-breaths>` tag. To decide which options to use for your content, copy the applicable examples to the Amazon Polly console and listen to the differences.

- Using automated mode without optional parameters.

```
<speak>
    <amazon:auto-breaths>Amazon Polly is a service that turns text into lifelike
 speech,
    allowing you to create applications that talk and build entirely new categories of
 speech-
    enabled products. Amazon Polly is a text-to-speech service that uses advanced deep
 learning
    technologies to synthesize speech that sounds like a human voice. With dozens of
 lifelike
    voices across a variety of languages, you can select the ideal voice and build
 speech-
```

```
     enabled applications that work in many different countries.</amazon:auto-breaths>
</speak>
```

- Using automated mode with volume control. The unspecified parameters (`duration` and `frequency`) are set to the default values (`medium`).

```
<speak>
     <amazon:auto-breaths volume="x-soft">Amazon Polly is a service that turns text into
 lifelike
     speech, allowing you to create applications that talk and build entirely new
 categories of
     speech-enabled products. Amazon Polly is a text-to-speech service, that uses
 advanced deep
     learning technologies to synthesize speech that sounds like a human voice. With
 dozens of
     lifelike voices across a variety of languages, you can select the ideal voice and
 build speech-
     enabled applications that work in many different countries.</amazon:auto-breaths>
</speak>
```

- Using automated mode with frequency control. The unspecified parameters (`duration` and `volume`) are set to the default values (`medium`).

```
<speak>
     <amazon:auto-breaths frequency="x-low">Amazon Polly is a service that turns text
 into lifelike
     speech, allowing you to create applications that talk and build entirely new
 categories of
     speech-enabled products. Amazon Polly is a text-to-speech service, that uses
 advanced deep
     learning technologies to synthesize speech that sounds like a human voice. With
 dozens of
     lifelike voices across a variety of languages, you can select the ideal voice and
 build speech-
     enabled applications that work in many different countries.</amazon:auto-breaths>
</speak>
```

- Using automated mode with multiple parameters. For the unspecified `Duration` parameter, Amazon Polly uses the default value (`medium`).

```
<speak>
     <amazon:auto-breaths volume="x-loud" frequency="x-low">Amazon Polly is a service
 that turns
     text into lifelike speech, allowing you to create applications that talk and build
 entirely new
     categories of speech-enabled products. Amazon Polly is a text-to-speech service,
 that uses
     advanced deep learning technologies to synthesize speech that sounds like a human
 voice. With
     dozens of lifelike voices across a variety of languages, you can select the ideal
 voice and build
     speech-enabled applications that work in many different countries.</amazon:auto-
breaths>
</speak>
```

This tag is currently supported only by the standard TTS format.

Conversational speaking style

<amazon:domain name="conversational">

The conversational style is available only for the Matthew or Joanna voices, which are available only in American English (en-US) in `Neural` format.

To use the conversational speaking style, you use SSML tags and the following syntax::

```
<amazon:domain name="conversational">text</amazon:domain>
```

For example, you might use the conversational style with the Matthew or Joanna voice as follows:

```
<speak>
<amazon:domain name="conversational">
I really didn't know how this morning was going to start. And if I had known, I think
I might have just stayed in bed. Even with a cat sleeping on my head and my dog deciding
I really didn't need to move my legs and falling asleep across both of them. As it was,
I stayed there as long as my bladder would let me.
</amazon:domain>
</speak>
```

Newscaster speaking style

<amazon:domain name="news">

The newscaster style is available only for the Matthew or Joanna voices, which are available only in American English (en-US) in `Neural` format.

To use the newscaster style, you use SSML tags and the following syntax::

```
<amazon:domain name="news">text</amazon:domain>
```

For example, you might use the newscaster style with the Matthew or Joanna voice as follows:

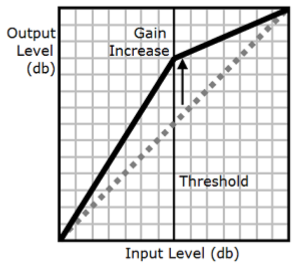
```
<speak>
<amazon:domain name="news">
From the Tuesday, April 16th, 1912 edition of The Guardian newspaper:

The maiden voyage of the White Star liner Titanic, the largest ship ever launched, has
 ended in disaster.

The Titanic started her trip from Southampton for New York on Wednesday. Late on Sunday
 night she struck
an iceberg off the Grand Banks of Newfoundland. By wireless telegraphy she sent out signals
 of distress,
and several liners were near enough to catch and respond to the call.
</amazon:domain>
</speak>
```

Adding Dynamic Range Compression

<amazon:effect name="drc">

Depending on the text, language, and voice used in an audio file, the sounds range from soft to loud. Environmental sounds, such as the sound of a moving vehicle, can often mask the softer sounds, which makes the audio track difficult to hear clearly. To enhance the volume of certain sounds in your audio file, use the dynamic range compression (`drc`) tag.

The `drc` tag sets a midrange "loudness" threshold for your audio, and increases the volume (the gain) of the sounds around that threshold. It applies the greatest gain increase closest to the threshold, and the gain increase is lessened farther away from the threshold.



This makes the middle-range sounds easier to hear in a noisy environment, which makes the entire audio file clearer.

The `drc` tag is a Boolean parameter (it's either present or it isn't). It uses the syntax: `<amazon:effect name="drc">` and is closed with `</amazon:effect>`.

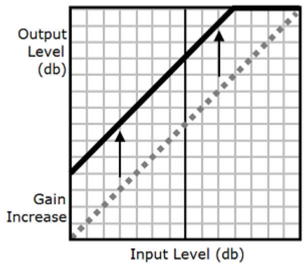
You can use the `drc` tag with any voice or language supported by Amazon Polly. You can apply it to an entire section of the recording, or for only a few words. For example:

```
<speak>
     Some audio is difficult to hear in a moving vehicle, but <amazon:effect name="drc">
 this audio
     is less difficult to hear in a moving vehicle.</amazon:effect>
</speak>
```

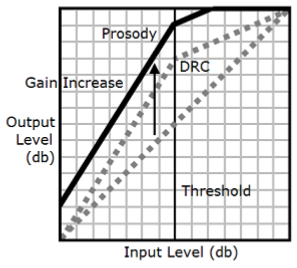
Note
When you use "`drc`" in the `amazon:effect` syntax, it is case-sensitive.

Using `drc` with the `prosody volume` Tag

As the following graphic shows, the `prosody volume` tag evenly increases the volume of an entire audio file from the original level (dotted line) to an adjusted level (solid line). To further increase the volume of certain parts of the file, use the `drc` tag with the `prosody volume` tag. Combining tags doesn't affect the settings of the `prosody volume` tag.



When you use the `drc` and `prosody volume` tags together, Amazon Polly applies the `drc` tag first, increasing the middle-range sounds (those near the threshold). It then applies the `prosody volume` tag and further increases the volume of the entire audio track evenly.



To use the tags together, nest one inside the other. For example:

```
<speak>
     <prosody volume="loud">This text needs to be understandable and loud. <amazon:effect
 name="drc">
     This text also needs to be more understandable in a moving car.</amazon:effect></
prosody>
</speak>
```

In this text, the `prosody volume` tag increases the volume of the entire passage to "loud." The `drc` tag enhances the volume of the middle-range values in the second sentence.

> **Note**
> When using the `drc` and `prosody volume` tags together, use standard XML practices for nesting tags.

This tag is supported by both neural and standard TTS formats.

Speaking Softly

<amazon:effect phonation="soft">

To specify that input text should be spoken in a softer-than-normal voice, use the <amazon:effect phonation="soft"> tag.

This uses the syntax:

```
<amazon:effect phonation="soft">text</amazon:effect>
```

For example, you might use this tag with the Matthew voice as follows:

```
<speak>
     This is Matthew speaking in my normal voice. <amazon:effect phonation="soft">This
     is Matthew speaking in my softer voice.</amazon:effect>
</speak>
```

This tag is currently supported only by the standard TTS format.

Controlling Timbre

<amazon:effect vocal-tract-length>

Timbre is the tonal quality of a voice that helps you tell the difference between voices, even when they have the same pitch and loudness. One of the most important physiological features that contributes to speech timbre is the length of the vocal tract. The vocal tract is a cavity of air that spans from the top of the vocal folds up to the edge of the lips.

To control the timbre of output speech in Amazon Polly, use the `vocal-tract-length` tag. This tag has the effect of changing the length of the speaker's vocal tract, which sounds like a change in the speaker's size. When you increase the `vocal-tract-length`, the speaker sounds physically bigger. When you decrease it, the speaker sounds smaller. You can use this tag with any of the voices in the Amazon Polly Text-to-Speech portfolio.

To change timbre, use the following values:

- `+n%` or `-n%`: Adjusts the vocal tract length by a relative percentage change in the current voice. For example, +4% or -2%. Valid values range from +100% to -50%. Values outside this range are clipped. For example, +111% sounds like +100% and -60% sounds like -50%.

- `n%`: Changes the vocal tract length to an absolute percentage of the tract length of the current voice. For example, 110% or 75%. An absolute value of 110% is equivalent to a relative value of +10%. An absolute value of 100% is the same as the default value for the current voice.

The following example shows how to change the vocal tract length to change timbre:

```
<speak>
     This is my original voice, without any modifications. <amazon:effect vocal-tract-
length="+15%">
     Now, imagine that I am much bigger. </amazon:effect> <amazon:effect vocal-tract-
length="-15%">
     Or, perhaps you prefer my voice when I'm very small. </amazon:effect> You can also
 control the
     timbre of my voice by making minor adjustments. <amazon:effect vocal-tract-
length="+10%">
     For example, by making me sound just a little bigger. </amazon:effect><amazon:effect
     vocal-tract-length="-10%"> Or, making me sound only somewhat smaller. </
amazon:effect>
</speak>
```

Combining Multiple Tags

You can combine the `vocal-tract-length` tag with any other SSML tag that is supported by Amazon Polly. Because timbre (vocal tract length) and pitch are closely connected, you might get the best results by using both the `vocal-tract-length` and the `<prosody pitch>` tags. To produce the most realistic voice, we recommend that you use different percentages of change for the two tags. Experiment with various combinations to get the results you want.

The following example shows how to combine tags.

```
<speak>
     The pitch and timbre of a person's voice are connected in human speech.
     <amazon:effect vocal-tract-length="-15%"> If you are going to reduce the vocal tract
 length,
     </amazon:effect><amazon:effect vocal-tract-length="-15%"> <prosody pitch="+20%"> you
     might consider increasing the pitch, too. </prosody></amazon:effect>
     <amazon:effect vocal-tract-length="+15%"> If you choose to lengthen the vocal tract,
     </amazon:effect> <amazon:effect vocal-tract-length="+15%"> <prosody pitch="-10%">
     you might also want to lower the pitch. </prosody></amazon:effect>
</speak>
```

This tag is currently supported only by the standard TTS format.

Whispering

<amazon:effect name="whispered">

This tag indicates that the input text should be spoken in a whispered voice rather than as normal speech. This can be used with any of the voices in the Amazon Polly Text-to-Speech portfolio.

This uses the following syntax:

```
<amazon:effect name="whispered">text</amazon:effect>
```

For example:

```
<speak>
     <amazon:effect name="whispered">If you make any noise, </amazon:effect>
     she said, <amazon:effect name="whispered">they will hear us.</amazon:effect>
</speak>
```

In this case, the synthesized speech spoken by the character is whispered, but the phrase "she said" is spoken in the normal synthesized speech of the selected Amazon Polly voice.

You can enhance the "whispered" effect by slowing down the prosody rate by up to 10%, depending on the effect you want.

For example:

```
<speak>
     When any voice is made to whisper, <amazon:effect name="whispered">
     <prosody rate="-10%">the sound is slower and quieter than normal speech
     </prosody></amazon:effect>
</speak>
```

When generating speech marks for a whispered voice, the audio stream must also include the whispered voice to ensure that the speech marks match the audio stream.

This tag is currently supported only by the standard TTS format.

Managing Lexicons

Pronunciation lexicons enable you to customize the pronunciation of words. Amazon Polly provides API operations that you can use to store lexicons in an AWS region. Those lexicons are then specific to that particular region. You can use one or more of the lexicons from that region when synthesizing the text by using the `SynthesizeSpeech` operation. This applies the specified lexicon to the input text before the synthesis begins. For more information, see SynthesizeSpeech (p. 231).

> **Note**
> These lexicons must conform with the Pronunciation Lexicon Specification (PLS) W3C recommendation. For more information, see Pronunciation Lexicon Specification (PLS) Version 1.0 on the W3C website.

The following are examples of ways to use lexicons with speech synthesis engines:

- Common words are sometimes stylized with numbers taking the place of letters, as with "g3t sm4rt" (get smart). Humans can read these words correctly. However, a Text-to-Speech (TTS) engine reads the text literally, pronouncing the name exactly as it is spelled. This is where you can leverage lexicons to customize the synthesized speech by using Amazon Polly. In this example, you can specify an alias (get smart) for the word "g3t sm4rt" in the lexicon.
- Your text might include an acronym, such as W3C. You can use a lexicon to define an alias for the word W3C so that it is read in the full, expanded form (World Wide Web Consortium).

Lexicons give you additional control over how Amazon Polly pronounces words uncommon to the selected language. For example, you can specify the pronunciation using a phonetic alphabet. For more information, see Pronunciation Lexicon Specification (PLS) Version 1.0 on the W3C website.

Topics

Applying Multiple Lexicons

You can apply up to five lexicons to your text. If the same grapheme appears in more than one lexicon that you apply to your text, the order in which they are applied can make a difference in the resulting speech. For example, given the following text, "Hello, my name is Bob." and two lexemes in different lexicons that both use the grapheme `Bob`.

LexA

```
<lexeme>
   <grapheme>Bob</grapheme>
   <alias>Robert</alias>
</lexeme>
```

LexB

```
<lexeme>
```

```
   <grapheme>Bob</grapheme>
   <alias>Bobby</alias>
</lexeme>
```

If the lexicons are listed in the order LexA and then LexB, the synthesized speech will be "Hello, my name is Robert." If they are listed in the order LexB and then LexA, the synthesized speech is "Hello, my name is Bobby."

Example – Applying LexA Before LexB

```
aws polly synthesize-speech \
--lexicon-names LexA LexB \
--output-format mp3 \
--text 'Hello, my name is Bob' \
--voice-id Justin \
bobAB.mp3
```

Speech output: "Hello, my name is Robert."

Example – Applying LexB before LexA

```
aws polly synthesize-speech \
--lexicon-names LexB LexA \
--output-format mp3 \
--text 'Hello, my name is Bob' \
--voice-id Justin \
bobBA.mp3
```

Speech output: "Hello, my name is Bobby."

For information about applying lexicons using the Amazon Polly console, see Applying Lexicons Using the Console (Synthesize Speech) (p. 130).

Managing Lexicons Using the Amazon Polly Console

You can use the Amazon Polly console to upload, download, apply, filter, and delete lexicons. The following procedures demonstrate each of these processes.

Uploading Lexicons Using the Console

To use a pronunciation lexicon, you must first upload it. There are two locations on the console from which you can upload a lexicon, the **Text-to-Speech** tab and the **Lexicons** tab.

The following processes describe how to add lexicons that you can use to customize how words and phrases uncommon to the chosen language are pronounced.

To add a lexicon from the Lexicons Tab

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Choose the **Lexicons** tab.
3. Choose **Upload**.

4. Browse to find the lexicon that you want to upload. You can use only PLS files that use the .pls and .xml extensions.
5. Choose **Open**. If a lexicon by the same name (whether a .pls or .xml file) already exists, uploading the lexicon will overwrite the existing lexicon.

To add a lexicon from the Text-to-Speech Tab

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Choose the **Text-to-Speech** tab.
3. Choose **Customize pronunciation of words or phrases using lexicons**, then choose **Upload lexicon**.
4. Browse to find the lexicon that you want to upload. You can use only PLS files that use the .pls and .xml extensions.
5. Choose **Open**. If a lexicon with the same name (whether a .pls or .xml file) already exists, uploading the lexicon will overwrite the existing lexicon.

Applying Lexicons Using the Console (Synthesize Speech)

The following procedure demonstrates how to apply a lexicon to your input text by applying the `W3c.pls` lexicon to substitute "World Wide Web Consortium" for "W3C". If you apply multiple lexicons to your text they are applied in a top-down order with the first match taking precedence over later matches. A lexicon is applied to the text only if the language specified in the lexicon is the same as the language chosen.

You can apply a lexicon to plain text or SSML input.

Example – Applying the W3C.pls Lexicon

To create the lexicon you'll need for this exercise, see Using the PutLexicon Operation (p. 132). Use a plain text editor to create the W3C.pls lexicon shown at the top of the topic. Remember where you save this file.

To apply the W3C.pls lexicon to your input

In this example we introduce a lexicon to substitute "World Wide Web Consortium" for "W3C". Compare the results of this exercise with that of Using SSML (Console) (p. 102) for both US English and another language.

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Do one of the following:

   - Choose the **Plain text** tab and then type or paste this text into the text input box.

   ```
   He was caught up in the game.
   In the middle of the 10/3/2014 W3C meeting
   he shouted, "Score!" quite loudly.
   ```

   - Choose the **SSML** tab and then type or paste this text into the text input box.

   ```
   <speak>He wasn't paying attention.<break time="1s"/>
   In the middle of the 10/3/2014 W3C meeting
   ```

```
he shouted, "Score!" quite loudly.</speak>
```

3. From the **Choose a language and region** list, choose English US, then choose a voice you want to use for this text.
4. Choose **Customize pronunciation of words or phrases using lexicons**.
5. From the list of lexicons, choose `W3C (English, US)`.

   If the `W3C (English, US)` lexicon is not listed, choose **Upload lexicon** and upload it, then choose it from the list. To create this lexicon, see Using the PutLexicon Operation (p. 132).
6. To listen to the speech immediately, choose **Listen to speech**.
7. To save the speech to a file,

   a. Choose **Save speech to MP3**.
   b. To change to a different file format, choose **Change file format**, choose the file format you want, and then choose **Change**.

Repeat the previous steps, but choose a different language and notice the difference in the output.

Filtering the Lexicon List Using the Console

The following procedure describes how to filter the lexicons list so that only lexicons of a chosen language are displayed.

To filter the lexicons listed by language

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Choose the **Lexicons** tab.
3. Choose **Filter**.
4. From the list of languages, choose the language you want to filter on.

   The list displays only the lexicons for the chosen language.

Downloading Lexicons Using the Console

The following process describes how to download one or more lexicons. You can add, remove, or modify lexicon entries in the file and then upload it again to keep your lexicon up-to-date.

To download one or more lexicons

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Choose the **Lexicons** tab.
3. Choose the lexicon or lexicons you want to download.

   a. To download a single lexicon, choose its name from the list.
   b. To download multiple lexicons as a single compressed archive file, select the check box next to each entry in the list that you want to download.
4. Choose **Download**.
5. Open the folder where you want to download the lexicon.
6. Choose **Save**.

Deleting a Lexicon Using the Console

To delete a lexicon

The following process describes how to delete a lexicon. After deleting the lexicon, you must add it back before you can use it again. You can delete one or more lexicons at the same time by selecting the check boxes next to individual lexicons.

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Choose the **Lexicons** tab.
3. Choose one or more lexicons that you want to delete from the list.
4. Choose **Delete**.
5. Choose **Delete** to remove the lexicon from the region or **Cancel** to keep it.

Managing Lexicons Using the AWS CLI

The following topics cover the AWS CLI commands needed to manage your pronunciation lexicons.

Topics

Using the PutLexicon Operation

With Amazon Polly, you can use PutLexicon (p. 223) to store pronunciation lexicons in a specific AWS Region for your account. Then, you can specify one or more of these stored lexicons in your SynthesizeSpeech (p. 231) request that you want to apply before the service starts synthesizing the text. For more information, see Managing Lexicons (p. 128).

This section provides example lexicons and step-by-step instructions for storing and testing them.

Note
These lexicons must conform to the Pronunciation Lexicon Specification (PLS) W3C recommendation. For more information, see Pronunciation Lexicon Specification (PLS) Version 1.0 on the W3C website.

Example 1: Lexicon with One Lexeme

Consider the following W3C PLS-compliant lexicon.

```
<?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>W3C</grapheme>
    <alias>World Wide Web Consortium</alias>
  </lexeme>
</lexicon>
```

Note the following:

- The two attributes specified in the `<lexicon>` element:
  - The `xml:lang` attribute specifies the language code, `en-US`, to which the lexicon applies. Amazon Polly can use this example lexicon if the voice you specify in the `SynthesizeSpeech` call has the same language code (en-US).

    Note
    You can use the `DescribeVoices` operation to find the language code associated with a voice.

  - The `alphabet` attribute specifies `IPA`, which means that the International Phonetic Alphabet (IPA) alphabet is used for pronunciations. IPA is one of the alphabets for writing pronunciations. Amazon Polly also supports the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA).
- The `<lexeme>` element describes the mapping between `<grapheme>` (that is, a textual representation of the word) and `<alias>`.

To test this lexicon, do the following:

1. Save the lexicon as `example.pls`.
2. Run the `put-lexicon` AWS CLI command to store the lexicon (with the name `w3c`), in the us-east-2 region.

   ```
   aws polly put-lexicon \
   --name w3c \
   --content file://example.pls
   ```
3. Run the `synthesize-speech` command to synthesize sample text to an audio stream (`speech.mp3`), and specify the optional `lexicon-name` parameter.

   ```
   aws polly synthesize-speech \
   --text 'W3C is a Consortium' \
   --voice-id Joanna \
   --output-format mp3 \
   --lexicon-names="w3c" \
   speech.mp3
   ```
4. Play the resulting `speech.mp3`, and notice that the word W3C in the text is replaced by World Wide Web Consortium.

The preceding example lexicon uses an alias. The IPA alphabet mentioned in the lexicon is not used. The following lexicon specifies a phonetic pronunciation using the `<phoneme>` element with the IPA alphabet.

```
<?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>pecan</grapheme>
    <phoneme>p##k##n</phoneme>
  </lexeme>
</lexicon>
```

Follow the same steps to test this lexicon. Make sure you specify input text that has word "pecan" (for example, "Pecan pie is delicious").

Example 2: Lexicon with Multiple Lexemes

In this example, the lexeme that you specify in the lexicon applies exclusively to the input text for the synthesis. Consider the following lexicon:

```
<?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>W3C</grapheme>
    <alias>World Wide Web Consortium</alias>
  </lexeme>
  <lexeme>
    <grapheme>W3C</grapheme>
    <alias>WWW Consortium</alias>
  </lexeme>
  <lexeme>
    <grapheme>Consortium</grapheme>
    <alias>Community</alias>
  </lexeme>
</lexicon>
```

The lexicon specifies three lexemes, two of which define an alias for the grapheme W3C as follows:

- The first `<lexeme>` element defines an alias (World Wide Web Consortium).
- The second `<lexeme>` defines an alternative alias (WWW Consortium).

Amazon Polly uses the first replacement for any given grapheme in a lexicon.

The third `<lexeme>` defines a replacement (Community) for the word Consortium.

First, let's test this lexicon. Suppose you want to synthesize the following sample text to an audio file (`speech.mp3`), and you specify the lexicon in a call to `SynthesizeSpeech`.

```
The W3C is a Consortium
```

`SynthesizeSpeech` first applies the lexicon as follows:

- As per the first lexeme, the word W3C is revised as World Wide Web Consortium. The revised text appears as follows:

  ```
  The World Wide Web Consortium is a Consortium
  ```

- The alias defined in the third lexeme applies only to the word Consortium that was part of the original text, resulting in the following text:

```
The World Wide Web Consortium is a Community.
```

You can test this using the AWS CLI as follows:

1. Save the lexicon as `example.pls`.
2. Run the `put-lexicon` command to store the lexicon with name w3c in the us-east-2 region.

```
aws polly put-lexicon \
--name w3c \
--content file://example.pls
```

3. Run the `list-lexicons` command to verify that the w3c lexicon is in the list of lexicons returned.

```
aws polly list-lexicons
```

4. Run the `synthesize-speech` command to synthesize sample text to an audio file (`speech.mp3`), and specify the optional `lexicon-name` parameter.

```
aws polly synthesize-speech \
--text 'W3C is a Consortium' \
--voice-id Joanna \
--output-format mp3 \
--lexicon-names="w3c" \
speech.mp3
```

5. Play the resulting `speech.mp3` file to verify that the synthesized speech reflects the text changes.

Example 3: Specifying Multiple Lexicons

In a call to `SynthesizeSpeech`, you can specify multiple lexicons. In this case, the first lexicon specified (in order from left to right) overrides any preceding lexicons.

Consider the following two lexicons. Note that each lexicon describes different aliases for the same grapheme W3C.

- Lexicon 1: `w3c.pls`

```
<?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>W3C</grapheme>
    <alias>World Wide Web Consortium</alias>
  </lexeme>
</lexicon>
```

- Lexicon 2: `w3cAlternate.pls`

```
<?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>W3C</grapheme>
    <alias>WWW Consortium</alias>
  </lexeme>
</lexicon>
```

Suppose you store these lexicons as `w3c` and `w3cAlternate` respectively. If you specify lexicons in order (`w3c` followed by `w3cAlternate`) in a `SynthesizeSpeech` call, the alias for W3C defined in the first lexicon has precedence over the second. To test the lexicons, do the following:

1. Save the lexicons locally in files called `w3c.pls` and `w3cAlternate.pls`.
2. Upload these lexicons using the `put-lexicon` AWS CLI command.

   - Upload the `w3c.pls` lexicon and store it as `w3c`.

     ```
     aws polly put-lexicon \
     --name w3c \
     --content file://w3c.pls
     ```

   - Upload the `w3cAlternate.pls` lexicon on the service as `w3cAlternate`.

     ```
     aws polly put-lexicon \
     --name w3cAlternate \
     --content file://w3cAlternate.pls
     ```

3. Run the `synthesize-speech` command to synthesize sample text to an audio stream (`speech.mp3`), and specify both lexicons using the `lexicon-name` parameter.

   ```
   aws polly synthesize-speech \
   --text 'PLS is a W3C recommendation' \
   --voice-id Joanna \
   --output-format mp3 \
   --lexicon-names '["w3c","w3cAlternative"]' \
   speech.mp3
   ```

4. Test the resulting `speech.mp3`. It should read as follows:

   ```
   PLS is a World Wide Web Consortium recommendation
   ```

Additional Code Samples for the PutLexicon API

- Java Sample: PutLexicon (p. 147)
- Python (Boto3) Sample: PutLexicon (p. 153)

Using the GetLexicon Operation

Amazon Polly provides the GetLexicon (p. 215) API operation to retrieve the content of a pronunciation lexicon you stored in your account in a specific region.

The following get-lexicon AWS CLI command retrieves the content of the example lexicon.

```
aws polly get-lexicon \
--name example
```

If you don't already have a lexicon stored in your account, you can use the PutLexicon operation to store one. For more information, see Using the PutLexicon Operation (p. 132).

The following is a sample response. In addition to the lexicon content, the response returns the metadata, such as the language code to which the lexicon applies, number of lexemes defined in the lexicon, the Amazon Resource Name (ARN) of the resource, and the size of the lexicon in bytes. The LastModified value is a Unix timestamp.

```
{
    "Lexicon": {
        "Content": "lexicon content in plain text PLS format",
        "Name": "example"
    },
    "LexiconAttributes": {
        "LanguageCode": "en-US",
        "LastModified": 1474222543.989,
        "Alphabet": "ipa",
        "LexemesCount": 1,
        "LexiconArn": "arn:aws:polly:us-east-2:account-id:lexicon/example",
        "Size": 495
    }
}
```

Additional Code Samples for the GetLexicon API

- Java Sample: GetLexicon (p. 146)
- Python (Boto3) Sample: GetLexicon (p. 151)

Using the ListLexicons Operations

Amazon Polly provides the ListLexicons (p. 219) API operation that you can use to get the list of pronunciation lexicons in your account in a specific AWS Region. The following AWS CLI call lists the lexicons in your account in the us-east-2 region.

```
aws polly list-lexicons
```

The following is an example response, showing two lexicons named w3c and tomato. For each lexicon, the response returns metadata such as the language code to which the lexicon applies, the number of lexemes defined in the lexicon, the size in bytes, and so on. The language code describes a language and locale to which the lexemes defined in the lexicon apply.

```
{
    "Lexicons": [
        {
            "Attributes": {
                "LanguageCode": "en-US",
                "LastModified": 1474222543.989,
                "Alphabet": "ipa",
                "LexemesCount": 1,
                "LexiconArn": "arn:aws:polly:aws-region:account-id:lexicon/w3c",
                "Size": 495
            },
            "Name": "w3c"
        },
        {
            "Attributes": {
                "LanguageCode": "en-US",
                "LastModified": 1473099290.858,
                "Alphabet": "ipa",
                "LexemesCount": 1,
                "LexiconArn": "arn:aws:polly:aws-region:account-id:lexicon/tomato",
                "Size": 645
            },
            "Name": "tomato"
        }
    ]
}
```

Additional Code Samples for the ListLexicon API

- Java Sample: ListLexicons (p. 146)
- Python (Boto3) Sample: ListLexicon (p. 152)

Using the DeleteLexicon Operation

Amazon Polly provides the DeleteLexicon (p. 210) API operation to delete a pronunciation lexicon from a specific AWS Region in your account. The following AWS CLI deletes the specified lexicon.

The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

```
aws polly delete-lexicon \
--name example
```

Additional Code Samples for the DeleteLexicon API

- Java Sample: DeleteLexicon (p. 144)
- Python (Boto3) Sample: DeleteLexicon (p. 151)

Creating Long Audio Files

To create TTS files for large passages of text, use Amazon Polly's *asynchronous synthesis* functionality. This uses the three `SpeechSynthesisTask` APIs:

- `StartSpeechSynthesisTask`: starts a new synthesis task.
- `GetSpeechSynthesisTask`: returns details about a previously submitted synthesis task.
- `ListSpeechSynthesisTasks`: lists all submitted synthesis tasks.

The `SynthesizeSpeech` operation produces audio in near-real time, with relatively little latency in most cases. To do this, the operation can only synthesize 3000 characters.

Amazon Polly's Asynchronous Synthesis feature overcomes the challenge of processing a larger text document by changing the way the document is both synthesized and returned. When a synthesis request is made by submitting input text using the `StartSpeechSynthesisTask`, Amazon Polly queues the requests, and then asynchronously processes them in the background as soon as the system resources are available. Amazon Polly then uploads the resulting speech or speech marks stream directly to your (required) Amazon Simple Storage Service (Amazon S3) bucket, and notifies you about the completed file's availability through your (optional) SNS topic.

In this way, all of the functionality except near-real time processing is available for texts of up to 100,000 billable characters (or 200,000 total characters) in length.

To synthesize a document using this method, you must have an Amazon S3 bucket that is writable to which the audio file can be saved. You can be notified when the synthesized audio is ready by providing an optional SNS Topic identifier. When the synthesis task is complete, Amazon Polly will publish a message on that topic. This message may also contain useful error information in cases where the synthesis task didn't succeed. To do this, make sure that the user creating the synthesis task can also publish to the SNS Topic. See the Amazon SNS documentation for more information on how to create and subscribe to an SNS Topic.

Encryption

You can store the output file in an encrypted form in your S3 bucket if desired. To do this, you enable Amazon S3 bucket encryption, which use one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256).

Topics

Setting Up the IAM Policy for Asynchronous Synthesis

In order to use the asynchronous synthesis functionality, you will need an IAM policy that allows the following:

- use of new Amazon Polly operations
- writing to the output S3 bucket

- publishing to the status SNS topic [optional]

The following policy grants only the necessary permissions required for asynchronous synthesis and can be attached to the IAM user.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "polly:StartSpeechSynthesisTask",
        "polly:GetSpeechSynthesisTask",
        "polly:ListSpeechSynthesisTasks"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::bucket-name/*"
    },
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:region:account:topic"
    }
  ]
}
```

Creating Long Audio Files (Console)

You can use the Amazon Polly console to create long speeches using asynchronous synthesis with the same functionality as you can use with the AWS CLI. This is done using the **Text-to-Speech** tab much like any other synthesis.

The other asynchronous synthesis functionality is also available via the console. The **S3 synthesis tasks** tab reflects the `ListSpeechSynthesisTasks` functionality, displaying all tasks saved to the S3 bucket and enabling you to filter them if you want. Clicking on a specific single task shows its details, reflecting `GetSpeechSynthesisTask` functionality.

To synthesize a large text using the Amazon Polly Console

1. Sign in to the AWS Management Console and open the Amazon Polly console at https://console.aws.amazon.com/polly/.
2. Choose the **Text-to-Speech** tab.
3. On either the **Plain Text** tab or the **SSML** tab, type or paste your text into the input box.
4. Choose the language, region, and voice for your text.
5. Choose **Synthesize to S3**.

   Note
   Both the **Download** and **Listen to Speech** options will be greyed out if the text length is above the limits for the real-time `SynthesizeSpeech` operation.
6. If you haven't used asynchronous synthesis previously, the **Change S3 synthesis task settings** box will be displayed so you can choose where to store the output file.

   a. Fill in the name of the destination Amazon S3 bucket.

b. Optionally, fill in the prefix key of the output.

   Note
   The output S3 bucket must be writable.

c. If you want to be notified when the synthesis task is complete, provide the optional SNS Topic identifier.

   Note
   The SNS must be open for publication by the current console user to use this option. For more information, see Amazon Simple Notification Service (SNS)

d. Choose **Synthesize**.

To change the S3 synthesis task settings

1. In the console, on the **Test-to-Speech** tab, choose **Change S3 task settings**.
2. Make any changes you want to the name of the destination Amazon S3 bucket, its prefix key, or the SNS topic identifier.
3. Choose **Synthesize** when done.

To retrieve information on your speech synthesis tasks

1. In the console, choose the **S3 Synthesis Tasks** tab.
2. The tasks are displayed in date order. To filter the tasks, choose **Filter** and then choose what the filter to use.
3. To view the details of a specific task, choose the linked **Task ID**.

Creating Long Audio Files (CLI)

Amazon Polly *asynchronous synthesis* functionality uses three `SpeechSynthesisTask` APIs to work with large amounts of text:

- `StartSpeechSynthesisTask`: starts a new synthesis task.
- `GetSpeechSynthesisTask`: returns details about a previously submitted synthesis task.
- `ListSpeechSynthesisTasks`: lists all submitted synthesis tasks.

Synthesizing large amounts of text (`StartSpeechSynthesisTask`)

When you want to create an audio file larger than one that you can create with the real-time `SynthesizeSpeech`, use the `StartSpeechSynthesisTask` operation. In addition to the arguments needed for the `SynthesizeSpeech` operation, `StartSpeechSynthesisTask` also requires the name of an Amazon S3 bucket. Two other optional arguments are also available: a key prefix for the output file and the ARN for an SNS Topic if you want to receive status notification about the task.

- `OutputS3BucketName`: The name of the Amazon S3 bucket where the synthesis should be uploaded. This bucket should be in the same region as the Amazon Polly service. Additionally, the IAM user being used to make the call should have access to the bucket. [Required]
- `OutputS3KeyPrefix`: Key prefix for the output file. Use this parameter if you want to save the output speech file in a custom directory-like key in your bucket. [Optional]
- `SnsTopicArn`: The SNS topic ARN to use if you want to receive notification about status of the task. This SNS topic should be in the same region as the Amazon Polly service. Additionally, the IAM user being used to make the call should have access to the topic. [Optional]

For example, the following example can be used to run the `start-speech-synthesis-task` AWS CLI command in the US East (Ohio) region:

The following AWS CLI example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

```
aws polly start-speech-synthesis-task \
  --region us-east-2 \
  --endpoint-url "https://polly.us-east-2.amazonaws.com/" \
  --output-format mp3 \
  --output-s3-bucket-name your-bucket-name \
  --output-s3-key-prefix optional/prefix/path/file \
  --voice-id Joanna \
  --text file://text_file.txt
```

This will result in a response that looks similar to this:

```
"SynthesisTask":
{
     "OutputFormat": "mp3",
     "OutputUri": "https://s3.us-east-2.amazonaws.com/your-bucket-name/optional/prefix/
path/file.<task_id>.mp3",
     "TextType": "text",
     "CreationTime": [..],
     "RequestCharacters": [..],
     "TaskStatus": "scheduled",
     "TaskId": [task_id],
     "VoiceId": "Joanna"
 }
```

The `start-speech-synthesis-task` operation returns several new fields:

- `OutputUri`: the location of your output speech file.
- `TaskId`: a unique identifier for the speech synthesis task generated by Amazon Polly.
- `CreationTime`: a timestamp for when the task was initially submitted.
- `RequestCharacters`: the number of billable characters in the task.
- `TaskStatus`: provides information on the status of the submitted task.

  When your task is submitted, the initial status will show `scheduled`. When Amazon Polly starts processing the task, the status will change to `inProgress` and later, to `completed` or `failed`. If the task fails, an error message will be returned when calling either the GetSpeechSynthesisTask or ListSpeechSynthesisTasks operation.

When the task is completed, the speech file is available at the location specified in `OutputUri`.

Retrieving information on your speech synthesis task

You can get information on a task, such as errors, status, and so on, using the `GetSpeechSynthesisTask` operation. To do this, you will need the `task-id` returned by the `StartSpeechSynthesisTask`.

For example, the following example can be used to run the `get-speech-synthesis-task` AWS CLI command:

```
aws polly get-speech-synthesis-task \
--region us-east-2 \
--endpoint-url "https:// polly.us-east-2.amazonaws.com/" \
--task-id task identifier
```

You can also list all speech synthesis tasks that you've run in the current region using the `ListSpeechSynthesisTasks` operation.

For example, the following example can be used to run the `list-speech-synthesis-tasks` AWS CLI command:

```

aws polly list-speech-synthesis-tasks \
--region us-east-2 \
--endpoint-url "https:// polly.us-east-2.amazonaws.com/"
```

Code and Application Examples

This section provides code samples and example applications that you can use to explore Amazon Polly.

Topics

The **Sample Code** topic contains snippets of code organized by programming language and separated into examples for different Amazon Polly functionality. The **Example Application** topic contains applications organized by programming language that can be used independently to explore Amazon Polly.

Before you start using these examples, we recommend that you first read How Amazon Polly Works (p. 3) and follow the steps described in Getting Started with Amazon Polly (p. 4).

Sample Code

This topic contains code samples for various functionality which can be used to explore Amazon Polly.

Sample Code by Programming Language

Java Samples

The following code samples show how to use Java-based applications to accomplish various tasks with Amazon Polly. These samples are not full examples, but can be included in larger Java applications that use the AWS SDK for Java.

Code Snipppets

DeleteLexicon

The following Java code sample show how to use Java-based applications to delete a specific lexicon stored in an AWS region. A lexicon which has been deleted is not available for speech synthesis, nor can it be retrieved using either the `GetLexicon` or `ListLexicon` APIs.

For more information on this operation, see the reference for the `DeleteLexicon` API.

```
package com.amazonaws.polly.samples;

import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.AmazonPollyClientBuilder;
import com.amazonaws.services.polly.model.DeleteLexiconRequest;

public class DeleteLexiconSample {
    private String LEXICON_NAME = "SampleLexicon";

    AmazonPolly client = AmazonPollyClientBuilder.defaultClient();

    public void deleteLexicon() {
        DeleteLexiconRequest deleteLexiconRequest = new
 DeleteLexiconRequest().withName(LEXICON_NAME);

        try {
            client.deleteLexicon(deleteLexiconRequest);
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

DescribeVoices

The following Java code sample show how to use Java-based applications to produce a list of the voices that are available for use when requesting speech synthesis. You can optionally specify a language code to filter the available voices. For example, if you specify en-US, the operation returns a list of all available US English voices.

For more information on this operation, see the reference for the `DescribeVoices` API.

```
package com.amazonaws.polly.samples;

import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.AmazonPollyClientBuilder;
import com.amazonaws.services.polly.model.DescribeVoicesRequest;
import com.amazonaws.services.polly.model.DescribeVoicesResult;

public class DescribeVoicesSample {
    AmazonPolly client = AmazonPollyClientBuilder.defaultClient();

    public void describeVoices() {
        DescribeVoicesRequest allVoicesRequest = new DescribeVoicesRequest();
        DescribeVoicesRequest enUsVoicesRequest = new
 DescribeVoicesRequest().withLanguageCode("en-US");

        try {
            String nextToken;
            do {
                DescribeVoicesResult allVoicesResult =
 client.describeVoices(allVoicesRequest);
                nextToken = allVoicesResult.getNextToken();
                allVoicesRequest.setNextToken(nextToken);

                System.out.println("All voices: " + allVoicesResult.getVoices());
            } while (nextToken != null);

            do {
                DescribeVoicesResult enUsVoicesResult =
 client.describeVoices(enUsVoicesRequest);
                nextToken = enUsVoicesResult.getNextToken();
                enUsVoicesRequest.setNextToken(nextToken);
                System.out.println("en-US voices: " + enUsVoicesResult.getVoices());
            } while (nextToken != null);
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

GetLexicon

The following Java code sample show how to use Java-based applications to produce the content of a specific pronunciation lexicon stored in a AWS Region.

For more information on this operation, see the reference for the `GetLexicon` API.

```
package com.amazonaws.polly.samples;

import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.AmazonPollyClientBuilder;
import com.amazonaws.services.polly.model.GetLexiconRequest;
import com.amazonaws.services.polly.model.GetLexiconResult;

public class GetLexiconSample {
    private String LEXICON_NAME = "SampleLexicon";

    AmazonPolly client = AmazonPollyClientBuilder.defaultClient();

    public void getLexicon() {
        GetLexiconRequest getLexiconRequest = new
 GetLexiconRequest().withName(LEXICON_NAME);

        try {
            GetLexiconResult getLexiconResult = client.getLexicon(getLexiconRequest);
            System.out.println("Lexicon: " + getLexiconResult.getLexicon());
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

ListLexicons

The following Java code sample shows how to use Java-based applications to produce a list of pronunciation lexicons stored in an AWS Region.

For more information on this operation, see the reference for the `ListLexicons` API.

```
package com.amazonaws.polly.samples;

import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.AmazonPollyClientBuilder;
import com.amazonaws.services.polly.model.LexiconAttributes;
import com.amazonaws.services.polly.model.LexiconDescription;
import com.amazonaws.services.polly.model.ListLexiconsRequest;
import com.amazonaws.services.polly.model.ListLexiconsResult;

public class ListLexiconsSample {
    AmazonPolly client = AmazonPollyClientBuilder.defaultClient();

    public void listLexicons() {
        ListLexiconsRequest listLexiconsRequest = new ListLexiconsRequest();
        try {
            String nextToken;
            do {
                ListLexiconsResult listLexiconsResult =
 client.listLexicons(listLexiconsRequest);
                nextToken = listLexiconsResult.getNextToken();
                listLexiconsRequest.setNextToken(nextToken);

                for (LexiconDescription lexiconDescription :
 listLexiconsResult.getLexicons()) {
                    LexiconAttributes attributes = lexiconDescription.getAttributes();
                    System.out.println("Name: " + lexiconDescription.getName()
                            + ", Alphabet: " + attributes.getAlphabet()
                            + ", LanguageCode: " + attributes.getLanguageCode()
                            + ", LastModified: " + attributes.getLastModified()
                            + ", LexemesCount: " + attributes.getLexemesCount()
                            + ", LexiconArn: " + attributes.getLexiconArn()
                            + ", Size: " + attributes.getSize());
                }
            } while (nextToken != null);
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

PutLexicon

The following Java code sample show how to use Java-based applications to store a pronunciation lexicon in an AWS Region.

For more information on this operation, see the reference for the `PutLexicon` API.

```
package com.amazonaws.polly.samples;

import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.AmazonPollyClientBuilder;
import com.amazonaws.services.polly.model.PutLexiconRequest;

public class PutLexiconSample {
    AmazonPolly client = AmazonPollyClientBuilder.defaultClient();

    private String LEXICON_CONTENT = "<?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>test1</grapheme><alias>test2</alias></lexeme>" +
            "</lexicon>";
    private String LEXICON_NAME = "SampleLexicon";

    public void putLexicon() {
        PutLexiconRequest putLexiconRequest = new PutLexiconRequest()
                .withContent(LEXICON_CONTENT)
                .withName(LEXICON_NAME);

        try {
            client.putLexicon(putLexiconRequest);
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

StartSpeechSynthesisTask

The following Java code sample show how to use Java-based applications to synthesize a long speech (up to 100,000 billed characters) and store it directly in an Amazon S3 bucket.

For more information, see the reference for `StartSpeechSynthesisTask` API.

```
package com.amazonaws.parrot.service.tests.speech.task;

import com.amazonaws.parrot.service.tests.AbstractParrotServiceTest;
import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.model.*;
import org.awaitility.Duration;

import java.util.concurrent.TimeUnit;

import static org.awaitility.Awaitility.await;

public class StartSpeechSynthesisTaskSample {

    private static final int SYNTHESIS_TASK_TIMEOUT_SECONDS = 300;
    private static final AmazonPolly AMAZON_POLLY_CLIENT =
 AmazonPollyClientBuilder.defaultClient();
    private static final String PLAIN_TEXT = "This is a sample text to be synthesized.";
    private static final String OUTPUT_FORMAT_MP3 = OutputFormat.Mp3.toString();
    private static final String OUTPUT_BUCKET = "synth-books-buckets";
    private static final String SNS_TOPIC_ARN = "arn:aws:sns:eu-
west-2:561828872312:synthesize-finish-topic";
    private static final Duration SYNTHESIS_TASK_POLL_INTERVAL = Duration.FIVE_SECONDS;
    private static final Duration SYNTHESIS_TASK_POLL_DELAY = Duration.TEN_SECONDS;

    public static void main(String... args) {
        StartSpeechSynthesisTaskRequest request = new StartSpeechSynthesisTaskRequest()
                .withOutputFormat(OUTPUT_FORMAT_MP3)
                .withText(PLAIN_TEXT)
                .withTextType(TextType.Text)
                .withVoiceId(VoiceId.Amy)
                .withOutputS3BucketName(OUTPUT_BUCKET)
                .withSnsTopicArn(SNS_TOPIC_ARN);

        StartSpeechSynthesisTaskResult result =
 AMAZON_POLLY_CLIENT.startSpeechSynthesisTask(request);
        String taskId = result.getSynthesisTask().getTaskId();

        await().with()
                .pollInterval(SYNTHESIS_TASK_POLL_INTERVAL)
                .pollDelay(SYNTHESIS_TASK_POLL_DELAY)
                .atMost(SYNTHESIS_TASK_TIMEOUT_SECONDS, TimeUnit.SECONDS)
                .until(
                        () ->
 getSynthesisTaskStatus(taskId).equals(TaskStatus.Completed.toString())
                );
    }

    private static SynthesisTask getSynthesisTask(String taskId) {
        GetSpeechSynthesisTaskRequest getSpeechSynthesisTaskRequest = new
 GetSpeechSynthesisTaskRequest()
                .withTaskId(taskId);
        GetSpeechSynthesisTaskResult result
 =AMAZON_POLLY_CLIENT.getSpeechSynthesisTask(getSpeechSynthesisTaskRequest);
        return result.getSynthesisTask();
    }
    private static String getSynthesisTaskStatus(String taskId) {
        GetSpeechSynthesisTaskRequest getSpeechSynthesisTaskRequest = new
 GetSpeechSynthesisTaskRequest()
                .withTaskId(taskId);
        GetSpeechSynthesisTaskResult result
 =AMAZON_POLLY_CLIENT.getSpeechSynthesisTask(getSpeechSynthesisTaskRequest);
        return result.getSynthesisTask().getTaskStatus();
    }

}
```

Speech Marks

The following code sample shows how to use Java-based applications to synthesize speech marks for inputed text. This functionality uses the SynthesizeSpeech API.

For more information on this functionality, see Speech Marks (p. 94).

For more information on the API, see the reference for `SynthesizeSpeech` API.

```
package com.amazonaws.polly.samples;

import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.AmazonPollyClientBuilder;
import com.amazonaws.services.polly.model.OutputFormat;
import com.amazonaws.services.polly.model.SpeechMarkType;
import com.amazonaws.services.polly.model.SynthesizeSpeechRequest;
import com.amazonaws.services.polly.model.SynthesizeSpeechResult;
import com.amazonaws.services.polly.model.VoiceId;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;

public class SynthesizeSpeechMarksSample {
    AmazonPolly client = AmazonPollyClientBuilder.defaultClient();

    public void synthesizeSpeechMarks() {
        String outputFileName = "/tmp/speechMarks.json";

        SynthesizeSpeechRequest synthesizeSpeechRequest = new SynthesizeSpeechRequest()
                .withOutputFormat(OutputFormat.Json)
                .withSpeechMarkTypes(SpeechMarkType.Viseme, SpeechMarkType.Word)
                .withVoiceId(VoiceId.Joanna)
                .withText("This is a sample text to be synthesized.");

        try (FileOutputStream outputStream = new FileOutputStream(new
 File(outputFileName))) {
            SynthesizeSpeechResult synthesizeSpeechResult =
 client.synthesizeSpeech(synthesizeSpeechRequest);
            byte[] buffer = new byte[2 * 1024];
            int readBytes;

            try (InputStream in = synthesizeSpeechResult.getAudioStream()){
                while ((readBytes = in.read(buffer)) > 0) {
                    outputStream.write(buffer, 0, readBytes);
                }
            }
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

SynthesizeSpeech

The following Java code sample show how to use Java-based applications to synthesize speech with shorter texts for near-real time processing.

For more information, see the reference for `SynthesizeSpeech` API.

```
package com.amazonaws.polly.samples;

import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.AmazonPollyClientBuilder;
import com.amazonaws.services.polly.model.OutputFormat;
import com.amazonaws.services.polly.model.SynthesizeSpeechRequest;
import com.amazonaws.services.polly.model.SynthesizeSpeechResult;
import com.amazonaws.services.polly.model.VoiceId;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;

public class SynthesizeSpeechSample {
    AmazonPolly client = AmazonPollyClientBuilder.defaultClient();

    public void synthesizeSpeech() {
        String outputFileName = "/tmp/speech.mp3";

        SynthesizeSpeechRequest synthesizeSpeechRequest = new SynthesizeSpeechRequest()
                .withOutputFormat(OutputFormat.Mp3)
                .withVoiceId(VoiceId.Joanna)
                .withText("This is a sample text to be synthesized.");

        try (FileOutputStream outputStream = new FileOutputStream(new
 File(outputFileName))) {
            SynthesizeSpeechResult synthesizeSpeechResult =
 client.synthesizeSpeech(synthesizeSpeechRequest);
            byte[] buffer = new byte[2 * 1024];
            int readBytes;

            try (InputStream in = synthesizeSpeechResult.getAudioStream()){
                while ((readBytes = in.read(buffer)) > 0) {
                    outputStream.write(buffer, 0, readBytes);
                }
            }
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

Python Samples

The following code samples show how to use Python (boto3)-based applications to accomplish various tasks with Amazon Polly. These samples are not intended to be full examples, but can be included in larger Python applications that use the AWS SDK for Python (Boto).

Code Snipppets

DeleteLexicon

The following Python code example uses the AWS SDK for Python (Boto) to delete a lexicon in the region specified in your local AWS configuration. The example deletes only the specified lexicon. It asks you to confirm that you want to proceed before actually deleting the lexicon.

The following code example uses default credentials stored in the AWS SDK configuration file. For information about creating the configuration file, see Step 3.1: Set Up the AWS Command Line Interface (AWS CLI) (p. 7).

For more information on this operation, see the reference for the `DeleteLexicon` API.

```
from argparse import ArgumentParser
from sys import version_info

from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError


# Define and parse the command line arguments
cli = ArgumentParser(description="DeleteLexicon example")
cli.add_argument("name", type=str, metavar="LEXICON_NAME")
arguments = cli.parse_args()

# Create a client using the credentials and region defined in the adminuser
# section of the AWS credentials and configuration files
session = Session(profile_name="adminuser")
polly = session.client("polly")

# Request confirmation
prompt = input if version_info >= (3, 0) else raw_input
proceed = prompt((u"This will delete the \"{0}\" lexicon,"
                  " do you want to proceed? [y,n]: ").format(arguments.name))

if proceed in ("y", "Y"):
    print(u"Deleting {0}...".format(arguments.name))

    try:
        # Request deletion of a lexicon by name
        response = polly.delete_lexicon(Name=arguments.name)
    except (BotoCoreError, ClientError) as error:
        # The service returned an error, exit gracefully
        cli.error(error)

    print("Done.")
else:
    print("Cancelled.")
```

GetLexicon

The following Python code uses the AWS SDK for Python (Boto) to retrieve all lexicons stored in an AWS Region. The example accepts a lexicon name as a command line parameter and fetches that lexicon only, printing out the tmp path where it has been saved locally.

The following code example uses default credentials stored in the AWS SDK configuration file. For information about creating the configuration file, see Step 3.1: Set Up the AWS Command Line Interface (AWS CLI) (p. 7).

For more information on this operation, see the reference for the `GetLexicon` API.

```
from argparse import ArgumentParser
from os import path
from tempfile import gettempdir

from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError

# Define and parse the command line arguments
cli = ArgumentParser(description="GetLexicon example")
cli.add_argument("name", type=str, metavar="LEXICON_NAME")
arguments = cli.parse_args()

# Create a client using the credentials and region defined in the adminuser
# section of the AWS credentials and configuration files
session = Session(profile_name="adminuser")
polly = session.client("polly")

print(u"Fetching {0}...".format(arguments.name))

try:
    # Fetch lexicon by name
    response = polly.get_lexicon(Name=arguments.name)
except (BotoCoreError, ClientError) as error:
    # The service returned an error, exit gracefully
    cli.error(error)

# Get the lexicon data from the response
lexicon = response.get("Lexicon", {})

# Access the lexicon's content
if "Content" in lexicon:
    output = path.join(gettempdir(), u"%s.pls" % arguments.name)
    print(u"Saving to %s..." % output)

    try:
        # Save the lexicon contents to a local file
        with open(output, "w") as pls_file:
            pls_file.write(lexicon["Content"])
    except IOError as error:
        # Could not write to file, exit gracefully
        cli.error(error)
else:
    # The response didn't contain lexicon data, exit gracefully
    cli.error("Could not fetch lexicons contents")

print("Done.")
```

ListLexicon

The following Python code example uses the AWS SDK for Python (Boto) to list the lexicons in your account in the region specified in your local AWS configuration. For information about creating the configuration file, see Step 3.1: Set Up the AWS Command Line Interface (AWS CLI) (p. 7).

For more information on this operation, see the reference for the `ListLexicons` API.

```
import sys
```

```
from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError

# Create a client using the credentials and region defined in the adminuser
# section of the AWS credentials and configuration files
session = Session(profile_name="adminuser")
polly = session.client("polly")

try:
    # Request the list of available lexicons
    response = polly.list_lexicons()
except (BotoCoreError, ClientError) as error:
    # The service returned an error, exit gracefully
    print(error)
    sys.exit(-1)

# Get the list of lexicons in the response
lexicons = response.get("Lexicons", [])
print("{0} lexicon(s) found".format(len(lexicons)))

# Output a formatted list of lexicons with some of the attributes
for lexicon in lexicons:
    print((u" - {Name} ({Attributes[LanguageCode]}), "
           "{Attributes[LexemesCount]} lexeme(s)").format(**lexicon))
```

PutLexicon

The following code sample show how to use Python (boto3)-based applications to store a pronunciation lexicon in an AWS Region.

For more information on this operation, see the reference for the `PutLexicon` API.

Note the following:

- You need to update the code by providing a local lexicon file name and a stored lexicon name.
- The example assumes you have lexicon files created in a subdirectory called `pls`. You need to update the path as appropriate.

The following code example uses default credentials stored in the AWS SDK configuration file. For information about creating the configuration file, see Step 3.1: Set Up the AWS Command Line Interface (AWS CLI) (p. 7).

For more information on this operation, see the reference for the `PutLexicon` API.

```
from argparse import ArgumentParser

from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError

# Define and parse the command line arguments
cli = ArgumentParser(description="PutLexicon example")
cli.add_argument("path", type=str, metavar="FILE_PATH")
cli.add_argument("-n", "--name", type=str, required=True,
                 metavar="LEXICON_NAME", dest="name")
arguments = cli.parse_args()

# Create a client using the credentials and region defined in the adminuser
# section of the AWS credentials and configuration files
session = Session(profile_name="adminuser")
polly = session.client("polly")
```

```
# Open the PLS lexicon file for reading
try:
    with open(arguments.path, "r") as lexicon_file:
        # Read the pls file contents
        lexicon_data = lexicon_file.read()

        # Store the PLS lexicon on the service.
        # If a lexicon with that name already exists,
        # its contents will be updated
        response = polly.put_lexicon(Name=arguments.name,
                                     Content=lexicon_data)
except (IOError, BotoCoreError, ClientError) as error:
    # Could not open/read the file or the service returned an error,
    # exit gracefully
    cli.error(error)

print(u"The \"{0}\" lexicon is now available for use.".format(arguments.name))
```

StartSpeechSynthesisTask

The following Python code example uses the AWS SDK for Python (Boto) to list the lexicons in your account in the region specified in your local AWS configuration. For information about creating the configuration file, see Step 3.1: Set Up the AWS Command Line Interface (AWS CLI) (p. 7).

For more information, see the reference for `StartSpeechSynthesisTask` API.

```
import boto3
import time

polly_client = boto3.Session(
                aws_access_key_id=’’,
    aws_secret_access_key=’’,
    region_name='eu-west-2').client('polly’)

response = polly_client.start_speech_synthesis_task(VoiceId='Joanna',
                OutputS3BucketName='synth-books-buckets',
                OutputS3KeyPrefix='key',
                OutputFormat='mp3',
                Text = 'This is a sample text to be synthesized.')

taskId = response['SynthesisTask']['TaskId']

print "Task id is {} ".format(taskId)

task_status = polly_client.get_speech_synthesis_task(TaskId = taskId)

print task_status
```

SynthesizeSpeech

The following Python code example uses the AWS SDK for Python (Boto) to list the lexicons in your account in the region specified in your local AWS configuration. For information about creating the configuration file, see Step 3.1: Set Up the AWS Command Line Interface (AWS CLI) (p. 7).

For more information on the API, see the reference for `SynthesizeSpeech` API.

```
import boto3

polly_client = boto3.Session(
                aws_access_key_id=,
    aws_secret_access_key=,
    region_name='us-west-2').client('polly')

response = polly_client.synthesize_speech(VoiceId='Joanna',
                OutputFormat='mp3',
                Text = 'This is a sample text to be synthesized.')

file = open('speech.mp3', 'wb')
file.write(response['AudioStream'].read())
file.close()
```

Example Applications

This section contains additional examples, in the form of example applications which can be used to explore Amazon Polly.

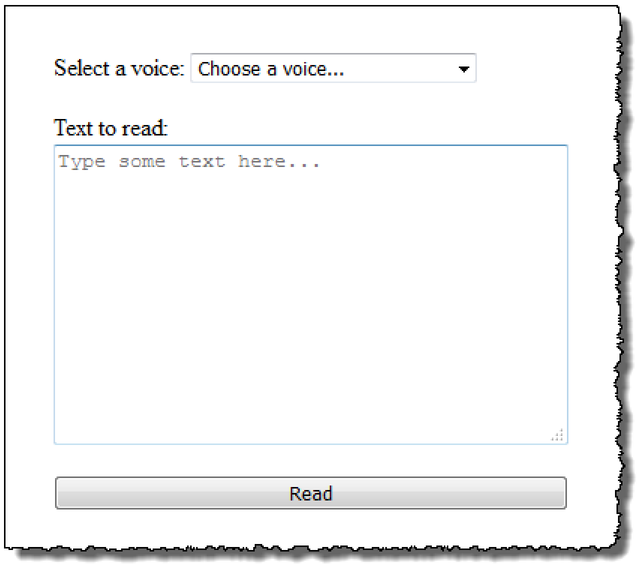
Example Applications by Programming Language

- Python Example (HTML5 Client and Python Server) (p. 155)
- Java Example (p. 165)
- iOS Example (p. 169)
- Android Example (p. 171)

Python Example (HTML5 Client and Python Server)

This example application consists of the following:

- An HTTP 1.1 server using the HTTP chunked transfer coding (see Chunked Transfer Coding)
- A simple HTML5 user interface that interacts with the HTTP 1.1 server (shown below):



The goal of this example is to show how to use Amazon Polly to stream speech from a browser-based HTML5 application. Consuming the audio stream produced by Amazon Polly as the text gets synthesized is the recommended approach for use cases where responsiveness is an important factor (for example, dialog systems, screen readers, etc.).

To run this example application you need the following:

- Web browser compliant with the HTML5 and EcmaScript5 standards (for example, Chrome 23.0 or higher, Firefox 21.0 or higher, Internet Explorer 9.0, or higher)
- Python version greater than 3.0

To test the application

1. Save the server code as `server.py`. For the code, see Python Example: Python Server Code (server.py) (p. 159).
2. Save the HTML5 client code as `index.html`. For the code, see Python Example: HTML5 User Interface (index.html) (p. 156).
3. Run the following command from the path where you saved server.py to start the application (on some systems you might need to use `python3` instead of `python` when running the command).

   ```
   $ python  server.py
   ```

   After the application starts, a URL appears on the terminal.
4. Open the URL shown in the terminal in a web browser.

   You can pass the address and port for the application server to use as a parameter to `server.py`. For more information, run `python server.py -h`.
5. To listen to speech, choose a voice from the list, type some text, and then choose **Read**. The speech starts playing as soon as Amazon Polly transfers the first usable chunk of audio data.
6. To stop the Python server when you're finished testing the application, press Ctrl+C in the terminal where the server is running.

Note
The server creates a Boto3 client using the AWS SDK for Python (Boto). The client uses the credentials stored in the AWS config file on your computer to sign and authenticate the requests to Amazon Polly. For more information on how to create the AWS config file and store credentials, see Configuring the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

Python Example: HTML5 User Interface (index.html)

This section provides the code for the HTML5 client described in Python Example (HTML5 Client and Python Server) (p. 155).

```
<html>

<head>
    <title>Text-to-Speech Example Application</title>
    <script>
        /*
         * This sample code requires a web browser with support for both the
         * HTML5 and ECMAScript 5 standards; the following is a non-comprehensive
         * list of compliant browsers and their minimum version:
         *
         * - Chrome 23.0+
         * - Firefox 21.0+
         * - Internet Explorer 9.0+
```

```
 * - Edge 12.0+
 * - Opera 15.0+
 * - Safari 6.1+
 * - Android (stock web browser) 4.4+
 * - Chrome for Android 51.0+
 * - Firefox for Android 48.0+
 * - Opera Mobile 37.0+
 * - iOS (Safari Mobile and Chrome) 3.2+
 * - Internet Explorer Mobile 10.0+
 * - Blackberry Browser 10.0+
 */

// Mapping of the OutputFormat parameter of the SynthesizeSpeech API
// and the audio format strings understood by the browser
var AUDIO_FORMATS = {
    'ogg_vorbis': 'audio/ogg',
    'mp3': 'audio/mpeg',
    'pcm': 'audio/wave; codecs=1'
};

/**
 * Handles fetching JSON over HTTP
 */
function fetchJSON(method, url, onSuccess, onError) {
    var request = new XMLHttpRequest();
    request.open(method, url, true);
    request.onload = function () {
        // If loading is complete
        if (request.readyState === 4) {
            // if the request was successful
            if (request.status === 200) {
                var data;

                // Parse the JSON in the response
                try {
                    data = JSON.parse(request.responseText);
                } catch (error) {
                    onError(request.status, error.toString());
                }

                onSuccess(data);
            } else {
                onError(request.status, request.responseText)
            }
        }
    };

    request.send();
}

/**
 * Returns a list of audio formats supported by the browser
 */
function getSupportedAudioFormats(player) {
    return Object.keys(AUDIO_FORMATS)
        .filter(function (format) {
            var supported = player.canPlayType(AUDIO_FORMATS[format]);
            return supported === 'probably' || supported === 'maybe';
        });
}

// Initialize the application when the DOM is loaded and ready to be
// manipulated
document.addEventListener("DOMContentLoaded", function () {
    var input = document.getElementById('input'),
        voiceMenu = document.getElementById('voice'),
        text = document.getElementById('text'),
        player = document.getElementById('player'),
        submit = document.getElementById('submit'),
        supportedFormats = getSupportedAudioFormats(player);

    // Display a message and don't allow submitting the form if the
    // browser doesn't support any of the available audio formats
    if (supportedFormats.length === 0) {
        submit.disabled = true;
        alert('The web browser in use does not support any of the' +
              ' available audio formats. Please try with a different' +
              ' one.');
    }

    // Play the audio stream when the form is submitted successfully
    input.addEventListener('submit', function (event) {
        // Validate the fields in the form, display a message if
        // unexpected values are encountered
        if (voiceMenu.selectedIndex <= 0 || text.value.length === 0) {
            alert('Please fill in all the fields.');
        } else {
            var selectedVoice = voiceMenu
                                    .options[voiceMenu.selectedIndex]
                                    .value;

            // Point the player to the streaming server
            player.src = '/read?voiceId=' +
                encodeURIComponent(selectedVoice) +
                '&text=' + encodeURIComponent(text.value) +
                '&outputFormat=' + supportedFormats[0];
            player.play();
        }

        // Stop the form from submitting,
        // Submitting the form is allowed only if the browser doesn't
        // support Javascript to ensure functionality in such a case
        event.preventDefault();
    });

    // Load the list of available voices and display them in a menu
    fetchJSON('GET', '/voices',
        // If the request succeeds
        function (voices) {
            var container = document.createDocumentFragment();

            // Build the list of options for the menu
            voices.forEach(function (voice) {
                var option = document.createElement('option');
                option.value = voice['Id'];
                option.innerHTML = voice['Name'] + ' (' +
                    voice['Gender'] + ', ' +
                    voice['LanguageName'] + ')';
                container.appendChild(option);
            });

            // Add the options to the menu and enable the form field
            voiceMenu.appendChild(container);
            voiceMenu.disabled = false;
        },
        // If the request fails
        function (status, response) {
            // Display a message in case loading data from the server
            // fails
            alert(status + ' - ' + response);
        });
});
```

```
    </script>
    <style>
        #input {
            min-width: 100px;
            max-width: 600px;
            margin: 0 auto;
            padding: 50px;
        }

        #input div {
            margin-bottom: 20px;
        }

        #text {
            width: 100%;
            height: 200px;
            display: block;
        }

        #submit {
            width: 100%;
        }
    </style>
</head>

<body>
    <form id="input" method="GET" action="/read">
        <div>
            <label for="voice">Select a voice:</label>
            <select id="voice" name="voiceId" disabled>
                <option value="">Choose a voice...</option>
            </select>
        </div>
        <div>
            <label for="text">Text to read:</label>
            <textarea id="text" maxlength="1000" minlength="1" name="text"
                    placeholder="Type some text here..."></textarea>
        </div>
        <input type="submit" value="Read" id="submit" />
    </form>
    <audio id="player"></audio>
</body>

</html>
```

Python Example: Python Server Code (server.py)

This section provides the code for the Python server described in Python Example (HTML5 Client and Python Server) (p. 155).

```
"""
Example Python 2.7+/3.3+ Application

This application consists of a HTTP 1.1 server using the HTTP chunked transfer
coding (https://tools.ietf.org/html/rfc2616#section-3.6.1) and a minimal HTML5
user interface that interacts with it.

The goal of this example is to start streaming the speech to the client (the
HTML5 web UI) as soon as the first consumable chunk of speech is returned in
order to start playing the audio as soon as possible.
For use cases where low latency and responsiveness are strong requirements,
this is the recommended approach.
```

```
The service documentation contains examples for non-streaming use cases where
waiting for the speech synthesis to complete and fetching the whole audio stream
at once are an option.

To test the application, run 'python server.py' and then open the URL
displayed in the terminal in a web browser (see index.html for a list of
supported browsers). The address and port for the server can be passed as
parameters to server.py. For more information, run: 'python server.py -h'
"""
from argparse import ArgumentParser
from collections import namedtuple
from contextlib import closing
from io import BytesIO
from json import dumps as json_encode
import os
import sys

if sys.version_info >= (3, 0):
    from http.server import BaseHTTPRequestHandler, HTTPServer
    from socketserver import ThreadingMixIn
    from urllib.parse import parse_qs
else:
    from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer
    from SocketServer import ThreadingMixIn
    from urlparse import parse_qs

from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError

ResponseStatus = namedtuple("HTTPStatus",
                            ["code", "message"])

ResponseData = namedtuple("ResponseData",
                          ["status", "content_type", "data_stream"])

# Mapping the output format used in the client to the content type for the
# response
AUDIO_FORMATS = {"ogg_vorbis": "audio/ogg",
                 "mp3": "audio/mpeg",
                 "pcm": "audio/wave; codecs=1"}
CHUNK_SIZE = 1024
HTTP_STATUS = {"OK": ResponseStatus(code=200, message="OK"),
               "BAD_REQUEST": ResponseStatus(code=400, message="Bad request"),
               "NOT_FOUND": ResponseStatus(code=404, message="Not found"),
               "INTERNAL_SERVER_ERROR": ResponseStatus(code=500, message="Internal server
 error")}
PROTOCOL = "http"
ROUTE_INDEX = "/index.html"
ROUTE_VOICES = "/voices"
ROUTE_READ = "/read"


# Create a client using the credentials and region defined in the adminuser
# section of the AWS credentials and configuration files
session = Session(profile_name="adminuser")
polly = session.client("polly")


class HTTPStatusError(Exception):
    """Exception wrapping a value from http.server.HTTPStatus"""

    def __init__(self, status, description=None):
        """
        Constructs an error instance from a tuple of
        (code, message, description), see http.server.HTTPStatus
        """
        super(HTTPStatusError, self).__init__()
        self.code = status.code
        self.message = status.message
        self.explain = description


class ThreadedHTTPServer(ThreadingMixIn, HTTPServer):
    """An HTTP Server that handle each request in a new thread"""
    daemon_threads = True


class ChunkedHTTPRequestHandler(BaseHTTPRequestHandler):
    """"HTTP 1.1 Chunked encoding request handler"""
    # Use HTTP 1.1 as 1.0 doesn't support chunked encoding
    protocol_version = "HTTP/1.1"

    def query_get(self, queryData, key, default=""):
        """Helper for getting values from a pre-parsed query string"""
        return queryData.get(key, [default])[0]

    def do_GET(self):
        """Handles GET requests"""

        # Extract values from the query string
        path, _, query_string = self.path.partition('?')
        query = parse_qs(query_string)

        response = None

        print(u"[START]: Received GET for %s with query: %s" % (path, query))

        try:
            # Handle the possible request paths
            if path == ROUTE_INDEX:
                response = self.route_index(path, query)
            elif path == ROUTE_VOICES:
                response = self.route_voices(path, query)
            elif path == ROUTE_READ:
                response = self.route_read(path, query)
            else:
                response = self.route_not_found(path, query)

            self.send_headers(response.status, response.content_type)
            self.stream_data(response.data_stream)

        except HTTPStatusError as err:
            # Respond with an error and log debug
            # information
            if sys.version_info >= (3, 0):
                self.send_error(err.code, err.message, err.explain)
            else:
                self.send_error(err.code, err.message)

            self.log_error(u"%s %s %s - [%d] %s", self.client_address[0],
                           self.command, self.path, err.code, err.explain)

        print("[END]")

    def route_not_found(self, path, query):
        """Handles routing for unexpected paths"""
        raise HTTPStatusError(HTTP_STATUS["NOT_FOUND"], "Page not found")

    def route_index(self, path, query):
        """Handles routing for the application's entry point'"""
        try:
            return ResponseData(status=HTTP_STATUS["OK"], content_type="text_html",
                                # Open a binary stream for reading the index
                                # HTML file
                                data_stream=open(os.path.join(sys.path[0],
                                                              path[1:]), "rb"))
        except IOError as err:
            # Couldn't open the stream
            raise HTTPStatusError(HTTP_STATUS["INTERNAL_SERVER_ERROR"],
                                  str(err))

    def route_voices(self, path, query):
        """Handles routing for listing available voices"""
        params = {}
        voices = []

        while True:
            try:
                # Request list of available voices, if a continuation token
                # was returned by the previous call then use it to continue
                # listing
                response = polly.describe_voices(**params)
            except (BotoCoreError, ClientError) as err:
                # The service returned an error
                raise HTTPStatusError(HTTP_STATUS["INTERNAL_SERVER_ERROR"],
                                      str(err))

            # Collect all the voices
            voices.extend(response.get("Voices", []))

            # If a continuation token was returned continue, stop iterating
            # otherwise
            if "NextToken" in response:
                params = {"NextToken": response["NextToken"]}
            else:
                break

        json_data = json_encode(voices)
        bytes_data = bytes(json_data, "utf-8") if sys.version_info >= (3, 0) \
            else bytes(json_data)

        return ResponseData(status=HTTP_STATUS["OK"],
                            content_type="application/json",
                            # Create a binary stream for the JSON data
                            data_stream=BytesIO(bytes_data))

    def route_read(self, path, query):
        """Handles routing for reading text (speech synthesis)"""
        # Get the parameters from the query string
        text = self.query_get(query, "text")
        voiceId = self.query_get(query, "voiceId")
        outputFormat = self.query_get(query, "outputFormat")

        # Validate the parameters, set error flag in case of unexpected
        # values
        if len(text) == 0 or len(voiceId) == 0 or \
                outputFormat not in AUDIO_FORMATS:
            raise HTTPStatusError(HTTP_STATUS["BAD_REQUEST"],
                                  "Wrong parameters")
        else:
            try:
                # Request speech synthesis
                response = polly.synthesize_speech(Text=text,
                                                   VoiceId=voiceId,
                                                   OutputFormat=outputFormat)
            except (BotoCoreError, ClientError) as err:
                # The service returned an error
                raise HTTPStatusError(HTTP_STATUS["INTERNAL_SERVER_ERROR"],
                                    str(err))

            return ResponseData(status=HTTP_STATUS["OK"],
                                content_type=AUDIO_FORMATS[outputFormat],
                                # Access the audio stream in the response
                                data_stream=response.get("AudioStream"))

    def send_headers(self, status, content_type):
        """Send out the group of headers for a successful request"""
        # Send HTTP headers
        self.send_response(status.code, status.message)
        self.send_header('Content-type', content_type)
        self.send_header('Transfer-Encoding', 'chunked')
        self.send_header('Connection', 'close')
        self.end_headers()

    def stream_data(self, stream):
        """Consumes a stream in chunks to produce the response's output'"""
        print("Streaming started...")

        if stream:
            # Note: Closing the stream is important as the service throttles on
            # the number of parallel connections. Here we are using
            # contextlib.closing to ensure the close method of the stream object
            # will be called automatically at the end of the with statement's
            # scope.
            with closing(stream) as managed_stream:
                # Push out the stream's content in chunks
                while True:
                    data = managed_stream.read(CHUNK_SIZE)
                    self.wfile.write(b"%X\r\n%s\r\n" % (len(data), data))

                    # If there's no more data to read, stop streaming
                    if not data:
                        break

                # Ensure any buffered output has been transmitted and close the
                # stream
                self.wfile.flush()

            print("Streaming completed.")
        else:
            # The stream passed in is empty
            self.wfile.write(b"0\r\n\r\n")
            print("Nothing to stream.")

# Define and parse the command line arguments
cli = ArgumentParser(description='Example Python Application')
cli.add_argument(
    "-p", "--port", type=int, metavar="PORT", dest="port", default=8000)
cli.add_argument(
    "--host", type=str, metavar="HOST", dest="host", default="localhost")
arguments = cli.parse_args()

# If the module is invoked directly, initialize the application
if __name__ == '__main__':
    # Create and configure the HTTP server instance
    server = ThreadedHTTPServer((arguments.host, arguments.port),
                                ChunkedHTTPRequestHandler)
    print("Starting server, use <Ctrl-C> to stop...")
    print(u"Open {0}://{1}:{2}{3} in a web browser.".format(PROTOCOL,
                                                            arguments.host,
                                                            arguments.port,
                                                            ROUTE_INDEX))

    try:
        # Listen for requests indefinitely
        server.serve_forever()
    except KeyboardInterrupt:
        # A request to terminate has been received, stop the server
        print("\nShutting down...")
        server.socket.close()
```

Java Example

This example shows how to use Amazon Polly to stream speech from a Java-based application. The example uses the AWS SDK for Java to read the specified text using a voice selected from a list.

The code shown covers major tasks, but does only minimal error checking. If Amazon Polly encounters an error, the application terminates.

To run this example application, you need the following:

- Java 8 Java Development Kit (JDK)
- AWS SDK for Java
- Apache Maven

To test the application

1. Ensure that the JAVA_HOME environment variable is set for the JDK.

   For example, if you installed JDK 1.8.0_121 on Windows at `C:\Program Files\Java\jdk1.8.0_121`, you would type the following at the command prompt:

   ```
   set JAVA_HOME=""C:\Program Files\Java\jdk1.8.0_121""
   ```

   If you installed JDK 1.8.0_121 in Linux at `/usr/lib/jvm/java8-openjdk-amd64` , you would type the following at the command prompt:

   ```
   export JAVA_HOME=/usr/lib/jvm/java8-openjdk-amd64
   ```
2. Set the Maven environment variables to run Maven from the command line.

   For example, if you installed Maven 3.3.9 on Windows at `C:\Program Files\apache-maven-3.3.9`, you would type the following:

   ```
   set M2_HOME=""C:\Program Files\apache-maven-3.3.9""
   set M2=%M2_HOME%\bin
   set PATH=%M2%;%PATH%
   ```

   If you installed Maven 3.3.9 on Linux at `/home/ec2-user/opt/apache-maven-3.3.9`, you would type the following:

   ```
   export M2_HOME=/home/ec2-user/opt/apache-maven-3.3.9
   export M2=$M2_HOME/bin
   export PATH=$M2:$PATH
   ```
3. Create a new directory called `polly-java-demo`.
4. In the `polly-java-demo` directory, create a new file called `pom.xml`, and paste the following code into it:

   ```
   <project xmlns="http://maven.apache.org/POM/4.0.0"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
   maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.amazonaws.polly</groupId>
    <artifactId>java-demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
   ```

```
 <dependencies>
  <!-- https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-polly -->
  <dependency>
   <groupId>com.amazonaws</groupId>
   <artifactId>aws-java-sdk-polly</artifactId>
   <version>1.11.77</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/com.googlecode.soundlibs/jlayer -->
  <dependency>
   <groupId>com.googlecode.soundlibs</groupId>
   <artifactId>jlayer</artifactId>
   <version>1.0.1-1</version>
  </dependency>

 </dependencies>
 <build>
  <plugins>
   <plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>exec-maven-plugin</artifactId>
    <version>1.2.1</version>
    <executions>
     <execution>
      <goals>
       <goal>java</goal>
      </goals>
     </execution>
    </executions>
    <configuration>
     <mainClass>com.amazonaws.demos.polly.PollyDemo</mainClass>
    </configuration>
   </plugin>
  </plugins>
 </build>
</project>
```

5. Create a new directory called `polly` at `src/main/java/com/amazonaws/demos`.
6. In the `polly` directory, create a new Java source file called `PollyDemo.java`, and paste in the following code:

```
package com.amazonaws.demos.polly;

import java.io.IOException;
import java.io.InputStream;

import com.amazonaws.ClientConfiguration;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.polly.AmazonPollyClient;
import com.amazonaws.services.polly.model.DescribeVoicesRequest;
import com.amazonaws.services.polly.model.DescribeVoicesResult;
import com.amazonaws.services.polly.model.OutputFormat;
import com.amazonaws.services.polly.model.SynthesizeSpeechRequest;
import com.amazonaws.services.polly.model.SynthesizeSpeechResult;
import com.amazonaws.services.polly.model.Voice;

import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

public class PollyDemo {
```

```
private final AmazonPollyClient polly;
private final Voice voice;
private static final String SAMPLE = "Congratulations. You have successfully built
this working demo
of Amazon Polly in Java. Have fun building voice enabled apps with Amazon Polly
(that's me!), and always
look at the AWS website for tips and tricks on using Amazon Polly and other great
services from AWS";

public PollyDemo(Region region) {
 // create an Amazon Polly client in a specific region
 polly = new AmazonPollyClient(new DefaultAWSCredentialsProviderChain(),
 new ClientConfiguration());
 polly.setRegion(region);
 // Create describe voices request.
 DescribeVoicesRequest describeVoicesRequest = new DescribeVoicesRequest();

 // Synchronously ask Amazon Polly to describe available TTS voices.
 DescribeVoicesResult describeVoicesResult =
polly.describeVoices(describeVoicesRequest);
 voice = describeVoicesResult.getVoices().get(0);
 }

public InputStream synthesize(String text, OutputFormat format) throws IOException {
 SynthesizeSpeechRequest synthReq =
 new SynthesizeSpeechRequest().withText(text).withVoiceId(voice.getId())
   .withOutputFormat(format);
 SynthesizeSpeechResult synthRes = polly.synthesizeSpeech(synthReq);

 return synthRes.getAudioStream();
 }

public static void main(String args[]) throws Exception {
 //create the test class
 PollyDemo helloWorld = new PollyDemo(Region.getRegion(Regions.US_EAST_1));
 //get the audio stream
 InputStream speechStream = helloWorld.synthesize(SAMPLE, OutputFormat.Mp3);

 //create an MP3 player
 AdvancedPlayer player = new AdvancedPlayer(speechStream,
   javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());

 player.setPlayBackListener(new PlaybackListener() {
  @Override
  public void playbackStarted(PlaybackEvent evt) {
   System.out.println("Playback started");
   System.out.println(SAMPLE);
  }

  @Override
  public void playbackFinished(PlaybackEvent evt) {
   System.out.println("Playback finished");
  }
 });


 // play it!
 player.play();

 }
}
```

7. Return to the `polly-java-demo` directory to clean, compile, and execute the demo:

```
mvn clean compile exec:java
```

iOS Example

The following example uses the iOS SDK for Amazon Polly to read the specified text using a voice selected from a list of voices.

The code shown here covers the major tasks but does not handle errors. For the complete code, see AWS SDK for iOS Amazon Polly demo.

Initialize

```
// Region of Amazon Polly.
let AwsRegion = AWSRegionType.usEast1

// Cognito pool ID. Pool needs to be unauthenticated pool with
// Amazon Polly permissions.
let CognitoIdentityPoolId = "YourCognitoIdentityPoolId"

// Initialize the Amazon Cognito credentials provider.
let credentialProvider = AWSCognitoCredentialsProvider(regionType: AwsRegion,
 identityPoolId: CognitoIdentityPoolId)

// Create an audio player
var audioPlayer = AVPlayer()
```

Get List of Available Voices

```
// Use the configuration as default
AWSServiceManager.default().defaultServiceConfiguration = configuration

// Get all the voices (no parameters specified in input) from Amazon Polly
// This creates an async task.
let task = AWSPolly.default().describeVoices(AWSPollyDescribeVoicesInput())

// When the request is done, asynchronously do the following block
// (we ignore all the errors, but in a real-world scenario they need
// to be handled)
task.continue(successBlock: { (awsTask: AWSTask) -> Any? in
      // awsTask.result is an instance of AWSPollyDescribeVoicesOutput in
      // case of the "describeVoices" method
      let voices = (awsTask.result! as AWSPollyDescribeVoicesOutput).voices

      return nil
})
```

Synthesize Speech

```
// First, Amazon Polly requires an input, which we need to prepare.
// Again, we ignore the errors, however this should be handled in
// real applications. Here we are using the URL Builder Request,
// since in order to make the synthesis quicker we will pass the
// presigned URL to the system audio player.
let input = AWSPollySynthesizeSpeechURLBuilderRequest()

// Text to synthesize
input.text = "Sample text"

// We expect the output in MP3 format
input.outputFormat = AWSPollyOutputFormat.mp3

// Choose the voice ID
input.voiceId = AWSPollyVoiceId.joanna
```

```
// Create an task to synthesize speech using the given synthesis input
let builder = AWSPollySynthesizeSpeechURLBuilder.default().getPreSignedURL(input)

// Request the URL for synthesis result
builder.continueOnSuccessWith(block: { (awsTask: AWSTask<NSURL>) -> Any? in
 // The result of getPresignedURL task is NSURL.
 // Again, we ignore the errors in the example.
 let url = awsTask.result!

 // Try playing the data using the system AVAudioPlayer
 self.audioPlayer.replaceCurrentItem(with: AVPlayerItem(url: url as URL))
 self.audioPlayer.play()

 return nil
})
```

Android Example

The following example uses the Android SDK for Amazon Polly to read the specified text using a voice selected from a list of voices.

The code shown here covers the major tasks but does not handle errors. For the complete code, see the AWS SDK for Android Amazon Polly demo.

Initialize

```
// Cognito pool ID. Pool needs to be unauthenticated pool with
// Amazon Polly permissions.
String COGNITO_POOL_ID = "YourCognitoIdentityPoolId";

// Region of Amazon Polly.
Regions MY_REGION = Regions.US_EAST_1;

// Initialize the Amazon Cognito credentials provider.
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider(
        getApplicationContext(),
        COGNITO_POOL_ID,
        MY_REGION
);

// Create a client that supports generation of presigned URLs.
AmazonPollyPresigningClient client = new AmazonPollyPresigningClient(credentialsProvider);
```

Get List of Available Voices

```
// Create describe voices request.
DescribeVoicesRequest describeVoicesRequest = new DescribeVoicesRequest();

// Synchronously ask Amazon Polly to describe available TTS voices.
DescribeVoicesResult describeVoicesResult = client.describeVoices(describeVoicesRequest);
List<Voice> voices = describeVoicesResult.getVoices();
```

Get URL for Audio Stream

```
// Create speech synthesis request.
SynthesizeSpeechPresignRequest synthesizeSpeechPresignRequest =
        new SynthesizeSpeechPresignRequest()
        // Set the text to synthesize.
        .withText("Hello world!")
        // Select voice for synthesis.
        .withVoiceId(voices.get(0).getId()) // "Joanna"
        // Set format to MP3.
        .withOutputFormat(OutputFormat.Mp3);

// Get the presigned URL for synthesized speech audio stream.
URL presignedSynthesizeSpeechUrl =
        client.getPresignedSynthesizeSpeechUrl(synthesizeSpeechPresignRequest);
```

Play Synthesized Speech

```
// Use MediaPlayer: https://developer.android.com/guide/topics/media/mediaplayer.html

// Create a media player to play the synthesized audio stream.
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
```

```
try {
    // Set media player's data source to previously obtained URL.
    mediaPlayer.setDataSource(presignedSynthesizeSpeechUrl.toString());
} catch (IOException e) {
    Log.e(TAG, "Unable to set data source for the media player! " + e.getMessage());
}

// Prepare the MediaPlayer asynchronously (since the data source is a network stream).
mediaPlayer.prepareAsync();

// Set the callback to start the MediaPlayer when it's prepared.
mediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mp) {
        mp.start();
    }
});

// Set the callback to release the MediaPlayer after playback is completed.
mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mp) {
 mp.release();
    }
});
```

Amazon Polly for Windows SAPI Compliant Applications

With the Amazon Polly for Windows plugin, Amazon Polly provides greatly expanded speech capability for Speech Application Programming Interface (SAPI)-compliant Windows applications. SAPI is a Microsoft Windows API that enables desktop applications to generate speech.

Windows provides one male and one female voice that can be used with for the SAPI API. With the Amazon Polly for Windows plug-in, the Windows application becomes an AWS client, enabling you to use all Amazon Polly voices, including both standard and neural voices, and the newscaster speaking style. The Amazon Polly for Windows plugin only supports 64-bit SAPI applications.

Amazon Polly for Windows supports all Amazon Polly, features such as Speech Synthesis Markup Language (SSML) . When using SSML with the plugin, the standard functionality is available and the standard limits apply. For more information about using SSML with Amazon Polly, see Generating Speech from SSML Documents (p. 100). For more information about limits, see Limits in Amazon Polly (p. 186).

Topics

Installing and Configuring Amazon Polly for Windows (SAPI)

To install and configure the Amazon Polly for Windows (SAPI) plugin, you need an AWS account. If you don't have an account, see Step 1.1: Sign up for AWS (p. 4).

Topics

Create an IAM User for the AWS Client

Before connecting the AWS client to your AWS account, you need to create an Identity and Access Management (IAM) user for the client, then attach a permissions policy to that user. An *IAM user* is a person or application in an AWS account that can make API calls to AWS products.

The AWS client uses the Amazon Polly service, so it needs the `AmazonPollyReadOnlyAccess` permissions policy .

To create an IAM user

1. Sign in to the AWS Management Console and open the IAM console .
2. Choose **Users**.
3. Choose **Add User**.
4. For **User Name**, enter `polly-windows-user`.
5. For **Access Type**, choose **Programmatic access**, then choose **Next: Permissions**.
6. Choose **Attach existing policies directly**
7. In the search box, enter `polly`.
8. Choose **AmazonPollyReadOnlyAccess**.
9. Choose **Next: Tags**.
10. Choose **Next: Review**.
11. Choose **Create User**.
12. Record the access key ID and secret access key. You need them to configure the AWS client.

    Important
    This is the only time you can access these keys, so be sure to record them.

Install the AWS CLI for Windows

The AWS Command Line Interface (AWS CLI) is an open source tool that enables you to interact with AWS services using commands from the Windows command prompt. To install the AWS CLI, see Installing the AWS CLI on Windows in the *AWS Command Line Interface User's Guide*.

Create a Profile for the AWS Client

The Amazon Polly for Windows plugin requires an AWS profile called `polly-windows`. This profile ensures that the Amazon Polly engine use the correct account.

To create the `polly-windows` profile for the AWS client

1. In Windows, open a command prompt.
2. At the command prompt, run the `aws configure --profile polly-windows` command.
3. When prompted for the **AWS Access Key ID** and **AWS Secret Access Key**, enter the values that you saved when you created the IAM user.
4. For **Default region**, enter the name of the desired AWS Region in lower-case letters.

    Note
    Not all Amazon Polly voices and features are available in all AWS Regions. For more information, see Feature and Region Compatibility (p. 87).
5. For **Default output format**, press **Enter**.
6. At the command prompt, verify the profile by running the `aws --profile polly-windows polly describe-voices` command. If you have successfully created the profile, the AWS CLI displays a list the available Amazon Polly names.

Install the Amazon Polly for Windows Plugin

Install the Amazon Polly for Windows plugin by downloading it from Amazon Simple Storage Service (Amazon S3) and running the installer. After you install the plugin, you can use Amazon Polly voices in all Windows application that use the SAPI API.

Note
If you are planning to use a neural voice in your SAPI application, ensure that your client is configured for an AWS Region that supports NTTS before installing the for Windows plugin. For more information, see Feature and Region Compatibility (p. 87).

To install the Amazon Polly for Windows plugin

1. In Windows, download the plugin.
2. Run `setup.exe`.
3. In the **Control Panel**, search for `speech settings`.
4. Under **Text-to-speech**, for **Voice**, choose an Amazon Polly voice.
5. Choose **Preview Voice**.

Using Amazon Polly in Applications

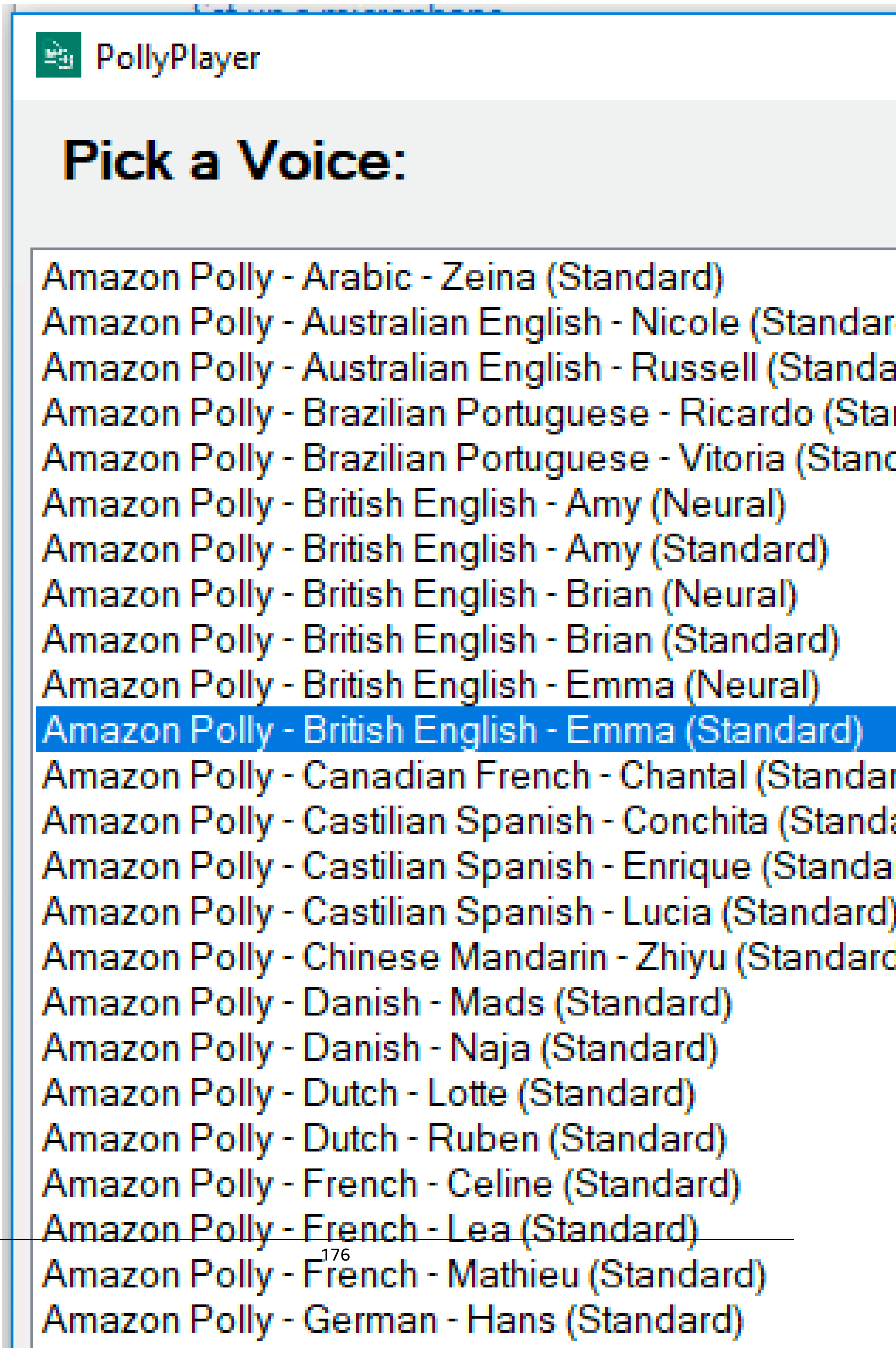
After you've installed the Amazon Polly for Windows plugin, the Amazon Polly voices can be accessed by any Windows application that implements Windows SAPI. To use a voice, choose it from the list of Amazon Polly voices in the application. For a complete list of available voices, see Available Voices (p. 12).

For an example of how to use Amazon Polly voices in a Windows application (Adobe Captivate), see Using Amazon Polly in Windows Applications on the AWS Machine Learning Blog.

To use Amazon Polly voices in a Windows application, you can use PollyPlayer. *PollyPlayer* is a simple Windows speech synthesis application that's installed when you install the Amazon Polly for Windows plugin.

To use PollyPlayer

1. On the Windows Start menu, under **Amazon Polly for Windows**, choose **PollyPlayer**.
2. In the PollyPlayer application, choose a voice, then enter the text that you want to hear.



Pick a Voice:

Amazon Polly - Arabic - Zeina (Standard)
Amazon Polly - Australian English - Nicole (Standar
Amazon Polly - Australian English - Russell (Standa
Amazon Polly - Brazilian Portuguese - Ricardo (Star
Amazon Polly - Brazilian Portuguese - Vitoria (Stanc
Amazon Polly - British English - Amy (Neural)
Amazon Polly - British English - Amy (Standard)
Amazon Polly - British English - Brian (Neural)
Amazon Polly - British English - Brian (Standard)
Amazon Polly - British English - Emma (Neural)
Amazon Polly - British English - Emma (Standard)
Amazon Polly - Canadian French - Chantal (Standar
Amazon Polly - Castilian Spanish - Conchita (Standa
Amazon Polly - Castilian Spanish - Enrique (Standa
Amazon Polly - Castilian Spanish - Lucia (Standard)
Amazon Polly - Chinese Mandarin - Zhiyu (Standard
Amazon Polly - Danish - Mads (Standard)
Amazon Polly - Danish - Naja (Standard)
Amazon Polly - Dutch - Lotte (Standard)
Amazon Polly - Dutch - Ruben (Standard)
Amazon Polly - French - Celine (Standard)
Amazon Polly - French - Lea (Standard)
Amazon Polly - French - Mathieu (Standard)
Amazon Polly - German - Hans (Standard)

3. To hear the text, choose **Say It**.

The AWS for WordPress Plugin

With the AWS for WordPress plugin, you can set up several AWS services, including Amazon Polly. With Amazon Polly, you can provide visitors to your WordPress website audio recordings of your content. Use the plugin to create audio files in any of the voices and languages supported by Amazon Polly. Your visitors can stream the audio at their convenience using inline audio players and mobile applications.

You can configure the plugin to do the following:

- Automatically create audio recordings for new content upon publication, or choose to create recordings for individual posts
- Create audio recordings of your archived content
- Use the Amazon Pollycast RSS feed to podcast audio content

Topics

Installation Prerequisites

To use the AWS for WordPress plugin, you need an AWS account, an AWS Identity and Access Management (IAM) user, and a WordPress website.

Topics

Creating an AWS Account

If you have an AWS account already, you can skip this section. Otherwise, create one.

To create an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Creating an IAM User

To use the AWS for WordPress plugin, you must create an *IAM user* for the plugin. An IAM user is a person or application under an AWS account that has permission to make API calls to AWS services.

Note
If you don't use WordPress.com and instead have a self-hosted WordPress website on Amazon EC2, you can use an IAM role instead of an IAM user. For more information, see IAM Roles for Amazon EC2 in the *Amazon EC2 User Guide*.

The following procedure contains the steps to create an IAM *policy*, and then attach it to the IAM user. An IAM policy is a document that defines the permissions that apply to the user.

To create an IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**. Then choose **Create policy**.
3. Choose **JSON**.
4. Delete everything in the policy text box, and then paste or enter the following JSON policy into the text box:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "acm:DeleteCertificate",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-1"
                }
            }
        }
    ]
}
```

5. Choose **Review policy**.
6. On the **Review policy** page, do the following:
   a. For **Name**, enter **`AWSForWordPressDeleteCert`**.
   b. Choose **Create policy**.
7. In the navigation pane, choose **Users**. Then choose **Add user**.
8. On the **Set user details** page, do the following:
   a. For **User name**, enter **`AWSForWordPressPlugin`**.
   b. For **Access type**, choose **Programmatic access**.
   c. Choose **Next: Permissions**.
9. On the **Set permissions** page, do the following:
   a. Choose **Attach existing policies directly**.
   b. In the search box, enter **`WordPress`**, and then select the check boxes next to **AWSForWordPressPolicy** and **AWSForWordPressDeleteCert**. Make sure to select the check boxes for both WordPress policies.

      Note
      The *AWSForWordPressPolicy* is an AWS managed policy that gives the user permission to use all the features included in the AWS for WordPress plugin. When new features are added to the plugin, AWS will update this policy to include the permissions necessary to use the new features.
   c. Choose **Next: Tags**.

10. Choose **Next: Review**.
11. Choose **Create user**.
12. Choose **Download .csv** to save the user's credentials (access key ID and secret access key) to your computer. You need them to configure the AWS for WordPress plugin.

    Important
    This is the only time that you can save the user's secret access key, so make sure to save it now.

Creating a WordPress Website

If you have a WordPress website already, you can skip ahead to Installing and Configuring the Plugin (p. 180).

If you don't have a WordPress website, you can create one using WordPress.com. To use the AWS for WordPress plugin, you need a WordPress.com Business or eCommerce plan.

You can also install the WordPress software on your own web server, using Amazon Lightsail, Amazon EC2, or another web hosting platform. Hosting your own WordPress website involves more steps than using WordPress.com, and requires the ability to configure and manage a web server, a load balancer, DNS records, and web server certificates.

Regardless of how you set up your WordPress website, you need the following before you can use the AWS for WordPress plugin:

- Your website must have its own *domain name*. A domain name, also known as a *web address* or a *URL (uniform resource locator)*, is the address that visitors use to go to your website. For example, Amazon's domain name is *amazon.com*. In this topic, we use *example.com* as a generic example domain name, but you need a custom domain name for your website.
- Your website must work using HTTPS. This is a security best practice, and the plugin assumes that your website works using HTTPS. To check, go to your website's address using HTTPS (for example, ***https://****example.com*) and make sure that your website displays correctly.

When your website has a domain name and works using HTTPS, proceed to the following section.

Installing and Configuring the Plugin

Before you install the plugin, make sure to complete the prerequisites (p. 178).

To install the plugin

1. Log in to the admin dashboard for your WordPress website, also known as **WP Admin**.
2. Choose **Plugins**.
3. Choose **Add New**.
4. In the search box, enter **`AWS for WordPress`**.
5. Find the **AWS for WordPress** plugin. Choose **Install Now**, and then choose **Activate**.

Configuring the Plugin

Once you've installed the **AWS for WordPress** plugin, configure it to enable podcasting, alternative storage locations, and other options.

Note
In the following procedure, command and field names might differ slightly from the names used in WordPress.

To configure the plugin

1. On the **WordPress Admin** page, choose **Settings**.
2. Configure the plugin using the following options:
   - **AWS access key and AWS secret key**—These AWS credentials allow the plugin to use Amazon Polly and Amazon Simple Storage Service (Amazon S3). Type the AWS access and secret keys that you created earlier. If you are hosting your WordPress site on Amazon EC2, you can use an IAM role instead of credentials. In that case, leave these two fields blank.
   - **Sample rate**—The sample rate for the audio files that will be generated, in Hz. Higher sampling rates produce higher-quality audio.
   - **Voice name**—The Amazon Polly voice to use in the audio file.

     Note
     Remember that the choice of voice is determined by the language you're using. For a complete set of available voices, see Available Voices (p. 12)
   - **Player position**—Where to position the audio player on the website. You can put it before or after the post, or not use it at all. If you want to make your files available as podcasts by using Amazon Pollycast, don't display the audio player.
   - **New post default**—Specifies whether Amazon Polly should automatically create an audio file for all new posts. Choose this option if you want Amazon Polly to create an audio file for each new post.
   - **Autoplay**—Specifies whether the audio player automatically starts playing the audio for a post when a visitor visits it.
   - **Store audio in Amazon S3**—If you want to store audio files in an S3 bucket instead of on your web server, choose this option. Amazon Polly creates the bucket for you. For more information and pricing, see Amazon S3.
   - **Amazon CloudFront (CDN) domain name**—If you want to broadcast your audio files with Amazon CloudFront, provide the name of your CloudFront domain. The plugin uses the domain to stream audio. If you don't already have a domain, create one in Amazon CloudFront.
   - **ITunes category**—The category for your podcast. Choosing a category makes it easier for podcast users to find your podcast in the podcast catalog.
   - **ITunes explicit**—Specifies whether to enable Amazon Pollycast podcasting with the ITunes explicit setting..
   - **Bulk update all posts**—If you want to convert all your previous WordPress posts to use the **AWS for WordPress** plugin, choose this option.
3. Choose **Save Changes**.

Customizing Your WordPress Page

You can use several options for customizing your WordPress content to make it work more effectively with the Amazon Polly WordPress plugin:

- Adjust the plugin settings to fine tune the audio files (p. 182)
- Use SSML in your content to modify how it will be spoken (p. 182)
- Use Audio Only and Word Only tags in your content (p. 182)

Adjust the plugin settings to fine tune the audio files

The **AWS for WordPress** plugin settings have three options that can help you customize the sound of your WordPress text for the audio file:

- Voice name: The voice name and language selected enables you to select the gender of the Amazon Polly voice. Specific voices are available for each language, and you have several options within each gender in many languages. For more information, see Voices in Amazon Polly (p. 12).
- Automated breaths: If enabled, Amazon Polly will automatically insert breathing sounds into your content at appropriate locations. Setting this option will only enable you to use automated breaths and not manually set them. For more information, see automated breathing.
- Audio speed: This option enables you to alter the speed of delivery of the audio version of your content, ranging from 20% to 200% of the default speed of the voice.

Use SSML in your content to modify how it will be spoken

Amazon Polly itself supports multiple SSML tags that enable you to control many aspects of how Amazon Polly generates speech from the text you provide. (For more information about SSML and Amazon Polly, see Supported SSML Tags (p. 107). A few of these tags are echoed in the plug-in settings when configuring your plug-in. Currently, however, only the <break> tag is available for direct use in the **AWS for WordPress** plugin.

The <break> tag enables you to add a pause into the spoken version of your text. You can adjust the length of this pause to meet your particular needs. The default length of the pause is equivalent to the pause following a comma. For more information on the <break> tag, see Supported SSML Tags (p. 107).

To use SSML tags to enhance your WordPress text, the Enable SSML support option must be selected in Amazon Polly Settings on the **WordPress Admin** page. The **Store audio in Amazon S3 option** must also be selected because audio files with SSML tags need to be stored in an S3 bucket.

Use Audio Only and Word Only tags in your content

Sometimes you want to add something to your audio podcast but don't want it to display in the browser. Or you want something to show in the browser but not include it in the audio file. This can be done with the `Audio Only` and `Word Only` tags, which you place in the WordPress content to control which part of the text will be displayed or spoken.

To convert text to audio but not display it in the browser

1. Isolate the selected text on your WordPress page by placing an empty line above and below it.
2. On the line above the selected text, insert the following tag:

   ```
   -AMAZONPOLLY-ONLYAUDIO-START-
   ```
3. On the line following the text, insert the following tag:

   ```
   -AMAZONPOLLY-ONLYAUDIO-END-
   ```

You can use the same procedure to show text in the browser without including it in the audio file by using the tags `-AMAZONPOLLY-ONLYWORDS-START-` and `-AMAZONPOLLY-ONLYWORDS-END-` in the same manner.

For example:

```
Initial text of your blog displayed in the browser and heard in the audio file.]
-AMAZONPOLLY-ONLYAUDIO-START-
[This part will not be displayed in the browser but will be heard in the audio file.]
-AMAZONPOLLY-ONLYAUDIO-END-
[Subsequent text of your blog displayed in the browser and heard in the audio file.]
```

and

```
[Initial text of your blog displayed in the browser and heard in the audio file.]
-AMAZONPOLLY-ONLYWORD-START-
This part will be displayed in the browser but will not be heard in the audio file.]
-AMAZONPOLLY-ONLYWORD-END-
Subsequent text of your blog displayed in the browser and heard in the audio file.
```

Adding Translated Text to Your Post

The **AWS for WordPress** plugin uses Amazon Translate to create translated versions of your post in one or more other languages. In addition to English, four languages are available for this service: Spanish, French, German, and Portuguese. The languages to be used and the voices for those languages can be configured on the **Settings** page under Amazon Translate configuration.

To translate your WordPress post into other languages

1. On the **Add New Post** page, create and publish your new WordPress post.
2. On the same page, ensure that the **Enable Amazon Polly option** is chosen.
3. To see the approximate cost of creating audio files in the original languages plus any additionally selected languages, choose **How much will this cost to convert?** Choose **OK** to return to the **Add New Post** page.
4. Choose **Translate**

To set the languages into which you translate your post

1. On the **Settings** page, under **Amazon Translate configuration**, choose the language of your post from the **Source language** drop-down list.
2. Under **Target languages**, choose the languages into which you want to translate your post.
3. From the **Voice** dropdown list, choose the voice you want to use for each language selected.
4. Enter a label for the language choice.
5. Choose **Save Changes**.

Podcasting with Amazon Pollycast

With Amazon Pollycast feeds., your visitors can listen to your audio content using standard podcast applications. RSS 2.0-compliant *Pollycast feeds* provide the XML data needed to aggregate podcasts by popular mobile podcast applications, such as iTunes, and podcast directories.

When you install the **AWS for WordPress** plugin, choose the **ITunes explicit** option to automatically add Amazon Pollycast endpoints to all WordPress archive URLs. This lets you syndicate both site-wide or targeted podcasts. If you didn't choose the **ITunes explicit** option when you installed the plugin, follow these steps:

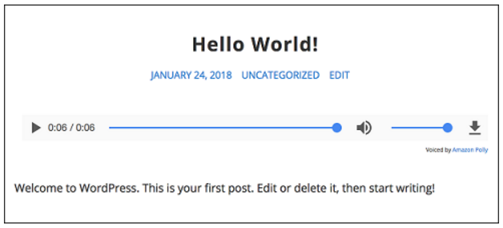
1. On the **WordPress Admin** page, choose **Settings**.
2. On the **Settings** page, choose **ITunes explicit**.

You can add Amazon Pollycast endpoints manually by adding `/amazon-pollycast/` to the URL for a page in a podcasting application. For example:

```
example.com/amazon-pollycast/
example.com/category/news/amazon-pollycast/
example.com/author/john/amazon-pollcast/
```

Positioning the Player

When you install the **AWS for WordPress** plugin plugin, it displays an audio player at the top of your WordPress website by default unless you choose to display it under your site's text or to not display it.



Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

You can reposition the player, remove it, or add it (if you chose not to display it) at any time.

To reposition the player, remove it, or add it to your WordPress website

1. On the **WordPress Admin** page, choose **Settings**.
2. On the **Amazon Polly Settings** page, for **Player position**, choose the appropriate option.

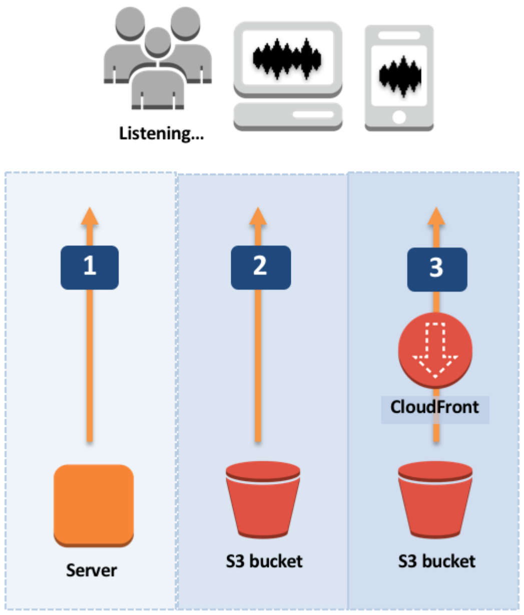
For more information about setting configuration options, see Configuring the Plugin (p. 180).

Storing Audio Files

When you publish content on your site, it's sent to Amazon Polly for synthesis. By default, Amazon Polly stores new audio files on your web server. You can also store the files on Amazon Simple Storage Service (Amazon S3) or on Amazon CloudFront, which is a global content delivery network (CDN).

Users have the same listening experience regardless of where you store your audio files. Only the broadcast location changes::

1. For audio files stored on the WordPress server, files are broadcast directly from the server.
2. For files stored in an S3 bucket, files are broadcast from the bucket.
3. If you use CloudFront, the files are stored on Amazon S3 and are broadcast with CloudFront.



You can choose where to store your files when you install the Amazon Polly plugin.

Limits in Amazon Polly

The following are limits to be aware of when using Amazon Polly.

Supported Regions

For a list of AWS Regions where Amazon Polly is available, see Amazon Polly Endpoints and Quotas in the *Amazon Web Services General Reference*.

Neural voices are currently supported in the following regions:

- US East (N. Virginia): us-east-1
- US West (Oregon): us-west-2
- EU (Ireland): eu-west-1
- Asia Pacific (Sydney): ap-southeast-2

Endpoints and protocols for these regions are identical to those used for standard voices.

Throttling

- Throttle rate per account: 100 transactions (requests or operations) per second (tps) with a burst limit of 120 tps.

  Concurrent connections per account: 90
- Throttle rate per operation:

Operation	Limit
Lexicon	

`DeleteLexicon`
`PutLexicon`
`GetLexicon`
`ListLexicons`
	Any 2 transactions per second (tps) from these operations combined.
	Maximum allowed burst of 4 tps.

Speech	
`DescribeVoices`	80 tps with a burst limit of 100 tps
`SynthesizeSpeech`	Standard voice: 80 tps with a burst limit of 100 tps Neural voice: 8 tps with a burst limit of 10 tps
`StartSpeechSynthesisTask`	Standard voice: 10 tps with a burst limit of 12 tps Neural voice: 1 tps

Operation	Limit
`GetSynthesizeSpeechTask` and `ListSynthesizeSpeechTask`	Maximum allowed 10 tps combined

Pronunciation Lexicons

- You can store up to 100 lexicons per account.
- Lexicon names can be an alphanumeric string up to 20 characters long.
- Each lexicon can be up to 4,000 characters in size. (Note that the size of the lexicon affects the latency of the SynthesizeSpeech operation.)
- You can specify up to 100 characters for each <phoneme> or <alias> replacement in a lexicon.

For information about using lexicons, see Managing Lexicons (p. 128).

SynthesizeSpeech API Operation

Note the following limits related to using the `SynthesizeSpeech` API operation:

- The size of the input text can be up to 3000 billed characters (6000 total characters). SSML tags are not counted as billed characters.
- You can specify up to five lexicons to apply to the input text.
- The output audio stream (synthesis) is limited to 10 minutes. After this is reached, any remaining speech is cut off.

For more information, see SynthesizeSpeech (p. 231).

Note
Some limitations of the `SynthesizeSpeech` API operation can be bypassed using the `StartSythensizeSpeechTask` API operation. For more information, see Creating Long Audio Files (p. 139).

SpeechSynthesisTask API Operations

Note the following limit relating to using the `StartSpeechSynthesisTask`, `GetSpeechSynthesisTask`, and `ListSpeechSynthesisTasks` API operations:

- The size of the input text can be up to 100,000 billed characters (200,000 total characters). SSML tags are not counted as billed characters.
- You can specify up to five lexicons to apply to the input text.

Speech Synthesis Markup Language (SSML)

Note the following limits related to using SSML:

- The `<audio>`, `<lexicon>`, `<lookup>`, and `<voice>` tags are not supported.

- `<break>` elements can specify a maximum duration of 10 seconds each.
- The `<prosody>` tag doesn't support values for the rate attribute lower than -80%.

For more information, see Generating Speech from SSML Documents (p. 100).

Logging Amazon Polly API Calls with AWS CloudTrail

Amazon Polly is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Polly. CloudTrail captures all API calls for Amazon Polly as events. The calls captured include calls from the Amazon Polly console and code calls to the Amazon Polly API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Polly. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Polly, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

Amazon Polly Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in Amazon Polly, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon Polly, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

Amazon Polly supports logging the following actions as events in CloudTrail log files:

- DeleteLexicon (p. 210)
- DescribeVoices (p. 212)
- GetLexicon (p. 215)
- GetSpeechSynthesisTask (p. 217)
- ListLexicons (p. 219)
- ListSpeechSynthesisTasks (p. 221)
- PutLexicon (p. 223)
- StartSpeechSynthesisTask (p. 225)
- SynthesizeSpeech (p. 231)

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Example: Amazon Polly Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `SynthesizeSpeech`.

```
{
    "Records": [
        {
            "awsRegion": "us-east-2",
            "eventID": "19bd70f7-5e60-4cdc-9825-936c552278ae",
            "eventName": "SynthesizeSpeech",
            "eventSource": "polly.amazonaws.com",
            "eventTime": "2016-11-02T03:49:39Z",
            "eventType": "AwsApiCall",
            "eventVersion": "1.05",
            "recipientAccountId": "123456789012",
            "requestID": "414288c2-a1af-11e6-b17f-d7cfc06cb461",
            "requestParameters": {
                "lexiconNames": [
                    "SampleLexicon"
                ],
                "outputFormat": "mp3",
                "sampleRate": "22050",
                "text": "**********",
                "textType": "text",
                "voiceId": "Kendra"
            },
            "responseElements": {
                "contentType": "audio/mpeg",
                "requestCharacters": 25
            },
            "sourceIPAddress": "1.2.3.4",
            "userAgent": "Amazon CLI/Polly 1.10 API 2016-06-10",
            "userIdentity": {
                "accessKeyId": "EXAMPLE_KEY_ID",
                "accountId": "123456789012",
                "arn": "arn:aws:iam::123456789012:user/Alice",
                "principalId": "EX_PRINCIPAL_ID",
                "type": "IAMUser",
                "userName": "Alice"
            }
        }

    ]
}
```

Integrating CloudWatch with Amazon Polly

When you interact with Amazon Polly, it sends the following metrics and dimensions to CloudWatch every minute. You can use the following procedures to view the metrics for Amazon Polly.

You can monitor Amazon Polly using CloudWatch, which collects and processes raw data from Amazon Polly into readable, near real-time metrics. These statistics are recorded for a period of two weeks, so that you can access `historical information` and gain a better perspective on how your web application or service is performing. By default, Amazon Polly metric data is sent to CloudWatch in 1 minute intervals. For more information, see What Is Amazon CloudWatch in the Amazon *CloudWatch User Guide*.

Getting CloudWatch Metrics (Console)

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. In the **CloudWatch Metrics by Category** pane, under the metrics category for Amazon Polly, select a metrics category, and then in the upper pane, scroll down to view the full list of metrics.

Getting CloudWatch Metrics (CLI)

The following code display available metrics for Amazon Polly.

```
aws cloudwatch list-metrics --namespace "AWS/Polly"
```

The preceding command returns a list of Amazon Polly metrics similar to the following. The `MetricName` element identifies what the metric is.

```
{
    "Metrics": [
        {
            "Namespace": "AWS/Polly",
            "Dimensions": [
                {
                    "Name": "Operation",
                    "Value": "SynthesizeSpeech"
                }
            ],
            "MetricName": "ResponseLatency"
        },
        {
            "Namespace": "AWS/Polly",
            "Dimensions": [
                {
                    "Name": "Operation",
                    "Value": "SynthesizeSpeech"
                }
            ],
            "MetricName": "RequestCharacters"
        }
```

For more information, see GetMetricStatistics in the *Amazon CloudWatch API Reference*.

Amazon Polly Metrics

Amazon Polly produces the following metrics for each request. These metrics are aggregated and in one minute intervals sent to CloudWatch where they are available.

Metric	Description
`RequestCharacters`	The number of characters in the request. This is billable characters only and does not include SSML tags. Valid Dimension: Operation Valid Statistics: Minimum, Maximum, Average, SampleCount, Sum Unit: Count
`ResponseLatency`	The latency between when the request was made and the start of the streaming response. Valid Dimensions: Operation Valid Statistics: Minimum, Maximum, Average, SampleCount Unit: milliseconds
`2XXCount`	HTTP 200 level code returned upon a successful response. Valid Dimensions: Operation Valid Statistics: Average, SampleCount, Sum Unit: Count
`4XXCount`	HTTP 400 level error code returned upon an error. For each successful response, a zero (0) is emitted. Valid Dimensions: Operation Valid Statistics: Average, SampleCount, Sum Unit: Count
`5XXCount`	HTTP 500 level error code returned upon an error. For each successful response, a zero (0) is emitted. Valid Dimensions: Operation Valid Statistics: Average, SampleCount, Sum Unit: Count

Dimensions for Amazon Polly Metrics

Amazon Polly metrics use the AWS/Polly namespace and provide metrics for the following dimension:

Dimension	**Description**
`Operation`	Metrics are grouped by the API method they refer to. Possible values are `SynthesizeSpeech`, `PutLexicon`, `DescribeVoices`, etc.

Security in Amazon Polly

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Polly, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon Polly. The following topics show you how to configure Amazon Polly to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon Polly resources.

Topics

Data Protection in Amazon Polly

Amazon Polly conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Amazon Polly or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon Polly or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

Encryption at Rest

Output of your Amazon Polly voice synthesis can be saved on your own system. You can also call Amazon Polly, and then encrypt the file with any encryption key of your choice and store it in Amazon Simple Storage Service (Amazon S3) or another secure storage. The Amazon Polly the section called "SynthesizeSpeech" (p. 231) operation is stateless and is not associated with a customer identity. You can't retrieve it from Amazon Polly later.

Encryption in Transit

All text submissions are protected by Secure Sockets Layer (SSL) while in transit. Amazon Polly does not reptain the content of text submissions.

Internetwork Traffic Privacy

Access to Amazon Polly is via AWS Console, CLI, or SDKs. Communications utilize Transport Layer Security (TLS) session encryption for confidentiality and digital signatures for authentication and integrity.

Identity and Access Management in Amazon Polly

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Polly resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in Amazon Polly.

Service user – If you use the Amazon Polly service to do your job, your administrator provides you with the credentials and permissions that you need. As you use more Amazon Polly features to do your work, you might need additional permissions. Understanding how access is managed helps you request the right permissions from your administrator.

Service administrator – If you're in charge of Amazon Polly resources at your company, you probably have full access to Amazon Polly. It's your job to determine which Amazon Polly features and resources your employees should access. Then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon Polly, see How Amazon Polly Works with IAM (p. 200).

IAM administrator – If you're an IAM administrator, you might want to learn details about writing policies to manage access to Amazon Polly. To view example Amazon Polly identity-based policies that you can use in IAM, see Amazon Polly Identity-Based Policy Examples (p. 202).

Authenticating With Identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see The IAM Console and Sign-in Page in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don’t use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.

AWS Account Root User

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM Users and Groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing Access Keys for IAM Users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to Create an IAM User (Instead of a Role) in the *IAM User Guide*.

IAM Roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM Roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.
- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in the *IAM User Guide*.

To learn whether to use IAM roles, see When to Create an IAM Role (Instead of a User) in the *IAM User Guide*.

Managing Access Using Policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions.

AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON Policies in the *IAM User Guide*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM Policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing Between Managed Policies and Inline Policies in the *IAM User Guide*.

Resource-Based Policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. Amazon Polly doesn't support resource-based policies.

Other Policy Types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions Boundaries for IAM Entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs Work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies.

Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session Policies in the *IAM User Guide*.

Multiple Policy Types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy Evaluation Logic in the *IAM User Guide*.

How Amazon Polly Works with IAM

Before you use IAM to manage access to Amazon Polly, you should understand what IAM features are available to use with Amazon Polly. To get a high-level view of how Amazon Polly and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

Topics

Amazon Polly Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Amazon Polly supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

Actions

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon Polly use the following prefix before the action: `polly:`. For example, to grant someone permission to delete a Amazon Polly lexicon with the Amazon Polly `DeleteLexicon` API operation, you include the `polly:DeletedLexicon` action in their policy. Policy statements must include either an `Action` or `NotAction` element. Amazon Polly defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
      "polly:action1",
      "polly:action2"]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Get`, include the following action:

```
"Action": "polly:Get*"
```

To see a list of Amazon Polly actions, see Actions Defined by Amazon Polly in the *IAM User Guide*.

Resources

Amazon Polly does not support specifying resource ARNs in a policy.

Condition Keys

Amazon Polly does not provide any service-specific condition keys, but it does support using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

Examples

To view examples of Amazon Polly identity-based policies, see Amazon Polly Identity-Based Policy Examples (p. 202).

Amazon Polly IAM Roles

You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in account A can create a role to grant cross-account permissions to another AWS account (for example, account B) or an AWS service as follows:

1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in account A.
2. Account A administrator attaches a trust policy to the role identifying account B as the principal who can assume the role.
3. Account B administrator can then delegate permissions to assume the role to any users in account B. Doing this allows users in account B to create or access resources in account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

For more information about using IAM to delegate permissions, see Access Management in the *IAM User Guide*.

The following is an example policy that grants permissions to put and get lexicons as well as to list those lexicons currently available.

Amazon Polly supports Identity-based policies for actions at the resource-level. In some cases, the resource can be limited by an ARN. This is true for the `SynthesizeSpeech`, `StartSpeechSynthesisTask`, `PutLexicon`, `GetLexicon`, and `DeleteLexicon` operations. In these cases, the `Resource` value is indicated by the ARN. For example: `arn:aws:polly:`*`us-east-2`*`:`*`account-id`*`:lexicon/*` as the `Resource` value specifies permissions on all owned lexicons within the `us-east-2` Region.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "AllowPut-Get-ListActions",
      "Effect": "Allow",
      "Action": [
         "polly:PutLexicon",
         "polly:GetLexicon",
         "polly:ListLexicons"],
      "Resource": "arn:aws:polly:us-east-2:account-id:lexicon/*"
      }
   ]
}
```

However, not all operations use ARNs. This is the case with the `DescribeVoices`, `ListLexicons`, `GetSpeechSynthesisTasks`, and `ListSpeechSynthesisTasks` operations.

For more information about users, groups, roles, and permissions, see Identities (Users, Groups, and Roles) in the *IAM User Guide*.

Amazon Polly Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify Amazon Polly resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

Topics

Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Amazon Polly resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** – To start using Amazon Polly quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get Started Using Permissions With AWS Managed Policies in the *IAM User Guide*.
- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

Using the Amazon Polly Console

For a user to work with the Amazon Polly console, that user must have a minimum set of permissions to describe the Amazon Polly resources in their AWS account.

If you create an IAM policy that is more restrictive than the minimum required permissions, the console doesn't function as intended for users with that IAM policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the Amazon Polly API.

To use the Amazon Polly console, grant permissions to all the Amazon Polly APIs. There are no additional permissions needed. To get full console functionality you can use following policy:.

```
}
  "Version": "2012-10-17",
     "Statement": [{
        "Sid": "Console-AllowAllPollyActions",
        "Effect": "Allow",
        "Action": [
           "polly:*"],
        "Resource": "*"
      }
   ]
}
```

Allow Users to View Their Own Permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "ViewOwnUserInfo",
               "Effect": "Allow",
               "Action": [
                   "iam:GetUserPolicy",
                   "iam:ListGroupsForUser",
                   "iam:ListAttachedUserPolicies",
                   "iam:ListUserPolicies",
                   "iam:GetUser"
               ],
               "Resource": [
                   "arn:aws:iam::*:user/${aws:username}"
               ]
           },
           {
               "Sid": "NavigateInConsole",
               "Effect": "Allow",
               "Action": [
                   "iam:GetGroupPolicy",
                   "iam:GetPolicyVersion",
                   "iam:GetPolicy",
                   "iam:ListAttachedGroupPolicies",
                   "iam:ListGroupPolicies",
                   "iam:ListPolicyVersions",
                   "iam:ListPolicies",
                   "iam:ListUsers"
               ],
               "Resource": "*"
           }
       ]
   }
```

AWS Managed (Predefined) Policies for Amazon Polly

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so that you can avoid having to investigate what permissions are needed. For more information, see AWS Managed Policies in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon Polly:

- **AmazonPollyReadOnlyAccess** – Grants read-only access to resources, allows listing lexicons, fetching lexicons, listing available voices and synthesizing speech (including, applying lexicons to the synthesized speech).
- **AmazonPollyFullAccess** – Grants full access to resources and all the supported operations.

Note
You can review these permissions policies by signing in to the IAM console and searching for specific policies there.

You can also create your own custom IAM policies to allow permissions for Amazon Polly actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various Amazon Polly actions. These policies work when you are using AWS SDKs or the AWS CLI. When you are using the console, grant permissions to all the Amazon Polly APIs.

Note
All examples use the us-east-2 Region and contain fictitious account IDs.

Examples

Example 1: Allow All Amazon Polly Actions

After you sign up (see Step 1.1: Sign up for AWS (p. 4)) create an administrator user to manage your account, including creating users and managing their permissions.

You might create a user who has permissions for all Amazon Polly actions. Think of this user as a service-specific administrator for working with Amazon Polly. You can attach the following permissions policy to this user.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "AllowAllPollyActions",
      "Effect": "Allow",
      "Action": [
         "polly:*"],
      "Resource": "*"
      }
   ]
}
```

Example 2: Allow All Amazon Polly Actions Except DeleteLexicon

The following permissions policy grants the user permissions to perform all actions except `DeleteLexicon`, with the permissions for delete explicitly denied in all Regions.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "AllowAllActions-DenyDelete",
      "Effect": "Allow",
      "Action": [
         "polly:DescribeVoices",
         "polly:GetLexicon",
         "polly:PutLexicon",
         "polly:SynthesizeSpeech",
         "polly:ListLexicons"],
      "Resource": "*"
      }
      {
      "Sid": "DenyDeleteLexicon",
      "Effect": "Deny",
      "Action": [
         "polly:DeleteLexicon"],
      "Resource": "*"
      }
   ]
}
```

Example 3: Allow DeleteLexicon

The following permissions policy grants the user permissions to delete any lexicon that you own regardless of the project or Region in which it is located.

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Sid": "AllowDeleteLexicon",
      "Effect": "Allow",
      "Action": [
         "polly:DeleteLexicon"],
      "Resource": "*"
      }
   ]
}
```

Example 4: Allow Delete Lexicon in a Specified Region

The following permissions policy grants the user permissions to delete any lexicon in any project that you own that is located in a single Region (in this case, us-east-2).

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Sid": "AllowDeleteSpecifiedRegion",
      "Effect": "Allow",
      "Action": [
         "polly:DeleteLexicon"],
      "Resource": "arn:aws:polly:us-east-2:123456789012:lexicon/*"
      }
   ]
}
```

Example 5: Allow DeleteLexicon for Specified Lexicon

The following permissions policy grants the user permissions to delete a specific lexicon that you own (in this case, myLexicon) in a specific Region (in this case, us-east-2).

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Sid": "AllowDeleteForSpecifiedLexicon",
      "Effect": "Allow",
      "Action": [
         "polly:DeleteLexicon"],
      "Resource": "arn:aws:polly:us-east-2:123456789012:lexicon/myLexicon"
      }
   ]
}
```

Amazon Polly API Permissions: Actions, Permissions, and Resources Reference

When you are setting up a permissions policy that you can attach to an IAM identity (identity-based policies), you can use the following list as a reference. The list includes each Amazon Polly API operation, the corresponding actions for which you can grant permissions to perform the action, and the AWS resource for which you can grant the permissions. You specify the actions in the policy's `Action` field, and you specify the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your Amazon Polly policies to express conditions. For a complete list of AWS-wide keys, see Available Keys in the *IAM User Guide*.

> **Note**
> To specify an action, use the `polly` prefix followed by the API operation name (for example, `polly:GetLexicon`).

Amazon Polly supports Identity-based policies for actions at the resource-level. Therefore, the `Resource` value is indicated by the ARN. For example: `arn:aws:polly:`*`us-east-2`*`:`*`account-id`*`:lexicon/*` as the `Resource` value specifies permissions on all owned lexicons within the `us-east-2` Region.

Because Amazon Polly doesn't support permissions for actions at the resource-level, most policies specify a wildcard character (*) as the `Resource` value. However, if it is necessary to limit permissions to a specific Region this wildcard character is replaced with the appropriate ARN: `arn:aws:polly:`*`region`*`:`*`account-id`*`:lexicon/*`.

Amazon Polly API and Required Permissions for Actions

API Operation: DeleteLexicon (p. 210)

Required Permissions (API Action): `polly:DeleteLexicon`

Resources: `arn:aws:polly:`*`region`*`:`*`account-id`*`:lexicon/`*`LexiconName`*

API Operation: DescribeVoices (p. 212)

Required Permissions (API Action): `polly:DescribeVoices`

Resources: `arn:aws:polly:`*`region`*`:`*`account-id`*`:lexicon/`*`voice-name`*

API Operation: GetLexicon (p. 215)

Required Permissions (API Action): `polly:GetLexicon`

Resources: `arn:aws:polly:`*`region`*`:`*`account-id`*`:lexicon/`*`voice-name`*

API Operation: ListLexicons (p. 219)

Required Permissions (API Action): `polly:ListLexicons`

Resources: `arn:aws:polly:region:account-id:lexicon/*`

API Operation: PutLexicon (p. 223)

Required Permissions (API Action): `polly:ListLexicons`

Resources: `*`

API Operation: SynthesizeSpeech (p. 231)

Required Permissions (API Action): `polly:SynthesizeSpeech`

Resources: `*`

Logging and Monitoring in Amazon Polly

Monitoring is an important part of maintaining the reliability, availability, and performance of your Amazon Polly applications. To monitor Amazon Polly API calls, you can use AWS CloudTrail. To monitor the status of your jobs, use Amazon CloudWatch Logs.

- **Amazon CloudWatch Alarms** – Using CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon Simple Notification Service topic or AWS Auto Scaling policy. CloudWatch alarms do not invoke actions when a metric is in a particular state. Rather the state must have changed and been maintained for a specified number of periods. For more information, see Integrating CloudWatch with Amazon Polly (p. 192).
- **AWS CloudTrail Logs** – CloudTrail provides a record of actions taken by a user, role, or an AWS service in Amazon Polly. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Polly. You can also determine the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging Amazon Polly API Calls with AWS CloudTrail (p. 189).

Compliance Validation for Amazon Polly

Third-party auditors assess the security and compliance of Amazon Polly as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using Amazon Polly is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.

- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon Polly

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure Security in Amazon Polly

As a managed service, Amazon Polly is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Amazon Polly through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Security Best Practices for Amazon Polly

Your trust, privacy, and the security of your content are our highest priorities. We implement responsible and sophisticated technical and physical controls designed to prevent unauthorized access to, or disclosure of, your content and ensure that our use complies with our commitments to you. For more information, see AWS Data Privacy FAQ.

Amazon Polly does not retain the the content of text submissions.

For a broad view of AWS security, including compliance, penetration testing, bulletins, and resources, visit the AWS Cloud Security website.

Amazon Polly API Reference

This section contains the Amazon Polly API reference.

Note
Authenticated API calls must be signed using the Signature Version 4 Signing Process. For more information, see Signing AWS API Requests in the *Amazon Web Services General Reference*.

Topics

Actions

The following actions are supported:

- DeleteLexicon (p. 210)
- DescribeVoices (p. 212)
- GetLexicon (p. 215)
- GetSpeechSynthesisTask (p. 217)
- ListLexicons (p. 219)
- ListSpeechSynthesisTasks (p. 221)
- PutLexicon (p. 223)
- StartSpeechSynthesisTask (p. 225)
- SynthesizeSpeech (p. 231)

DeleteLexicon

Deletes the specified pronunciation lexicon stored in an AWS Region. A lexicon which has been deleted is not available for speech synthesis, nor is it possible to retrieve it using either the `GetLexicon` or `ListLexicon` APIs.

For more information, see Managing Lexicons.

Request Syntax

```
DELETE /v1/lexicons/LexiconName HTTP/1.1
```

URI Request Parameters

The request requires the following URI parameters.

LexiconName (p. 210)

The name of the lexicon to delete. Must be an existing lexicon in the region.

Pattern: `[0-9A-Za-z]{1,20}`

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

LexiconNotFoundException

Amazon Polly can't find the specified lexicon. This could be caused by a lexicon that is missing, its name is misspelled or specifying a lexicon that is in a different region.

Verify that the lexicon exists, is in the region (see ListLexicons (p. 219)) and that you spelled its name is spelled correctly. Then try again.

HTTP Status Code: 404

ServiceFailureException

An unknown condition has caused a service failure.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeVoices

Returns the list of voices that are available for use when requesting speech synthesis. Each voice speaks a specified language, is either male or female, and is identified by an ID, which is the ASCII version of the voice name.

When synthesizing speech (`SynthesizeSpeech`), you provide the voice ID for the voice you want from the list of voices returned by `DescribeVoices`.

For example, you want your news reader application to read news in a specific language, but giving a user the option to choose the voice. Using the `DescribeVoices` operation you can provide the user with a list of available voices to select from.

You can optionally specify a language code to filter the available voices. For example, if you specify `en-US`, the operation returns a list of all available US English voices.

This operation requires permissions to perform the `polly:DescribeVoices` action.

Request Syntax

```
GET /v1/voices?
Engine=Engine&IncludeAdditionalLanguageCodes=IncludeAdditionalLanguageCodes&LanguageCode=LanguageCode&N
 HTTP/1.1
```

URI Request Parameters

The request requires the following URI parameters.

Engine (p. 212)

Specifies the engine (`standard` or `neural`) used by Amazon Polly when processing input text for speech synthesis.

Valid Values: `standard | neural`

IncludeAdditionalLanguageCodes (p. 212)

Boolean value indicating whether to return any bilingual voices that use the specified language as an additional language. For instance, if you request all languages that use US English (es-US), and there is an Italian voice that speaks both Italian (it-IT) and US English, that voice will be included if you specify `yes` but not if you specify `no`.

LanguageCode (p. 212)

The language identification tag (ISO 639 code for the language name-ISO 3166 country code) for filtering the list of voices returned. If you don't specify this optional parameter, all available voices are returned.

Valid Values: `arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR`

NextToken (p. 212)

An opaque pagination token returned from the previous `DescribeVoices` operation. If present, this indicates where to continue the listing.

Length Constraints: Minimum length of 0. Maximum length of 4096.

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Voices": [
      {
         "AdditionalLanguageCodes": [ "string" ],
         "Gender": "string",
         "Id": "string",
         "LanguageCode": "string",
         "LanguageName": "string",
         "Name": "string",
         "SupportedEngines": [ "string" ]
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken (p. 213)

The pagination token to use in the next request to continue the listing of voices. `NextToken` is returned only if the response is truncated.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Voices (p. 213)

A list of voices with their properties.

Type: Array of Voice (p. 243) objects

Errors

InvalidNextTokenException

The NextToken is invalid. Verify that it's spelled correctly, and then try again.

HTTP Status Code: 400

ServiceFailureException

An unknown condition has caused a service failure.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetLexicon

Returns the content of the specified pronunciation lexicon stored in an AWS Region. For more information, see Managing Lexicons.

Request Syntax

```
GET /v1/lexicons/LexiconName HTTP/1.1
```

URI Request Parameters

The request requires the following URI parameters.

LexiconName (p. 215)

Name of the lexicon.

Pattern: `[0-9A-Za-z]{1,20}`

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Lexicon": {
      "Content": "string",
      "Name": "string"
   },
   "LexiconAttributes": {
      "Alphabet": "string",
      "LanguageCode": "string",
      "LastModified": number,
      "LexemesCount": number,
      "LexiconArn": "string",
      "Size": number
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Lexicon (p. 215)

Lexicon object that provides name and the string content of the lexicon.

Type: Lexicon (p. 236) object

LexiconAttributes (p. 215)

Metadata of the lexicon, including phonetic alphabetic used, language code, lexicon ARN, number of lexemes defined in the lexicon, and size of lexicon in bytes.

Type: LexiconAttributes (p. 237) object

Errors

LexiconNotFoundException

Amazon Polly can't find the specified lexicon. This could be caused by a lexicon that is missing, its name is misspelled or specifying a lexicon that is in a different region.

Verify that the lexicon exists, is in the region (see ListLexicons (p. 219)) and that you spelled its name is spelled correctly. Then try again.

HTTP Status Code: 404

ServiceFailureException

An unknown condition has caused a service failure.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

GetSpeechSynthesisTask

Retrieves a specific SpeechSynthesisTask object based on its TaskID. This object contains information about the given speech synthesis task, including the status of the task, and a link to the S3 bucket containing the output of the task.

Request Syntax

```
GET /v1/synthesisTasks/TaskId HTTP/1.1
```

URI Request Parameters

The request requires the following URI parameters.

TaskId (p. 217)

The Amazon Polly generated identifier for a speech synthesis task.

Pattern: `^[a-zA-Z0-9_-]{1,100}$`

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "SynthesisTask": {
      "CreationTime": number,
      "Engine": "string",
      "LanguageCode": "string",
      "LexiconNames": [ "string" ],
      "OutputFormat": "string",
      "OutputUri": "string",
      "RequestCharacters": number,
      "SampleRate": "string",
      "SnsTopicArn": "string",
      "SpeechMarkTypes": [ "string" ],
      "TaskId": "string",
      "TaskStatus": "string",
      "TaskStatusReason": "string",
      "TextType": "string",
      "VoiceId": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

SynthesisTask (p. 217)

SynthesisTask object that provides information from the requested task, including output format, creation time, task status, and so on.

Type: SynthesisTask (p. 240) object

Errors

InvalidTaskIdException

The provided Task ID is not valid. Please provide a valid Task ID and try again.

HTTP Status Code: 400

ServiceFailureException

An unknown condition has caused a service failure.

HTTP Status Code: 500

SynthesisTaskNotFoundException

The Speech Synthesis task with requested Task ID cannot be found.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListLexicons

Returns a list of pronunciation lexicons stored in an AWS Region. For more information, see Managing Lexicons.

Request Syntax

```
GET /v1/lexicons?NextToken=NextToken HTTP/1.1
```

URI Request Parameters

The request requires the following URI parameters.

NextToken (p. 219)

An opaque pagination token returned from previous `ListLexicons` operation. If present, indicates where to continue the list of lexicons.

Length Constraints: Minimum length of 0. Maximum length of 4096.

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "Lexicons": [
      {
         "Attributes": {
            "Alphabet": "string",
            "LanguageCode": "string",
            "LastModified": number,
            "LexemesCount": number,
            "LexiconArn": "string",
            "Size": number
         },
         "Name": "string"
      }
   ],
   "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Lexicons (p. 219)

A list of lexicon names and attributes.

Type: Array of LexiconDescription (p. 239) objects

NextToken (p. 219)

The pagination token to use in the next request to continue the listing of lexicons. `NextToken` is returned only if the response is truncated.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

Errors

InvalidNextTokenException

The NextToken is invalid. Verify that it's spelled correctly, and then try again.

HTTP Status Code: 400

ServiceFailureException

An unknown condition has caused a service failure.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListSpeechSynthesisTasks

Returns a list of SpeechSynthesisTask objects ordered by their creation date. This operation can filter the tasks by their status, for example, allowing users to list only tasks that are completed.

Request Syntax

```
GET /v1/synthesisTasks?MaxResults=MaxResults&NextToken=NextToken&Status=Status HTTP/1.1
```

URI Request Parameters

The request requires the following URI parameters.

MaxResults (p. 221)

Maximum number of speech synthesis tasks returned in a List operation.

Valid Range: Minimum value of 1. Maximum value of 100.

NextToken (p. 221)

The pagination token to use in the next request to continue the listing of speech synthesis tasks.

Length Constraints: Minimum length of 0. Maximum length of 4096.

Status (p. 221)

Status of the speech synthesis tasks returned in a List operation

Valid Values: `scheduled | inProgress | completed | failed`

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "SynthesisTasks": [
      {
         "CreationTime": number,
         "Engine": "string",
         "LanguageCode": "string",
         "LexiconNames": [ "string" ],
         "OutputFormat": "string",
         "OutputUri": "string",
         "RequestCharacters": number,
         "SampleRate": "string",
         "SnsTopicArn": "string",
         "SpeechMarkTypes": [ "string" ],
         "TaskId": "string",
         "TaskStatus": "string",
         "TaskStatusReason": "string",
         "TextType": "string",
         "VoiceId": "string"
```

```
        }
    ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken (p. 221)

An opaque pagination token returned from the previous List operation in this request. If present, this indicates where to continue the listing.

Type: String

Length Constraints: Minimum length of 0. Maximum length of 4096.

SynthesisTasks (p. 221)

List of SynthesisTask objects that provides information from the specified task in the list request, including output format, creation time, task status, and so on.

Type: Array of SynthesisTask (p. 240) objects

Errors

InvalidNextTokenException

The NextToken is invalid. Verify that it's spelled correctly, and then try again.

HTTP Status Code: 400

ServiceFailureException

An unknown condition has caused a service failure.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutLexicon

Stores a pronunciation lexicon in an AWS Region. If a lexicon with the same name already exists in the region, it is overwritten by the new lexicon. Lexicon operations have eventual consistency, therefore, it might take some time before the lexicon is available to the SynthesizeSpeech operation.

For more information, see Managing Lexicons.

Request Syntax

```
PUT /v1/lexicons/LexiconName HTTP/1.1
Content-type: application/json

{
   "Content": "string"
}
```

URI Request Parameters

The request requires the following URI parameters.

LexiconName (p. 223)

Name of the lexicon. The name must follow the regular express format [0-9A-Za-z]{1,20}. That is, the name is a case-sensitive alphanumeric string up to 20 characters long.

Pattern: `[0-9A-Za-z]{1,20}`

Request Body

The request accepts the following data in JSON format.

Content (p. 223)

Content of the PLS lexicon as string data.

Type: String

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

InvalidLexiconException

Amazon Polly can't find the specified lexicon. Verify that the lexicon's name is spelled correctly, and then try again.

HTTP Status Code: 400

LexiconSizeExceededException

The maximum size of the specified lexicon would be exceeded by this operation.

HTTP Status Code: 400

MaxLexemeLengthExceededException

The maximum size of the lexeme would be exceeded by this operation.

HTTP Status Code: 400

MaxLexiconsNumberExceededException

The maximum number of lexicons would be exceeded by this operation.

HTTP Status Code: 400

ServiceFailureException

An unknown condition has caused a service failure.

HTTP Status Code: 500

UnsupportedPlsAlphabetException

The alphabet specified by the lexicon is not a supported alphabet. Valid values are `x-sampa` and `ipa`.

HTTP Status Code: 400

UnsupportedPlsLanguageException

The language specified in the lexicon is unsupported. For a list of supported languages, see Lexicon Attributes.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StartSpeechSynthesisTask

Allows the creation of an asynchronous synthesis task, by starting a new `SpeechSynthesisTask`. This operation requires all the standard information needed for speech synthesis, plus the name of an Amazon S3 bucket for the service to store the output of the synthesis task and two optional parameters (OutputS3KeyPrefix and SnsTopicArn). Once the synthesis task is created, this operation will return a SpeechSynthesisTask object, which will include an identifier of this task as well as the current status.

Request Syntax

```
POST /v1/synthesisTasks HTTP/1.1
Content-type: application/json

{
   "Engine": "string",
   "LanguageCode": "string",
   "LexiconNames": [ "string" ],
   "OutputFormat": "string",
   "OutputS3BucketName": "string",
   "OutputS3KeyPrefix": "string",
   "SampleRate": "string",
   "SnsTopicArn": "string",
   "SpeechMarkTypes": [ "string" ],
   "Text": "string",
   "TextType": "string",
   "VoiceId": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

Engine (p. 225)

Specifies the engine (`standard` or `neural`) for Amazon Polly to use when processing input text for speech synthesis. Using a voice that is not supported for the engine selected will result in an error.

Type: String

Valid Values: `standard | neural`

Required: No

LanguageCode (p. 225)

Optional language code for the Speech Synthesis request. This is only necessary if using a bilingual voice, such as Aditi, which can be used for either Indian English (en-IN) or Hindi (hi-IN).

If a bilingual voice is used and no language code is specified, Amazon Polly will use the default language of the bilingual voice. The default language for any voice is the one returned by the DescribeVoices operation for the `LanguageCode` parameter. For example, if no language code is specified, Aditi will use Indian English rather than Hindi.

Type: String

Valid Values: `arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR`

Required: No

LexiconNames (p. 225)

List of one or more pronunciation lexicon names you want the service to apply during synthesis. Lexicons are applied only if the language of the lexicon is the same as the language of the voice.

Type: Array of strings

Array Members: Maximum number of 5 items.

Pattern: `[0-9A-Za-z]{1,20}`

Required: No

OutputFormat (p. 225)

The format in which the returned output will be encoded. For audio stream, this will be mp3, ogg_vorbis, or pcm. For speech marks, this will be json.

Type: String

Valid Values: `json | mp3 | ogg_vorbis | pcm`

Required: Yes

OutputS3BucketName (p. 225)

Amazon S3 bucket name to which the output file will be saved.

Type: String

Pattern: `^[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]$`

Required: Yes

OutputS3KeyPrefix (p. 225)

The Amazon S3 key prefix for the output speech file.

Type: String

Pattern: `^[0-9a-zA-Z\/\!\-_\.\*\'\(\)]{0,800}$`

Required: No

SampleRate (p. 225)

The audio frequency specified in Hz.

The valid values for mp3 and ogg_vorbis are "8000", "16000", "22050", and "24000". The default value for standard voices is "22050". The default value for neural voices is "24000".

Valid values for pcm are "8000" and "16000" The default value is "16000".

Type: String

Required: No

SnsTopicArn (p. 225)

ARN for the SNS topic optionally used for providing status notification for a speech synthesis task.

Type: String

Pattern: `^arn:aws(-(cn|iso(-b)?|us-gov))?:sns:[a-z0-9_-]{1,50}:\d{12}:[a-zA-Z0-9_-]{1,256}$`

Required: No

SpeechMarkTypes (p. 225)

The type of speech marks returned for the input text.

Type: Array of strings

Array Members: Maximum number of 4 items.

Valid Values: `sentence | ssml | viseme | word`

Required: No

Text (p. 225)

The input text to synthesize. If you specify ssml as the TextType, follow the SSML format for the input text.

Type: String

Required: Yes

TextType (p. 225)

Specifies whether the input text is plain text or SSML. The default value is plain text.

Type: String

Valid Values: `ssml | text`

Required: No

VoiceId (p. 225)

Voice ID to use for the synthesis.

Type: String

Valid Values: `Aditi | Amy | Astrid | Bianca | Brian | Camila | Carla | Carmen | Celine | Chantal | Conchita | Cristiano | Dora | Emma | Enrique | Ewa | Filiz | Geraint | Giorgio | Gwyneth | Hans | Ines | Ivy | Jacek | Jan | Joanna | Joey | Justin | Karl | Kendra | Kimberly | Lea | Liv | Lotte | Lucia | Lupe | Mads | Maja | Marlene | Mathieu | Matthew | Maxim | Mia | Miguel | Mizuki | Naja | Nicole | Penelope | Raveena | Ricardo | Ruben | Russell | Salli | Seoyeon | Takumi | Tatyana | Vicki | Vitoria | Zeina | Zhiyu`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "SynthesisTask": {
      "CreationTime": number,
      "Engine": "string",
      "LanguageCode": "string",
      "LexiconNames": [ "string" ],
      "OutputFormat": "string",
      "OutputUri": "string",
      "RequestCharacters": number,
      "SampleRate": "string",
      "SnsTopicArn": "string",
      "SpeechMarkTypes": [ "string" ],
      "TaskId": "string",
      "TaskStatus": "string",
      "TaskStatusReason": "string",
      "TextType": "string",
      "VoiceId": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

SynthesisTask (p. 227)

SynthesisTask object that provides information and attributes about a newly submitted speech synthesis task.

Type: SynthesisTask (p. 240) object

Errors

EngineNotSupportedException

This engine is not compatible with the voice that you have designated. Choose a new voice that is compatible with the engine or change the engine and restart the operation.

HTTP Status Code: 400

InvalidS3BucketException

The provided Amazon S3 bucket name is invalid. Please check your input with S3 bucket naming requirements and try again.

HTTP Status Code: 400

InvalidS3KeyException

The provided Amazon S3 key prefix is invalid. Please provide a valid S3 object key name.

HTTP Status Code: 400

InvalidSampleRateException

The specified sample rate is not valid.

HTTP Status Code: 400

InvalidSnsTopicArnException

The provided SNS topic ARN is invalid. Please provide a valid SNS topic ARN and try again.

HTTP Status Code: 400

InvalidSsmlException

The SSML you provided is invalid. Verify the SSML syntax, spelling of tags and values, and then try again.

HTTP Status Code: 400

LanguageNotSupportedException

The language specified is not currently supported by Amazon Polly in this capacity.

HTTP Status Code: 400

LexiconNotFoundException

Amazon Polly can't find the specified lexicon. This could be caused by a lexicon that is missing, its name is misspelled or specifying a lexicon that is in a different region.

Verify that the lexicon exists, is in the region (see ListLexicons (p. 219)) and that you spelled its name is spelled correctly. Then try again.

HTTP Status Code: 404

MarksNotSupportedForFormatException

Speech marks are not supported for the `OutputFormat` selected. Speech marks are only available for content in `json` format.

HTTP Status Code: 400

ServiceFailureException

An unknown condition has caused a service failure.

HTTP Status Code: 500

SsmlMarksNotSupportedForTextTypeException

SSML speech marks are not supported for plain text-type input.

HTTP Status Code: 400

TextLengthExceededException

The value of the "Text" parameter is longer than the accepted limits. For the `SynthesizeSpeech` API, the limit for input text is a maximum of 6000 characters total, of which no more than 3000 can be billed characters. For the `StartSpeechSynthesisTask` API, the maximum is 200,000 characters, of which no more than 100,000 can be billed characters. SSML tags are not counted as billed characters.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript

- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

SynthesizeSpeech

Synthesizes UTF-8 input, plain text or SSML, to a stream of bytes. SSML input must be valid, well-formed SSML. Some alphabets might not be available with all the voices (for example, Cyrillic might not be read at all by English voices) unless phoneme mapping is used. For more information, see How it Works.

Request Syntax

```
POST /v1/speech HTTP/1.1
Content-type: application/json

{
   "Engine": "string",
   "LanguageCode": "string",
   "LexiconNames": [ "string" ],
   "OutputFormat": "string",
   "SampleRate": "string",
   "SpeechMarkTypes": [ "string" ],
   "Text": "string",
   "TextType": "string",
   "VoiceId": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

Engine (p. 231)

Specifies the engine (`standard` or `neural`) for Amazon Polly to use when processing input text for speech synthesis. Using a voice that is not supported for the engine selected will result in an error.

Type: String

Valid Values: `standard | neural`

Required: No

LanguageCode (p. 231)

Optional language code for the Synthesize Speech request. This is only necessary if using a bilingual voice, such as Aditi, which can be used for either Indian English (en-IN) or Hindi (hi-IN).

If a bilingual voice is used and no language code is specified, Amazon Polly will use the default language of the bilingual voice. The default language for any voice is the one returned by the DescribeVoices operation for the `LanguageCode` parameter. For example, if no language code is specified, Aditi will use Indian English rather than Hindi.

Type: String

Valid Values: `arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR`

Required: No

LexiconNames (p. 231)

List of one or more pronunciation lexicon names you want the service to apply during synthesis. Lexicons are applied only if the language of the lexicon is the same as the language of the voice. For information about storing lexicons, see PutLexicon.

Type: Array of strings

Array Members: Maximum number of 5 items.

Pattern: `[0-9A-Za-z]{1,20}`

Required: No

OutputFormat (p. 231)

The format in which the returned output will be encoded. For audio stream, this will be mp3, ogg_vorbis, or pcm. For speech marks, this will be json.

When pcm is used, the content returned is audio/pcm in a signed 16-bit, 1 channel (mono), little-endian format.

Type: String

Valid Values: `json | mp3 | ogg_vorbis | pcm`

Required: Yes

SampleRate (p. 231)

The audio frequency specified in Hz.

The valid values for mp3 and ogg_vorbis are "8000", "16000", "22050", and "24000". The default value for standard voices is "22050". The default value for neural voices is "24000".

Valid values for pcm are "8000" and "16000" The default value is "16000".

Type: String

Required: No

SpeechMarkTypes (p. 231)

The type of speech marks returned for the input text.

Type: Array of strings

Array Members: Maximum number of 4 items.

Valid Values: `sentence | ssml | viseme | word`

Required: No

Text (p. 231)

Input text to synthesize. If you specify `ssml` as the `TextType`, follow the SSML format for the input text.

Type: String

Required: Yes

TextType (p. 231)

Specifies whether the input text is plain text or SSML. The default value is plain text. For more information, see Using SSML.

Type: String

Valid Values: `ssml | text`

Required: No

VoiceId (p. 231)

Voice ID to use for the synthesis. You can get a list of available voice IDs by calling the DescribeVoices operation.

Type: String

Valid Values: `Aditi | Amy | Astrid | Bianca | Brian | Camila | Carla | Carmen | Celine | Chantal | Conchita | Cristiano | Dora | Emma | Enrique | Ewa | Filiz | Geraint | Giorgio | Gwyneth | Hans | Ines | Ivy | Jacek | Jan | Joanna | Joey | Justin | Karl | Kendra | Kimberly | Lea | Liv | Lotte | Lucia | Lupe | Mads | Maja | Marlene | Mathieu | Matthew | Maxim | Mia | Miguel | Mizuki | Naja | Nicole | Penelope | Raveena | Ricardo | Ruben | Russell | Salli | Seoyeon | Takumi | Tatyana | Vicki | Vitoria | Zeina | Zhiyu`

Required: Yes

Response Syntax

```
HTTP/1.1 200
Content-Type: ContentType
x-amzn-RequestCharacters: RequestCharacters

AudioStream
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The response returns the following HTTP headers.

ContentType (p. 233)

Specifies the type audio stream. This should reflect the `OutputFormat` parameter in your request.

- If you request `mp3` as the `OutputFormat`, the `ContentType` returned is audio/mpeg.
- If you request `ogg_vorbis` as the `OutputFormat`, the `ContentType` returned is audio/ogg.
- If you request `pcm` as the `OutputFormat`, the `ContentType` returned is audio/pcm in a signed 16-bit, 1 channel (mono), little-endian format.
- If you request `json` as the `OutputFormat`, the `ContentType` returned is audio/json.

RequestCharacters (p. 233)

Number of characters synthesized.

The response returns the following as the HTTP body.

AudioStream (p. 233)

Stream containing the synthesized speech.

Errors

EngineNotSupportedException

This engine is not compatible with the voice that you have designated. Choose a new voice that is compatible with the engine or change the engine and restart the operation.

HTTP Status Code: 400

InvalidSampleRateException

The specified sample rate is not valid.

HTTP Status Code: 400

InvalidSsmlException

The SSML you provided is invalid. Verify the SSML syntax, spelling of tags and values, and then try again.

HTTP Status Code: 400

LanguageNotSupportedException

The language specified is not currently supported by Amazon Polly in this capacity.

HTTP Status Code: 400

LexiconNotFoundException

Amazon Polly can't find the specified lexicon. This could be caused by a lexicon that is missing, its name is misspelled or specifying a lexicon that is in a different region.

Verify that the lexicon exists, is in the region (see ListLexicons (p. 219)) and that you spelled its name is spelled correctly. Then try again.

HTTP Status Code: 404

MarksNotSupportedForFormatException

Speech marks are not supported for the `OutputFormat` selected. Speech marks are only available for content in `json` format.

HTTP Status Code: 400

ServiceFailureException

An unknown condition has caused a service failure.

HTTP Status Code: 500

SsmlMarksNotSupportedForTextTypeException

SSML speech marks are not supported for plain text-type input.

HTTP Status Code: 400

TextLengthExceededException

The value of the "Text" parameter is longer than the accepted limits. For the `SynthesizeSpeech` API, the limit for input text is a maximum of 6000 characters total, of which no more than 3000 can be billed characters. For the `StartSpeechSynthesisTask` API, the maximum is 200,000 characters, of which no more than 100,000 can be billed characters. SSML tags are not counted as billed characters.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Data Types

The following data types are supported:

- Lexicon (p. 236)
- LexiconAttributes (p. 237)
- LexiconDescription (p. 239)
- SynthesisTask (p. 240)
- Voice (p. 243)

Lexicon

Provides lexicon name and lexicon content in string format. For more information, see Pronunciation Lexicon Specification (PLS) Version 1.0.

Contents

Content

Lexicon content in string format. The content of a lexicon must be in PLS format.

Type: String

Required: No

Name

Name of the lexicon.

Type: String

Pattern: `[0-9A-Za-z]{1,20}`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3

LexiconAttributes

Contains metadata describing the lexicon such as the number of lexemes, language code, and so on. For more information, see Managing Lexicons.

Contents

Alphabet

Phonetic alphabet used in the lexicon. Valid values are `ipa` and `x-sampa`.

Type: String

Required: No

LanguageCode

Language code that the lexicon applies to. A lexicon with a language code such as "en" would be applied to all English languages (en-GB, en-US, en-AUS, en-WLS, and so on.

Type: String

Valid Values: `arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR`

Required: No

LastModified

Date lexicon was last modified (a timestamp value).

Type: Timestamp

Required: No

LexemesCount

Number of lexemes in the lexicon.

Type: Integer

Required: No

LexiconArn

Amazon Resource Name (ARN) of the lexicon.

Type: String

Required: No

Size

Total size of the lexicon, in characters.

Type: Integer

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3

LexiconDescription

Describes the content of the lexicon.

Contents

Attributes

Provides lexicon metadata.

Type: LexiconAttributes (p. 237) object

Required: No

Name

Name of the lexicon.

Type: String

Pattern: `[0-9A-Za-z]{1,20}`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3

SynthesisTask

SynthesisTask object that provides information about a speech synthesis task.

Contents

CreationTime

Timestamp for the time the synthesis task was started.

Type: Timestamp

Required: No

Engine

Specifies the engine (`standard` or `neural`) for Amazon Polly to use when processing input text for speech synthesis. Using a voice that is not supported for the engine selected will result in an error.

Type: String

Valid Values: `standard | neural`

Required: No

LanguageCode

Optional language code for a synthesis task. This is only necessary if using a bilingual voice, such as Aditi, which can be used for either Indian English (en-IN) or Hindi (hi-IN).

If a bilingual voice is used and no language code is specified, Amazon Polly will use the default language of the bilingual voice. The default language for any voice is the one returned by the DescribeVoices operation for the `LanguageCode` parameter. For example, if no language code is specified, Aditi will use Indian English rather than Hindi.

Type: String

Valid Values: `arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR`

Required: No

LexiconNames

List of one or more pronunciation lexicon names you want the service to apply during synthesis. Lexicons are applied only if the language of the lexicon is the same as the language of the voice.

Type: Array of strings

Array Members: Maximum number of 5 items.

Pattern: `[0-9A-Za-z]{1,20}`

Required: No

OutputFormat

The format in which the returned output will be encoded. For audio stream, this will be mp3, ogg_vorbis, or pcm. For speech marks, this will be json.

Type: String

Valid Values: `json | mp3 | ogg_vorbis | pcm`

Required: No

OutputUri

Pathway for the output speech file.

Type: String

Required: No

RequestCharacters

Number of billable characters synthesized.

Type: Integer

Required: No

SampleRate

The audio frequency specified in Hz.

The valid values for mp3 and ogg_vorbis are "8000", "16000", "22050", and "24000". The default value for standard voices is "22050". The default value for neural voices is "24000".

Valid values for pcm are "8000" and "16000" The default value is "16000".

Type: String

Required: No

SnsTopicArn

ARN for the SNS topic optionally used for providing status notification for a speech synthesis task.

Type: String

Pattern: `^arn:aws(-(cn|iso(-b)?|us-gov))?:sns:[a-z0-9_-]{1,50}:\d{12}:[a-zA-Z0-9_-]{1,256}$`

Required: No

SpeechMarkTypes

The type of speech marks returned for the input text.

Type: Array of strings

Array Members: Maximum number of 4 items.

Valid Values: `sentence | ssml | viseme | word`

Required: No

TaskId

The Amazon Polly generated identifier for a speech synthesis task.

Type: String

Pattern: `^[a-zA-Z0-9_-]{1,100}$`

Required: No

TaskStatus

Current status of the individual speech synthesis task.

Type: String

Valid Values: `scheduled | inProgress | completed | failed`

Required: No

TaskStatusReason

Reason for the current status of a specific speech synthesis task, including errors if the task has failed.

Type: String

Required: No

TextType

Specifies whether the input text is plain text or SSML. The default value is plain text.

Type: String

Valid Values: `ssml | text`

Required: No

VoiceId

Voice ID to use for the synthesis.

Type: String

Valid Values: `Aditi | Amy | Astrid | Bianca | Brian | Camila | Carla | Carmen | Celine | Chantal | Conchita | Cristiano | Dora | Emma | Enrique | Ewa | Filiz | Geraint | Giorgio | Gwyneth | Hans | Ines | Ivy | Jacek | Jan | Joanna | Joey | Justin | Karl | Kendra | Kimberly | Lea | Liv | Lotte | Lucia | Lupe | Mads | Maja | Marlene | Mathieu | Matthew | Maxim | Mia | Miguel | Mizuki | Naja | Nicole | Penelope | Raveena | Ricardo | Ruben | Russell | Salli | Seoyeon | Takumi | Tatyana | Vicki | Vitoria | Zeina | Zhiyu`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3

Voice

Description of the voice.

Contents

AdditionalLanguageCodes

Additional codes for languages available for the specified voice in addition to its default language.

For example, the default language for Aditi is Indian English (en-IN) because it was first used for that language. Since Aditi is bilingual and fluent in both Indian English and Hindi, this parameter would show the code `hi-IN`.

Type: Array of strings

Valid Values: `arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR`

Required: No

Gender

Gender of the voice.

Type: String

Valid Values: `Female | Male`

Required: No

Id

Amazon Polly assigned voice ID. This is the ID that you specify when calling the `SynthesizeSpeech` operation.

Type: String

Valid Values: `Aditi | Amy | Astrid | Bianca | Brian | Camila | Carla | Carmen | Celine | Chantal | Conchita | Cristiano | Dora | Emma | Enrique | Ewa | Filiz | Geraint | Giorgio | Gwyneth | Hans | Ines | Ivy | Jacek | Jan | Joanna | Joey | Justin | Karl | Kendra | Kimberly | Lea | Liv | Lotte | Lucia | Lupe | Mads | Maja | Marlene | Mathieu | Matthew | Maxim | Mia | Miguel | Mizuki | Naja | Nicole | Penelope | Raveena | Ricardo | Ruben | Russell | Salli | Seoyeon | Takumi | Tatyana | Vicki | Vitoria | Zeina | Zhiyu`

Required: No

LanguageCode

Language code of the voice.

Type: String

Valid Values: `arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR`

Required: No

LanguageName

Human readable name of the language in English.

Type: String

Required: No

Name

Name of the voice (for example, Salli, Kendra, etc.). This provides a human readable voice name that you might display in your application.

Type: String

Required: No

SupportedEngines

Specifies which engines (`standard` or `neural`) that are supported by a given voice.

Type: Array of strings

Valid Values: `standard | neural`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3

Document History for Amazon Polly

The following table describes important changes in each release of the *Amazon Polly Developer Guide*. For notification about updates to this documentation, you can subscribe to an RSS feed.

- **Latest documentation update:** November 25, 2019

update-history-change	update-history-description	update-history-date
New Feature (p. 245)	In addition to the Newscaster speaking style, Amazon Polly now provides a second NTTS speaking style to help you synthesize even better text to speech passages. The Conversational style uses the neural system to generate speech in a more friendly and expressive conversational style that can be used in many use cases. For more information, see NTTS Speaking Styles.	November 25, 2019
New Voices Added (p. 245)	Two new voices added: Camila (female, Portuguese-Brazil) and Lupe (female, Spanish-US).	October 23, 2019
New Feature Added (p. 245)	Addition of Amazon Polly for Windows plugin to incorporate the full range of Amazon Polly voices into Windows SAPI-compliant applications.	September 26, 2019
Major New Feature (p. 245)	In addition to the standard text-to-speech (TTS) voices supported by Amazon Polly since its launch, Amazon Polly now provides an improved Neural TTS (NTTS) system that can provide even higher quality voices, thereby providing you with the most natural and human-like text-to-speech voices possible. For more information, see Neural Text-to-Speech.	July 30, 2019
New Voices Added (p. 245)	New voices added: Lucia (female, Spanish), and Bianca (female, Italian).	August 2, 2018
New Language Added (p. 245)	New language added: Mexican Spanish (es-MX). This language uses the female voice of Mia.	August 2, 2018

New Language Added (p. 245)	New language added: Hindi (hi-IN). This voice uses the female voice of Aditi, which is also used for Indian English, making Aditi Amazon Polly's first bilingual voice.	August 2, 2018
New SSML Feature Added (p. 245)	Addition of Maximum Duration for Synthesized Speech.	July 17, 2018
New Feature Added (p. 245)	Addition of Speech synthesis of long text passages (up to 100,000 billed characters).	July 17, 2018
New Voice Added (p. 245)	New voice added: Léa (female, French).	June 5, 2018
Region Expansion (p. 245)	Expansion of Amazon Polly service to all commercial regions.	June 4, 2018
New Language Added (p. 245)	New language added: Korean (ko-KR).	June 4, 2018
Expanded Feature (p. 245)	Expansion of Amazon Polly WordPress Plugin feature, including addition of Amazon Translate capabilities.	June 4, 2018
New Voices Added (p. 245)	Two new voices added: Aditi (female, Indian English) and Seoyeon (female, Korean).	November 15, 2017
New Feature (p. 245)	Addition of new Speech Marks feature, as well as an expansion of SSML capabilities..	April 19, 2017
New Guide (p. 245)	This is the first release of the Amazon Polly Developer Guide.	November 30, 2016

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.