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.

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 not certified for use with regulated workloads such as Payment Card Industry (PCI) Data Security Standard (DSS), HIPAA (Health Insurance Portability and Accountability Act of 1996), or FedRAMP.

Some of the benefits of using Amazon Polly include:

- **High quality** – Amazon Polly uses best-in-class Text-to-Speech (TTS) technology to synthesize natural speech with high pronunciation accuracy (including abbreviations, acronym expansions, date/time interpretations, and homograph disambiguation).

- **Low latency** – Amazon Polly ensures fast response times, 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 and multiple languages, offering male and female voice options for most languages.

- **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 Text-to-Speech 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, etc. In contrast, Text-to-Speech conversion done in the 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. **Amazon Polly: How It 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. 6)** – In this section, you set up your account and test Amazon Polly speech synthesis.
3. **Example Applications (p. 114)** – This section provides additional examples that you can use to explore Amazon Polly.

Amazon Polly: How It Works

Amazon Polly converts input text into life-like speech. You just need to call the `SynthesizeSpeech` method, provide the text you wish to synthesize, select one of the available 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 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 Using SSML (p. 75).

- **Available voices** – Amazon Polly provides a portfolio of multiple languages and a variety of voices. For most languages you can select from several different voices, including both male and female. You only need to specify the voice name when calling the `SynthesizeSpeech` operation, and then the service uses this voice to convert the text to speech. Amazon Polly is not a translation service—the synthesized speech is in the language of the text. Numbers using digits (for example, *53*, not *fifty-three*) are synthesized in the language of the voice.

- **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 to consume in web and mobile applications. Or, you might request the PCM output format for 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. 6)
- Example Applications (p. 114)
- Limits in Amazon Polly (p. 136)

FAQs

This topic provides answers to questions that are commonly asked about Amazon Polly.

Topics

General Questions

Q: Can I save the synthesized speech?

You can save the output of the synthesis for use on your own system. You can also call Amazon Polly, and then encrypt the file with any encryption key and store it in Amazon Simple Storage Service (Amazon S3) or any other secure storage. The Amazon Polly `SynthesizeSpeech` call is stateless and is not associated with a customer identity. You can't retrieve it from Amazon Polly later.

Content Rendering

Q: When I use bullet points in my text, Amazon Polly doesn't render them correctly. It says "minus" every time it encounters one. What do I do?

If you use "-" (a hyphen) as a substitute for a bullet point, in some languages,Amazon Polly renders it as a minus sign. If you want to use hyphens as substitutes for a bullet point, you can do so with a lexicon entry. For more information, see Managing Lexicons (p. 90).

Q: I use the "/" (forward slash) symbol frequently in my text, especially when saying "and/or" and "yes/no." How does Amazon Polly render this?

In English, Amazon Polly renders "and/or" in speech as "and or." Currently, this rule isn't available in other languages. In languages other than English, Amazon Polly renders "yes/no" as "yes slash no." If you want to change this behavior, you can use a lexicon entry. For more information, see Managing Lexicons (p. 90).

Q: When I use text from an existing source in order to synthesize speach using the AWS CLI on a Linux machine, some UTF-8 characters do not seem work with Amazon Polly, even though the same characters seem to work properly using the Console. What is happening?

This is based in how the Unix Shell handles Unicode and isn't a Amazon Polly-specific problem. Two options are available: you can find the problem characters and replace them in the input text, or you can u tilize an alternate means of accessing Amazon Polly that does not experience this issue, such as the PHP interface. This is a known issue that we are working to address and only a few uncommon unicode characters have this issue.

Q: When I try to synthesize text from a source containing International Phonetic Alphabet (IPA) symbols, Amazon Polly doesn't recognize them and even tries to pronounce some of them. How do I fix this?

Amazon Polly does not recognize IPA symbols unless SSML (Speech Synthesis Markup Language) is used to delineate it. However, since small sections of IPA symbols usually indicate a pronunciation guide for a reader, in many cases, this section can be safely removed from the input text by simple deletion. You can

also use a lexicon to change the way this is rendered by Amazon Polly. For more information, see Using SSML (p. 75) and Managing Lexicons (p. 90).

Voices

Q: How long will the voices be available? If I choose a voice for my application now, will it still be available in five years?

To ensure continuous support for customers, we don't plan to retire any voices. This applies to both currently available and future voices.

Data Security and Confidentiality

Q: Can I choose to mask certain data fields so that they are not stored?(For instance, if I convert text with some sensitive data, but don't want it stored on the AWS systems, can I mask it?

No. Amazon Polly doesn’t currently support this functionality.

Q: The text I want to use with Amazon Polly is confidential. How is my data protected?

All text submissions are protected by Secure Sockets Layer (SSL) while in transit, and are stored using RSA encryption. We keep the service logs and text separate, so that the content can't be linked with the customer ID. As a result, Amazon Polly does not associate text submissions with customer identity.

Q: How long is data retained?

Amazon Polly retains data for 14 days. After that, it's automatically deleted from our system.

Q: Can I request that data be wiped earlier?

Yes, you can request that by contacting AWS Support.

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. 144). 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. 6)
2. Step 1.2: Create an IAM User (p. 7)

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://aws.amazon.com/, and then choose **Create an AWS Account**.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using 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 Getting Started 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. 172).

For more information about IAM, see the following:

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

Next Step

Step 2: Getting Started Using the Console (p. 8)

Step 2: Getting Started Using the 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

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

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. (Optional) Choose **SSML**.
4. Choose a language and region, then choose a voice.
5. Choose **Listen to speech**.

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

- Exercise 2: Synthesizing Speech (Plain Text Input) (p. 8)
- Using SSML with the Amazon Polly Console (p. 75)
- Applying Lexicons Using the Console (Synthesize Speech) (p. 92)

Exercise 2: Synthesizing Speech (Plain Text Input)

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

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 **Choose a language and region**, choose **English US**, then choose the voice you want to use for this text.

5. To listen to the speech immediately, choose **Listen to speech**.
6. 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 you want, and then choose **Change**.

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

- Applying Lexicons Using the Console (Synthesize Speech) (p. 92)
- Using SSML with the Amazon Polly Console (p. 75)

Next Step

Step 3: Getting Started Using the AWS CLI (p. 9)

Step 3: Getting Started Using the AWS CLI

Using the AWS CLI you can perform almost all Amazon Polly operations you can perform using the Amazon Polly console. You cannot listen to the 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 the file.

Topics

- Step 3.1: Set Up the AWS Command Line Interface (AWS CLI) (p. 9)
- Step 3.2: Getting Started Exercise Using the AWS CLI (p. 11)

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

Follow the steps to download and configure the AWS Command Line Interface (AWS CLI).

> **Important**
> You don't need the AWS CLI to perform the steps in this Getting Started 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. 11), 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 executing 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*.

If you specify one of the Amazon Polly supported regions when you configure the AWS CLI, you can omit the following line from the AWS CLI code examples. If you specify a region not supported by Amazon Polly in your AWS CLI configuration (for example, if you're an existing AWS customer using other services in regions that don't support Amazon Polly), you must include the following line:

```
--region polly-supported-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 may not be available without reconfiguring 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 command prompt:

```
aws polly help
```

If a description of Amazon Polly and a list of valid commands is displayed, appears in the AWS CLI window. If Amazon Polly is available in the AWS CLI and can be used immediately. In this case, you can skip the remainder 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 the following topic in the *AWS Command Line Interface User Guide*: Installing the AWS Command Line Interface.

      or

   b. Download the file service-2.json.

      At the command prompt, run the following:

      ```
      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. 11)

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.

If you specified one of the Amazon Polly supported regions when you configured the AWS CLI, you can omit the following line from the AWS CLI code examples. If you specified a region not supported by Amazon Polly in your AWS CLI configuration (for example, if you're an existing AWS customer using other services in regions that don't support Amazon Polly), you must include the following line:

```
--region polly-supported-aws-region
```

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 above operation produces the following output to the console.

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

2. Play the resulting `hello.mp3` file to verify the synthesized speech.
3. You can 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": "Kendra",
            "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 DescribeVoices (p. 147). 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
```

What's Next?

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 you to set up Python and test the example code provided in the following section. For additional examples, see Example Applications (p. 114).

Set Up Python and Test an Example

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

To test Example Python Code

The following Python code example does the following:

- 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 on your local disk (`speech.mp3`).
- 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 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(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 Mac 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. 114).

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.

Topics

Speech Mark Types

You request speech marks in the AWS CLI using the `speech-mark-types` option for the `synthesize-speech` command. You specify the metadata elements that you want to return from your input text. You can request as many as four types of metada but you must specify at least one per request. No audio output is generated with the request.

```
--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. 15).
- **ssml** – Describes a <mark> element from the SSML input text. For more information, see Using SSML (p. 75).

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 lists the full set of International Phonetic Alphabet (IPA) phonemes and the Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) symbols as well as their corresponding visemes for American English language voices.

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

IPA	X-SAMPA	Description	Example	Viseme
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**rena	@
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

IPA	X-SAMPA	Description	Example	Viseme
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

For all available languages, see Phoneme/Viseme Tables for Supported Languages (p. 21).

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 of the start of the object in the input text (not including viseme marks)
- **end** – the offset in bytes 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 foolowing `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. 15).

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 Using the Amazon Polly 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 from the 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 Using SSML (p. 75).
12. Choose **Change**.
13. Choose **Download Speech Marks**.

Phoneme/Viseme Tables for Supported Languages

This section provides phoneme/viseme tables for the languages supported by Amazon Polly.

Topics

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

IPA	X-SAMPA	Description	Example	Viseme
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
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	X
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

IPA	X-SAMPA	Description	Example	Viseme
œː	9:	long open-mid front rounded vowel	h**ø**ne, g**ø**re	O
ə	@	mid central vowel	an**e**	@
æː	{:	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
ɑː	A:	long open back unrounded vowel	r**a**se	a

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
ʤ	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

IPA	X-SAMPA	Description	Example	Viseme
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
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

IPA	X-SAMPA	Description	Example	Viseme
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
o:	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
y:	y:	long close front rounded vowel	f**uu**t	u
ʏ	Y	near-close near-front rounded vowel	h**u**t	u

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
ʤ	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

IPA	X-SAMPA	Description	Example	Viseme
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
ɫ	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
z	z	voiced alveolar fricative	**z**ero	s
ʒ	Z	voiced postalveolar fricative	vi**s**ion	S
Vowels				

IPA	X-SAMPA	Description	Example	Viseme
ə	@	mid central vowel	**a**rena	@
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

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.

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
ʤ	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
ʃ	S	voiceless postalveolar fricative	**sh**ip	S
t	t	voiceless alveolar plosive	**t**ask	t

IPA	X-SAMPA	Description	Example	Viseme
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**rena	@
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

IPA	X-SAMPA	Description	Example	Viseme
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

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
ʤ	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

IPA	X-SAMPA	Description	Example	Viseme
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
z	z	voiced alveolar fricative	**z**ero	s
ʒ	Z	voiced postalveolar fricative	vi**s**ion	S
Vowels				
ə	@	mid central vowel	**a**rena	@
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

IPA	X-SAMPA	Description	Example	Viseme
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

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
ʤ	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
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				

IPA	X-SAMPA	Description	Example	Viseme
ə	@	mid-central vowel	**a**rena	@
ɚ	@'	mid-central r-colored vowel	reade**r**	@
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
iː	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

English, Welsh (en-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				

IPA	X-SAMPA	Description	Example	Viseme
b	b	voiced bilabial plosive	**b**ed	p
d	d	voiced alveolar plosive	**d**ig	t
ʤ	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
ʃ	S	voiceless postalveolar fricative	**sh**ip	S
t	t	voiceless alveolar plosive	**t**ask	t
ʧ	tS	voiceless postalveolar affricate	**ch**art	S

IPA	X-SAMPA	Description	Example	Viseme
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**rena	@
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

IPA	X-SAMPA	Description	Example	Viseme
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

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
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	parki**ng**	k
p	p	voiceless bilabial plosive	**p**omme	p
ʁ	R\	uvular trill	amou**r**eux	k

IPA	X-SAMPA	Description	Example	Viseme
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	j**e**	@
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
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**o**mme	o
ɔ	O	open-mid back rounded vowel	h**ô**pital	O

IPA	X-SAMPA	Description	Example	Viseme
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

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
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	parki**ng**	k
p	p	voiceless bilabial plosive	**p**omme	p
ʁ	R\	uvular trill	amou**r**eux	k

IPA	X-SAMPA	Description	Example	Viseme
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	j**e**	@
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
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**o**mme	o
ɔ	O	open-mid back rounded vowel	h**ô**pital	O

IPA	X-SAMPA	Description	Example	Viseme
õ	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

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
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

IPA	X-SAMPA	Description	Example	Viseme
p	p	Voiceless bilabial plosive	**P**ark	p
p͡f	pf	Voiceless labiodental affricate	A**pf**el	
ʁ	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
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

IPA	X-SAMPA	Description	Example	Viseme
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
ɔʏ	OY	diphthong	n**eu**	O
uː	u:	long close back rounded vowel	Br**u**der	u
ʊ	U	near-close near-back rounded vowel	W**u**nder	u
yː	y:	long close front rounded vowel	k**ühl**	u
ʏ	Y	near-close near-front rounded vowel	K**ü**che	u

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

IPA	X-SAMPA	Description	Example	Viseme
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
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	gól**f**	t
l̥	l_0	voiceless alveolar lateral approximant	fól**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

IPA	X-SAMPA	Description	Example	Viseme
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
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

IPA	X-SAMPA	Description	Example	Viseme
ɔ:	O:	long open-mid back rounded vowel	ullarb**o**lur	O
ɔu	Ou	diphthong	t**ó**lf	O
ɔu:	Ou:	diphthong	fj**ó**rir	O
u	u	close back rounded vowel	st**ú**lkan	u
u:	u:	long close back rounded vowel	fr**ú**	u
ʏ	Y	near-close near-front rounded vowel	t**í**u	u
ʏ:	Y	long near-close near-front rounded vowel	gr**u**ninn	u

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

IPA	X-SAMPA	Description	Example	Viseme
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
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

IPA	X-SAMPA	Description	Example	Viseme
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

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, **k**i**ky**ou	k
ç	C	voiceless palatal fricative	人, **h**ito	k
d	d	voiced alveolar plosive	濁点, **d**akuten	t
ʥ	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

IPA	X-SAMPA	Description	Example	Viseme
ɺj	l\j	alveolar lateral flap, palatal approximant	流行, **ry**uukou	r
m	m	bilabial nasal	飯, **m**eshi	p
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**ch**i	J
w	w	labial-velar approximant	電話, den**w**a	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**a**n**a**	a
eː	e:_o	long mid front unrounded vowel	学生, gakus**ei**	@
e	e_o	mid front unrounded vowel	歴, **r**eki	@
i	i	close front unrounded vowel	気, k**i**	i
iː	i:	long close front unrounded vowel	詩歌, sh**ii**ka	i

IPA	X-SAMPA	Description	Example	Viseme
ɯ	M	close back unrounded vowel	運, **u**n	i
ɯː	M:	long close back unrounded vowel	宗教, sh**uu**kyou	i
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

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	pr**ø**v	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

IPA	X-SAMPA	Description	Example	Viseme
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
ʈ	t`	voiceless retroflex plosive	ha**rdt**	t
ʊ	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

IPA	X-SAMPA	Description	Example	Viseme
e:	e:	long close-mid front unrounded vowel	s**e**n	e
ɛ	E	open-mid front unrounded vowel	s**e**nd	E
i:	i:	long close front unrounded vowel	v**i**n	i
ɪ	I	near-close near-front unrounded vowel	v**i**nd	i
o:	o:	long close-mid back rounded vowel	v**å**t	o
ɔ	O	open-mid back rounded vowel	v**å**tt	O
u:	u:	long close back rounded vowel	b**o**k	u
ʊ	U	near-close near-back rounded vowel	b**u**kk	u
y:	y:	long close front rounded vowel	l**y**n	u
ʏ	Y	near-close near-front rounded vowel	l**y**nne	u

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, łó**dź**	J
d͡ʐ	dz`	voiced retroflex affricate	**dż**em, **dż**ungla	S

IPA	X-SAMPA	Description	Example	Viseme
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
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, **n**ie	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, be**z**	s
ʑ	z\	voiced alveolo-palatal fricative	**ź**rebię, bieli**ź**nie	J

IPA	X-SAMPA	Description	Example	Viseme
ʐ	z`	voiced retroflex fricative	**ż**ar, **ż**ona	S
Vowels				
a	a	open front unrounded vowel	j**a**	a
ɛ	E	open-mid front unrounded vowel	**e**cho	E
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

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

IPA	X-SAMPA	Description	Example	Viseme
k	k	voiceless velar plosive	**c**acto	k
l	l	alveolar lateral approximant	ga**l**o	t
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
ʁ	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

IPA	X-SAMPA	Description	Example	Viseme
i	i	close front unrounded vowel	**l**ingueta	i
ɪ̃	i~	nasal close front unrounded vowel	c**i**nto	i
o	o	close-mid back rounded vowel	p**o**der	o
õ̞	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**ui**	u
ũ̞	u~	nasal close back rounded vowel	s**u**nto	u

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	p**i**ra	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	**i**ngueta	i

IPA	X-SAMPA	Description	Example	Viseme
ɪ̃	i~	nasal close front unrounded vowel	c**i**nto	i
o	o	close-mid back rounded vowel	p**o**der	o
o̜	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
u̜	u~	nasal close back rounded vowel	s**u**nto	u

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
k	k	voiceless velar plosive	**c**oș	k
l	l	alveolar lateral approximant	**l**ampa	t
m	m	bilabial nasal	**m**ama	p

IPA	X-SAMPA	Description	Example	Viseme
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	maș**i**nă	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
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

IPA	X-SAMPA	Description	Example	Viseme
ɨ	1	close central unrounded vowel	Româ**î**nia	i

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
bʲ	b'	palatalized voiced bilabial plosive	**б**юро	p
d	d	voiced alveolar plosive	**д**ом	t
dʲ	d'	palatalized voiced alveolar plosive	**д**я**д**я	t
f	f	voiceless labiodental fricative	**ф**лаг	f
fʲ	f'	palatalized voiceless labiodental fricative	**ф**евраль	f
g	g	voiced velar plosive	но**г**а	k
gʲ	g'	palatalized voiced velar plosive	**г**ерой	k
j	j	palatal approximant	дизай**й**н, **я**щик	i
k	k	voiceless velar plosive	**к**от	k
kʲ	k'	palatalized voiceless velar plosive	**к**ино	k
l	l	alveolar lateral approximant	**л**ампа	t
lʲ	l'	palatalized alveolar lateral approximant	**л**ес	t
m	m	bilabial nasal	**м**ама	p
mʲ	m'	palatalized bilabial nasal	**м**яч	p

IPA	X-SAMPA	Description	Example	Viseme
n	n	alveolar nasal	**н**ос	t
nʲ	n'	palatalized alveolar nasal	**н**я**н**я	t
p	p	voiceless bilabial plosive	**п**апа	p
pʲ	p'	palatalized voiceless bilabial plosive	**п**еро	p
r	r	alveolar trill	**р**оза	r
rʲ	r'	palatalized alveolar trill	**р**юмка	r
s	s	voiceless alveolar fricative	**с**ыр	s
sʲ	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
tʲ	t'	palatalized voiceless alveolar plosive	**тёт**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
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

IPA	X-SAMPA	Description	Example	Viseme
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**ш	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				
ɾ	4	alveolar flap	pe**r**o, b**r**avo, amo**r**, ete**r**no	t
b	b	voiced bilabial plosive	b**e**stia	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

IPA	X-SAMPA	Description	Example	Viseme
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, radi**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
ʧ	tS	voiceless postalveolar affricate	**ch**ubasco	S
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				

IPA	X-SAMPA	Description	Example	Viseme
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
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	long 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

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
ʤ	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

IPA	X-SAMPA	Description	Example	Viseme
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**rena	@
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
iː	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

IPA	X-SAMPA	Description	Example	Viseme
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

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
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

IPA	X-SAMPA	Description	Example	Viseme
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
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

IPA	X-SAMPA	Description	Example	Viseme
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
yː	y:	long close front rounded vowel	s**y**l, s**y**l	u

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
ɾ̥	4_0_r	voiceless fricated alveolar flap	bi**r**	t
ɾ̠	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

IPA	X-SAMPA	Description	Example	Viseme
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
l	l	alveolar lateral approximant	**l**a**l**e	t
ɬ	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
͡ʧ	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

IPA	X-SAMPA	Description	Example	Viseme
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
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	k**ı**l	i
o	o	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
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

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
d	d	voiced alveolar plosive	**d**eg	t
ʤ	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**ng**henion	k
p	p	voiceless bilabial plosive	**p**ump	p
r	r	alveolar trill	**rh**oi	r
r̥	r_0	voiceless alveolar trill	ga**r**w	r
s	s	voiceless alveolar fricative	**s**aith	s

IPA	X-SAMPA	Description	Example	Viseme
ʃ	S	voiceless postalveolar fricative	**si**awns	S
t	t	voiceless alveolar plosive	**t**egan	t
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

IPA	X-SAMPA	Description	Example	Viseme
oː	o:	long close-mid back rounded vowel	**o**ddi	o
ɔ	O	open-mid back rounded vowel	**o**ddieithr	O
ɔi	Oi	diphthong	tr**oi**	O
ɔu	Ou	diphthong	r**ow**nd	O
uː	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

Using SSML

Amazon Polly generates speech from both plain text input and Speech Synthesis Markup Language (SSML) documents that conform to SSML version 1.1. Using SSML tags, you can customize and control aspects of speech such as pronunciation, volume, and speech rate.

Amazon Polly supports SSML 1.1 as defined in the following W3C recommendation:

- Speech Synthesis Markup Language (SSML) Version 1.1, W3C Recommendation 7 September 2010

Some of the elements in the SSML W3C recommendation are not supported. For more information, see Limits in Amazon Polly (p. 136).

This section provides simple examples of SSML that can be used to generate and control speech output. The examples also provide the `synthesize-speech` AWS CLI command to test these examples.

Topics

Using SSML with the Amazon Polly Console

Amazon Polly supports version 1.1 SSML as defined by the W3C. This section covers how to use SSML input for speech synthesis in the Amazon Polly console.

Using SSML with the Amazon Polly Console

The following procedure synthesizes speech using SSML input. Except for steps 3 and 4 below, the steps in this example are identical to those in Exercise 2: Synthesizing Speech (Plain Text Input) (p. 8).

To synthesize speech using the Amazon Polly console (SSML input)

In this example we introduce SSML tagging to substitute "World Wide Web Consortium" for "W3C". Compare the results of this exercise with that of Applying Lexicons Using the Console (Synthesize Speech) (p. 92) 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. If needed, 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, "Score!" quite loudly. When his boss stared at him, he repeated
<amazon:effect name="whispered">"Score"</amazon:effect> in a whisper.</speak>
```

   The SSML tags inform Amazon Polly that the text should be rendered in a specified way.

   - `<break time="1s"/>` instructs Amazon Polly to pause 1 second between the initial two sentences.
   - `<sub alias="World Wide Web Consortium">W3C</sub>` instructs Amazon Polly to substitute "World Wide Web Consortium" for the acronym "W3C".
   - `<amazon:effect name="whispered">Score</amazon:effect>` instructs Amazon Polly to say the second "Score" in a whispered voice.

   For more information on SSML with examples, see Supported SSML Tags (p. 81)
5. For **Choose a language and region**, choose **English US**, then choose the voice you want.
6. To listen to the speech immediately, choose **Listen to speech**.
7. To save the speech file, choose **Download [format]** if the format is the one you want. Otherwise choose **Change file format**, choose the format you want, and then choose **Change**. Choose **Download [format]**.

Related Console Examples

- Exercise 2: Synthesizing Speech (Plain Text Input) (p. 8)
- Applying Lexicons Using the Console (Synthesize Speech) (p. 92)

Note
When entering the input text in the AWS CLI, quotation marks are used around the input text to differentiate it from the surrounding code. Because the Amazon Polly console does not show you the code, quotation marks are not used around the input text in the console.

Next Step

Applying Lexicons Using the Console (Synthesize Speech) (p. 92)

Using SSML with the AWS CLI

The following topics demonstrate how to use SSML input with the AWS CLI for Amazon Polly.

Example 1: Passing SSML Through the Synthesize-Speech Command

In the following `synthesize-speech` command, you specify a simple SSML string with only the required opening and closing `<speak></speak>` tags and the quotation marks that surround them. (Optionally, you can specify the full document header too). Because plain text is the default, the command also specifies the `--text-type` parameter to indicate that the input text is SSML. The only required elements for an SSML string are the input text (targeted by the generated speech), `output-format`, and `voice-id`.

Important
Although you do not use quotation marks around the input text in the Amazon Polly Console, they are required when using the AWS CLI or other code, both for plain text and SSML. It is also important that the quotation marks around the entire input text and the quotation marks within the text are differentiated.
For example, you can use single quote marks (') surrounding the entire input text, and standard quotation marks (") in any interior tags or use the reverse. (When using the AWS CLI, both options will work for Unix, Linux, and macOS. Standard quotations marks for the input text and single quote marks for interior tags are required when using a Windows command prompt.)
Thus you can use either

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

or

```
--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
```

Play the resulting `speech.mp3` file to verify the synthesized speech.

Example 2: Synthesizing a Full SSML Document

In this example you save SSML content to a file and specify the file name in the `synthesize-speech` command. This example uses the following SSML.

```
<?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>
```

Note that 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 voice selected affect the `SynthesizeSpeech` operation, see Using the xml:lang Attribute (p. 81).

To test the SSML

1. Save the SSML to a file (`example.xml`).
2. Run the following `synthesize-speech` command from the path where the XML file is stored and specify the SSML as input. Note that because this points to a file rather than containing the actual input text, no quotation marks are needed.

   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 file://example.xml \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

3. Play the `speech.mp3` file to verify the synthesized speech.

Example 3: Using Common SSML Tags

This section explains how to use some common SSML tags to achieve specific results. For more examples, see Speech Synthesis Markup Language (SSML) Version 1.1.

You can use the `synthesize-speech` command to test the examples in this section.

Using 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" in the resulting speech.

```
<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 <break time="300ms"/> World</speak>' \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

Play the resulting `speech.mp3` file to verify the synthesized speech.

Using the <prosody> Element

This element enables you to control pitch, speaking rate, and volume of speech.

- The following SSML uses the `<prosody>` element to control volume:

  ```
  <speak><prosody volume="+20dB">Hello world</prosody></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><prosody volume="+20dB">Hello world</prosody></speak>' \
  --output-format mp3 \
  --voice-id Joanna \
  speech.mp3
  ```

- The following SSML uses the `<prosody>` element to control pitch:

```
<speak><prosody pitch="x-high">Hello world</prosody></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><prosody pitch="x-high">Hello world</prosody></speak>' \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

- The following SSML uses the `<prosody>` element to specify the speech rate:

```
<speak><prosody rate="x-fast">Hello world</prosody></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><prosody rate="x-fast">Hello world</prosody></speak>' \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

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

```
<speak><prosody volume="x-loud" pitch="x-high" rate="x-fast">Hello world</prosody></
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><prosody volume="x-loud" pitch="x-high" rate="x-fast">Hello world</
prosody></speak>' \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

Play the resulting `speech.mp3` file to verify the synthesized speech.

Using a whispered voice

The following synthesize-speech command uses the `<amazon:effect name="whispered">` element to say the words "little lamb" in a whispered voice in the resulting speech:

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

This effect can be enhanced by slowing the whispered speech slightly using the <prosody> element.

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> Mary has a <prosody rate="-10%"><amazon:effect name="whispered"> \
little lamb.</amazon:effect></prosody></speak>' \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

Play the resulting `speech.mp3` file to verify the synthesized speech.

Using the <emphasis> Element

This element enables you to specify the stress or prominence to apply when speaking a specified word or phrase.

```
<speak><emphasis level="strong">Hello</emphasis> world how are you?</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><emphasis level="strong">Hello</emphasis> world how are you?</speak>' \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

Play the resulting `speech.mp3` file to verify the synthesized speech.

Example 4: Controlling Pronunciation

This example explains some of the common SSML tags you can use to control pronunciation.

Using the <say-as> Element

This element enables you to provide information about the type of text contained within the element. In the following SSML, `<say-as>` indicates that the text 4/6 should be treated as a date value with format day/month.

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

```
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

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

Play the resulting `speech.mp3` file to verify the synthesized speech.

Using the xml:lang Attribute

You can improve pronunciation of words that are foreign to the input text language by specifying the target language using the `xml:lang` attribute. This forces the TTS engine to apply different pronunciation rules for the words that are specific to the target language. The examples below show different combinations of languages for the input text and the voices you can specify in the `synthesize-speech` call. You can test these examples using the `synthesize-speech` command to verify the results.

For a complete list of which languages are available, see Languages Supported by Amazon Polly (p. 134).

In this example, the chosen voice is a US English voice. Amazon Polly assumes the input text is in the same language as the selected voice. To achieve a Spanish pronunciation of specific words, you need to mark the targeted words as Spanish.

```
aws polly synthesize-speech \
--text-type ssml \
--text '<speak>That restaurant is terrific. <lang xml:lang="es-ES">Mucho gusto.</lang></
speak>' \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

Because the language of the input text is specified, Amazon Polly maps the resulting Spanish phonemes to English phonemes of the smallest acoustic distance. As a result, Salli reads the text as a US native speaker who knows the correct Spanish pronunciation, but in a US English accent.

> **Note**
> This practice is limited to pairs of languages available in the Amazon Polly language portfolio. Some language pairs work better than others because of the phonological structure of the languages.

Play the resulting `speech.mp3` file to listen to the synthesized speech.

SSML Tags in Amazon Polly

Using SSML-enhanced input text enables you to exert 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 alter it in another way such as changing the speech rate or pitch. Amazon Polly provides this type of control and more using a subset of the SSML markup tags as defined by Speech Synthesis Markup Language (SSML) Version 1.1, W3C Recommendation

Supported SSML Tags

Amazon Polly supports the following SSML tags:

Note that any SSML tags in your input text that are unsupported are automatically ignored when Amazon Polly processes it.

<speak>

The `<speak>` tag is the root element of all Amazon Polly SSML text. All SSML-enhanced text to be spoken must be included within this tag.

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

<break>

This tag indicates a pause in the speech. The length of the pause can be set with either a `strength` or `time` attribute. If no attribute is used with a `break` tag, the default `<break strength="medium">` is used.

The following values can be used with the `strength` attribute:

- `none`: No pause occurs. This can be used to remove a pause that would normally occur (such as after a period).
- `x-weak`: Same as `none`.
- `weak`: Treats adjacent words as if they are separated by a single comma.
- `medium`: Same as `weak`.
- `strong`: Treats adjacent words as if they are separated by a sentence break (equivalent to the `<s>` tag).
- `x-strong`: Treats adjacent words as if they are separated by a paragraphy break (equivalent to the `<p>` tag).

The following values can be used with the `time` attribute:

- `s`: Duration of the pause in seconds. A pause may be up to `10s` in duration.
- `ms`: Duration of the pause in milliseconds. A pause may be up to `10000ms`.

For example:

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

<lang>

This tag indicates the language of a specific word or phrase. Foreign language words and phrases are generally rendered better audibly when they are enclosed in inside a `<lang>` tag. The language in which you want to render the pronunciation must be specified using the `xml:lang` attribute.

For a complete list of which languages are available, see Languages Supported by Amazon Polly (p. 134).

The actual language in which the word or phrase is rendered is the language of the `voice-id` used, and not the language of the `<lang>` tag. The tag serves only to set the pronunciation of the text, when it is different from the language of the `voice-id`.

For example:

When the `voice-id` is Joanna (American English), run the following:

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

Amazon Polly will say this in the Joanna voice without attempting to use a French accent.

When the same voice is used with the following `<lang>` tab:

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

Amazon Polly will pronounce this in the Joanna voice using American-accented French.

However, because Joanna is not a native French voice, pronunciation is based on the language the voice speaks natively. For example, while a perfect French pronunciation would feature an uvual trill /R/ in the word adore, Joanna's American English voice will pronounce it as this phoneme as the corresponding sound /r/.

In another, when the `voice-id` of Giorgio (Italian) is used with the following text:

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

Amazon Polly will say this in the Giorgio voice with no attempt to use a non-Italian pronunciation.

When the same voice is used with the following `<lang>` tab:

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

Amazon Polly will pronounce this in the Giorgio voice using Italian-accented English.

<mark>

This tag provides the user with the ability to place a custom tag within the text. No action is taken on the tag by Amazon Polly, but when SSML metadata is returned, the position of this tag will also be returned.

This tag may be anything designated by the user with the following format:

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

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

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

the following SSML metadata might be returned by Amazon Polly

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

<p>

This tag indicates a paragraph in the text. This is equivalent to specifying a pause with `<break strength="x-strong"/>`. As with the `<s>` tag, this tag needs to enclose the sentence in question.

```
<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>
```

<phoneme>

This tag provides a phonetic pronunciation (the phonemes) for the indicated text.

Two attributes are required with the `phoneme` tag:

- `alphabet`
  - `ipa` — Indicates the phoneme uses The International Phonetic Alphabet (IPA) system.
  - `x-sampa` — Indicates the phoneme uses The Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA) system.
- `ph`
  - Indicates the phonetic symbols to be used for pronunciation. Amazon Polly supports standard IPA and X-SAMPA phonetic symbols. For more information see Phonetic Tables Used by Amazon Polly (p. 88)

When using the `phoneme` tag, Amazon Polly uses the pronunciation provided in the `ph` attribute rather than the one associated by default with the text contained within the tags. However, you should still provide human-readable text within the tags.

For instance, the word "pecan" can be pronounced two different ways. In the following example, the word “pecan” is assigned a different custom pronunciation in each line. Amazon Polly thus uses the pronunciation provided in the `ph` attributes instead of the default pronunciation:

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

Note
In some cases phonetic symbols may include single or double quotes, which conflict with the quote marks used to mark the strings being spoken. Use a backslash in these cases to mark the symbol so it doesn’t conflict with the string marker.
For example, when you use the IPA symbol for primary stress (a single quote mark), you would mark it as **\'**. The X-SAMPA symbol for an open mid-central unrounded r-colored vowel (**3'**) is marked as **3\'**. The single or double quote string marker is not changed.

<prosody>

The `prosody` tag enables you to control the volume, rate, and pitch of the delivery of the text.

The following values can be used for the `volume` attribute, to modify the volume of the speech:

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

The following values can be used for the `rate` attribute, to modify the rate of speech:

- `x-slow`, `slow`, `medium`, `fast`, `x-fast`: Sets the rate of speech to a predefined value for the current voice.

The following values can be used for the `pitch` attribute, to modify the pitch of the voice:

- `default`: Resets pitch to the default pitch 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 percentile change in the current pitch level of the current voice. For example, `+4%`, or `-2%`.

For example, prosody for a passage could be set in the following ways:

```
<speak>Prosody can be used to change the way words sound. The following words are \
<prosody volume="x-loud"> quite a bit louder than the rest of this passage. \
</prosody> Each morning when I wake up, <prosody rate="x-slow">I speak \
quite slowly and deliberately until I have my coffee.</prosody> I can also \
change the pitch of my voice using prosody. Do you like <prosody pitch="+5%"> \
speech with a pitch that is higher,</prosody> or <prosody pitch="-10%"> \
is a lower pitch preferable?</prosody></speak>
```

<s>

This tag indicates a sentence in the text. This is equivalent to:

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

Unlike the `<break strength="strong"/>` tag, this tag needs to enclose the sentence in question.

In this example, the `<s>` tag creates a short pause after both the first and second sentences. The final sentence has no `<s>` tag but also has a short pause after it because it contains 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>
```

<say-as>

This tag indicates how the input text should be interpreted. This enables you provide additional context to eliminate any ambiguity on how Amazon Polly should render the text.

When using the `say-as` tag, you need to indicate how Amazon Polly should interpret the text with the `interpret-as` attribute.

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

- `character` or `spell-out`: spells out each letter, as in a-b-c.
- `cardinal` or `number`: pronounces the value as a cardinal number, as in 1,234.
- `ordinal`: pronounces the value as an ordinal number, as in 1,234th.
- `digits`: pronounces each digit of the number individually, as in 1-2-3-4.
- `fraction`: pronounces the value as a fraction. This works for both common fractions such as 3/20, and mixed fractions, such as 2 1/2.
- `unit`: interprets a value as a measurement. The value should be followed by either a number or a fraction followed by a unit (with no space in between), or by just a unit, as in `1meter`.
- `date`: interprets the value as a date. The format of the date must be specified with the format attribute (see below).
- `time`: interprets the value as duration in minutes and seconds, as in `1'21"`.
- `address`: interprets the value as part of a street address.
- `expletive`: "bleeps" out the content included within the tag.
- `telephone`: interprets the value as a 7-digit or 10-digit telephone, as in `2025551212` This can also handle telephone extensions, as in `2025551212x345`.

The Amazon Polly service 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 will interpret it as a 10-digit telephone number and say each individual digit individually, with a brief pause for each dash. It isn't necessary to use `<say-as interpret-as="telephone">` in this case. However, if you provide the text "2025551212" and want Amazon Polly to say it as a phone number, you need to use `<say-as interpret-as="telephone">`.

The logic underlying the interpretation of 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, e.g. "double five" or "triple four"). You can test the following example with a US voice and with a UK voice to demonstrate the difference:

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

Dates

The following values can be used when `interpret-as` is set to `date` to indicate the format of the date:

- `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 this format is used, you can include question marks for portions of the date to leave out. For instance, `<say-as interpret-as="date">????0922</say-as>` would be "September 22nd."

<sub>

This tag substitutes the pronunciation inside the `alias` attribute for the pronunciation of the text enclosed in the tag. A common use for this tag is to provide expanded pronunciation for acronyms and abbreviations.

```
<speak>My favorite chemical element is <sub alias="mercury">Hg</sub>, it looks cool. </
speak>
```

<w>

Similar to `<say-as>`, this tag customizes word pronunciation by specifying the word's part of speech. The attribute `role` is included with the tag to specify the part of speech indicated.

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

- `amazon:VB`: interprets the word as a verb (present simple).
- `amazon:VBD`: interprets the word as past tense or as a past participle.
- `amazon:SENSE_1`: uses the non-default sense of the word when present. For example, the default meaning of "bass" is that of the low note in a chord or the lowest part of the musical range. This is pronounced differently than the alternate sense a freshwater fish, also called a "bass" but pronounced differently. Using `<w role="amazon:SENSE_1">bass</w>` renders the non-default pronunciation (freshwater fish) for the audio text.

Depending how you intend to use it, the American English pronunciation of the word "read," depending on how it's being used:

```
<speak>
The present simple form of the word is pronounced <w role="amazon:VB">read</w>,
where the past tense or past participle is pronounced <w role="amazon:VBD">read</w>.
</speak>
```

<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.

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 will be whispered, whereas the phrase "she said" will be 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 desire.

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.

Phonetic Tables Used by Amazon Polly

The following provides a phonetic pronunciation (the phonemes) for Amazon Polly SSML tags for American English.

IPA	X-SAMPA	Description	Example
Consonants			
b	b	Voiced bilabial plosive	**b**ed
d	d	Voiced alveolar plosive	**d**ig
d͡ʒ	dZ	Voiced postalveolar affricate	**j**ump
ð	D	Voiced dental fricative	**th**en
f	f	Voiceless labiodental fricative	**f**ive
g	g	Voiced velar plosive	**g**ame
h	h	Voiceless glottal fricative	**h**ouse
j	j	Palatal approximant	**y**es
k	k	Voiceless velar plosive	**c**at
l	l	Alveolar lateral approximant	**l**ay
m	m	Bilabial nasal	**m**ouse
n	n	Alveolar nasal	**n**ap
ŋ	N	Velar nasal	thi**ng**
p	p	Voiceless bilabial plosive	s**p**eak
ɹ	r\	Alveolar approximant	**r**ed
s	s	Voiceless alveolar fricative	**s**eem
ʃ	S	Voiceless postalveolar fricative	**sh**ip
t	t	Voiceless alveolar plosive	**t**rap
t͡ʃ	tS	Voiceless postalveolar affricate	**ch**art
Θ	T	Voiceless dental fricative	**th**in
v	v	Voiced labiodental fricative	**v**est

IPA	X-SAMPA	Description	Example
w	w	Labial-velar approximant	**w**est
z	z	Voiced alveolar fricative	**z**ero
ʒ	Z	Voiced postalveolar fricative	vi**s**ion
Vowels			
ə	@	Mid central vowel	**a**rena
ɚ	@'	Mid central r-colored vowel	read**er**
æ	{	Near open-front unrounded vowel	tr**a**p
aɪ	aI	Diphthong	pr**i**ce
aʊ	aU	Diphthong	m**ou**th
ɑ	A	Long open-back unrounded vowel	f**a**ther
eɪ	eI	Diphthong	f**a**ce
ɝ	3'	Open mid-central unrounded r-colored vowel	n**ur**se
ɛ	E	Open mid-front unrounded vowel	dr**e**ss
i:	i	Long close front unrounded vowel	fl**ee**ce
ɪ	I	Near-close near-front unrounded vowel	k**i**t
oʊ	oU	Diphthong	g**oa**t
ɔ	O	Long open mid-back rounded vowel	th**ou**ght
ɔɪ	OI	Diphthong	ch**oi**ce
u	u	Long close-back rounded vowel	g**oo**se
ʊ	U	Near-close near-back rounded vowel	f**oo**t
ʌ	V	Open-mid-back unrounded vowel	str**u**t
Other Symbols			
'	"	primary stress	Ala**ba**ma
ˌ	%	secondary stress	**A**labama
.	.	syllable boundary	A.la.ba.ma

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. 155).

> **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. 92).

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. 94). 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 with the Amazon Polly Console (p. 75) 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. 94).
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. 153) 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. 155) request that you want to apply before the service starts synthesizing the text. For more information, see Managing Lexicons (p. 90).

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 WWW Consortium. The revised text appears as follows:

```
The WWW 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
   ```

Example 4: Python Code for PutLexicon

The following Python code example uses the AWS SDK for Python (Boto) to store a lexicon. 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. 9).

```
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))
```

Using the GetLexicon Operation

Amazon Polly provides the GetLexicon (p. 149) 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. 94).

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
    }
}
```

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. 9).

```
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.")
```

Using the ListLexicons Operations

Amazon Polly provides the ListLexicons (p. 151) 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"
        }
    ]
}
```

The following Python code example uses the AWS SDK for Python (Boto) 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. 9).

```
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))
```

Using the DeleteLexicon Operation

Amazon Polly provides the DeleteLexicon (p. 145) 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
```

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. 9).

```
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.")
```

Creating Longer Audio Files

Currently, you can't convert large documents, such as books or reports, to audio with Amazon Polly. That's because Amazon Polly has two limitations that present challenges for large text-to-speech applications:

- The maximum size of input text for the `SynthesizeSpeech` API is 1,500 billed characters.
- The maximum number of concurrent requests to the `SynthesizeSpeech` API per second is 80, with a burst limit of 100.

To create audio files for large documents, use the Polly-batch-process application. This application overcomes the challenge of processing a text document that exceeds the maximum number of characters supported by Amazon Polly. Polly-batch-processor takes a large text document and breaks it into chunks, and then uses Amazon Polly to generate an audio file for each chunk. Polly-batch-processor generates each sentence asynchronously and concurrently, reducing the time to create the audio file and overcoming any potential throttling issues. After processing the complete text file, it consolidates the chunks into a single large MP3 file. If you prefer to listen to the chunks (small) files individually, you can do so. You can also use Polly-batch-processor to create audio files for documents containing multiple short phrases that are less than 1500 characters long, such as movie titles.

Next Step

How Polly-batch-processor Works (p. 104)

How Polly-batch-processor Works

The Polly-batch-processor application uses AWS Batch to asynchronously, and concurrently, break a document into small chunks that can be easily passed through Amazon Polly.By using AWS Batch in conjunction with Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), and AWS Lambda, Polly-batch-processor enables you to quickly create an MP3 file and avoid potential throttling issues.

This is how the Polly-batch-processor application works:

1. Upload your document to an S3 bucket. After you've uploaded the document, you can optionally apply S3 object tags to add metadata, such as the title and the author's name. You can also change the voice that Amazon Polly uses or specify that you don't want to consolidate the MP3 chunks.
2. A Lambda function passes in the S3 bucket name and object key as input parameters for a new AWS Batch job in the AWS Batch queue.
3. AWS Batch downloads the document from the S3 bucket, and Polly-batch-processor breaks it into paragraph-sized chunks. It detects paragraph boundaries by searching for two line feed characters ((\n\n, which is the equivalent of two presses of the Enter key). If a paragraph is too large, the application splits the paragraph into sentences separated by the dot ('.') character.

   Note
   If a sentence contains an abbreviation (for example, St. Paul), Polly-batch-processor might split it after the period. In that case, the intonation of the resulting speech will be incorrect.
4. AWS Batch processes each text chunk in parallel,calling the Amazon Polly `SynthesizeSpeech` API for each chunk, and returning an MP3 file for each. , The number of chunks processed in parallel depends on the number of virtual CPUs (vCPUs) configured in the AWS Batch job definition. By default, AWS Batch combines the files into a single MP3 file, and uploads it to the S3 bucket where the original document is stored. If you don't want save all of the audio files in a single MP3 file, you can direct the application upload each chunk to the S3 bucket individually. To do this, apply an S3 object tag with the name `consolidated` and the value `false`.
5. AWS Batch publishes a message to an Amazon Simple Notification Service (Amazon SNS) topic that includes the URL of the MP3 file or that of the list of files if you chose not to consolidate the audio files.
6. Amazon SNS then sends you an email message that contains the URL of the MP3 file or list of files.

Next Step

Polly-batch-processor Resources (p. 105)

Polly-batch-processor Resources

To install Polly-batch-processor, you launch apair of AWS CloudFormation templates. These templates create all of the AWS resources that you need to run the application. The Polly-batch-processor application then creates three AWS Batch resources: a compute environment, a job queue, and a job definition.

Polly-batch-processor Components

Launching the AWS CloudFormation template,creates the following AWS resources:

- A number of Lambda functions, including:
  - `PollyBookSplitterFunction`, submits a new AWS Batch job for each new document whe it is uploaded to Amazon S3.
  - `CodeBuildTriggerFunction`, which triggers building a Docker image that the AWS Batch job uses to synthesize a chunk of text into speech with Amazon Polly. AWS CloudFormation calls this function as a custom resource.
  - `AWSBatchComputeEnvFunction`, which creates the AWS Batch compute environment as an AWS CloudFormation custom resource, and deletes it when it's no longer needed.
  - `AWSBatchJobQueueFunction`, which creates the AWS Batch job queue as an AWS CloudFormation custom resource, and deletes it when it's no longer needed.
  - `WSBatchJobDefinitionFunction`, which registers the AWS Batch job definition as an AWS CloudFormation custom resource, and later deregisters it.

- AWS Identity and Access Management IAM) service roles for AWS CodeBuild and AWS Batch.
- IAM roles and a security group for the EC2 instances used by AWS Batch.
- An S3 bucket configured as an Amazon S3 website, which stores the document that you want to convert to an audio file, the audio file, and a manifest file.
- An Amazon EC2 Container Registry (Amazon ECR) repository, named polly_document_processor, that stores the Docker image run by the AWS Batch jobs.
- An AWS CodeBuild project that builds the Docker image used by AWS Batch to generate the audio file. The AWS CodeBuild project pushes the built Docker image to the Amazon ECR repository named polly_document_processor.
- A custom AWS CloudFormation resource that triggers building the AWS CodeBuild project.
- An AWS Batch compute environment that launches the EC2 resources needed to run the AWS Batch jobs.
- An AWS Batch job queue where the `PollyBookSplitterFunction` Lambda function submits jobs.
- An AWS Batch job definition that specifies the Docker image and other parameters that are used to execute the AWS Batch job.

AWS Batch Resources

In the Polly-batch-processor application, AWS Batch synthesizes text chunks into speech using Amazon Polly. To do this, Polly-batch-processor creates the following AWS Batch resources as custom AWS CloudFormation resources:

- **The AWS Batch compute environment:** The AWS Batch compute environment manages the EC2 instances used to run the containerized batch jobs. It is mapped to an AWS Batch job queue. The AWS Batch scheduler takes jobs from the queue and schedules them to run on an EC2 host in the compute environment. After all of the jobs in the queue are successfully processed, AWS Batch automatically terminates the EC2 resources in your compute environment when they reach the end of the billing hour. To reduce costs, you can use Spot Instances for your EC2 resources. For more information, see Compute Environment.
- **The AWS Batch job definition:** The AWS Batch job definition specifies how the batch jobs should be run. In Polly-batch-processor, each job requires 16 vCPUs and 2048 MiB of memory. That's because each chunk is processed by a single vCPU in parallel. To prevent launching unnecessary EC2 instances in advance, we set the number of desired vCPUs to 16. To avoid having idle EC2 resources running in the compute environment when no jobs are in the queue, we configure the minimum vCPU to be 0. We also provide a security group for each EC2 instance and an IAM instance profile. For more information, see Job Definition in the *AWS Batch Developer's Guide*.
- **The AWS Batch job queue:** AWS Batch jobs are submitted to an AWS Batch job queue, where they remain until they can be scheduled to run on a compute resource. For each compute environment, you can have multiple job queues with different priorities. By default, there is only one job queue, so priority is not an issue. For more information, see Job Queue.

Next Step

Launching Polly-batch-processor (p. 106)

Launching Polly-batch-processor

AWS Batch and Amazon Polly are not available in all AWS Regions. You can use the Polly-batch-processor application only in AWS Regions where both services are available. For more information on valid Regions, see AWS Regions and Endpoints.

To use the Polly-batch-processor application, you need a virtual private cloud (VPC) with at least one public subnet and one private subnet. To set up these resources, you use two AWS CloudFormation templates:

- `PollyBatchNetwork` creates the VPC resources, including subnets, a NAT gateway, and an Internet gateway. It also provides the VPC ID and subnets as output parameters.
- `PollyBatchMaster` sets up the remaining AWS resources necessary for the application.

These templates also provide the name of the S3 bucket where the original documents that you want to convert are stored and the audio files are created.

To launch the AWS CloudFormation templates

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Choose the AWS Region.
3. Choose **Create Stack**.
4. For **Choose a Template**, choose **Specify an Amazon S3 template URL**, and type `https://s3.amazonaws.com/aws-bigdata-blog/artifacts/PollyBatch/cfn/network.yaml`.
5. Choose **Next**.
6. For **Stack Name**, type `PollyBatchNetwork`.
7. Choose **Next**, and then choose **Next** again.
8. Choose **Create**, and then wait for stack creation to complete.
9. Choose **Create Stack**.
10. For **Choose a Template**, choose **Specify an Amazon S3 template URL**, and type `https://s3.amazonaws.com/aws-bigdata-blog/artifacts/PollyBatch/cfn/master.yaml`.
11. For **Stack Name**, type `PollyBatchMaster`.
12. For **Mandatory Parameters**, specify an email address that can receive notifications when a new audio file has been created.
13. For **VPC**, choose a VPC.
14. Choose the private subnet where you want to run the AWS Batch resources.
15. Choose **Next**, and then choose **Next** again.
16. Choose **Create**.
17. When Amazon SNS sends an email to the address that you provided, follow the directions to confirm the subscription to the topic named **PollyTopic**.

Next Step

Converting a Text File to Audio (p. 107)

Converting a Text File to Audio

To convert a text file to audio, copy it to Polly-batch-processor's S3 bucket. The Lambda function begins processing the file immediately.

To prepare a document for the conversion

1. Encode it in `UTF-8` format.
2. Save it in text format (with the `.txt` extension).

To upload a file to the S3 bucket

- In the AWS CLI, type the following command.

  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 (^) and use full quotation marks (") around the input text with single quotes (') for interior tags.

```
aws s3api put-object
   --bucket bucket-name
   --key books/filename.txt
   --body filename.txt
   --tagging "voiceid=voice&author=Firstname%20Lastname&title=The%20Content%20Title"
```

  To change default settings or provide more information, you can add the following optional Amazon S3 object tags:

  - To prevent the application from consolidating the audio files into a single MP3 file, upload the document with the S3 object tag name `consolidated` and value `false`:

```
--tagging "consolidated=false"
```

    You can combine this command with other S3 object tags, such as `voiceid`, `author`, and `title`.
  - To use a voice other than the default voice for your language (for English, the default voice is Joanna), apply the `voiceid` tag and specify another voice.
  - To make it easier to manage your audio files, add S3 `Author` and `Title` tags.

  The application sends you an email with the URL for the consolidated file or the list of files.

For example, to convert the text of Oscar Wilde's story "The Happy Prince" to an audio file, type the following command in the AWS CLI. It uploads the document, tagged with the author's name and the title of the story, and uses the Brian voice.

```
aws s3api put-object
  --bucket bucket-name
  --key books/the-happy-prince.txt
  --body the-happy-prince.txt
  --tagging "voiceid=Brian&author=Oscar%20Wilde&title=The%20Happy%20Prince"
```

Next Step

Cleaning Up (p. 108)

Cleaning Up

When you're done using the application, we recommend that you delete the resources that the AWS CloudFormation templates created by deleting the `PollyBatchMaster` stack. This prevents you from accruing charges for resources that you're not using.

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 Amazon Polly: How It Works (p. 3) and follow the steps described in Getting Started with Amazon Polly (p. 6).

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 Java code samples show how to use Java-based applications to accomplish various tasks with Amazon Polly. These samples are not examples, but can be included in larger Java applications that use the AWS SDK for Java.

Code Snipppets

- SynthesizeSpeech (p. 114)

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 a 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);
        }
    }
}
```

Speech Marks

The following Java code sample show 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. 14).

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 from for inputed text.

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);
        }
    }
}
```

Example Applications

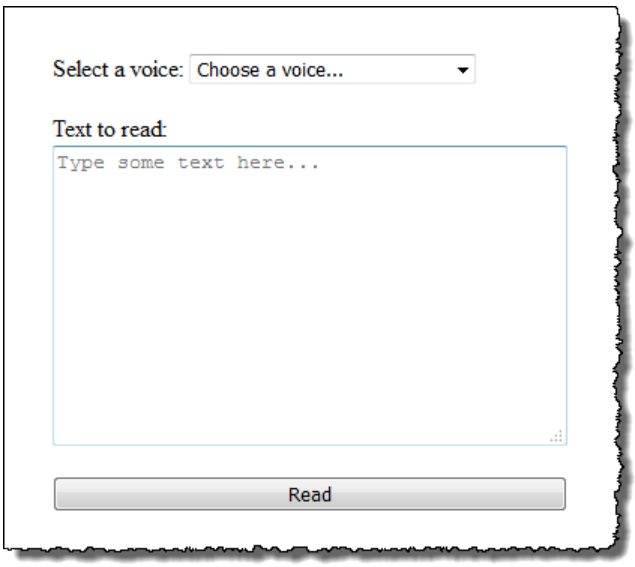
This section contains additional examples, in the form of example applications which can be used to explore Amazon Polly.

Example Applications by Programming Language

Python Example (HTML5 Client and Python Server)

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. 119).
2. Save the HTML5 client code as `index.html`. For the code, see Python Example: HTML5 User Interface (index.html) (p. 116).
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. 115).

```
<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. 115).

```
"""
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 Reference

This section provides additional reference material for the Amazon Polly documentation.

Topics

Available Voices

The following voices are available when using Amazon Polly

Name/ID	Gender
Danish (da-DK)	
Mads	Male
Naja	Female
Dutch (nl-NL)	
Lotte	Female
Ruben	Male
English (Australian) (en-AU)	
Nicole	Female
Russell	Male
English (British) (en-GB)	
Amy	Female
Brian	Male
Emma	Female
English (Indian) (en-IN)	
Raveena	Female

Name/ID	Gender
English (US) (en-US)	
Ivy	Female
Joanna	Female
Joey	Male
Justin	Male
Kendra	Female
Kimberly	Female
Salli	Female
English (Welsh) (en-GB-WLS)	
Geraint	Male
French (fr-FR)	
Céline/Celine	Female
Mathieu	Male
French (Canadian) (fr-CA)	
Chantal	Female
German (de-DE)	
Hans	Male
Marlene	Female
Vicki	Female
Icelandic (is-IS)	
Dóra/Dora	Female
Karl	Male
Italian (it-IT)	
Carla	Female
Giorgio	Male
Japanese (ja-JP)	
Mizuki	Female
Norwegian (nb-NO)	
Liv	Female
Polish (pl-PL)	
Jacek	Male

Name/ID	Gender
Jan	Male
Ewa	Female
Maja	Female
Portuguese (Brazilian) (pt-BR)	
Ricardo	Male
Vitória/Vitoria	Female
Portuguese (European) (pt-PT)	
Cristiano	Male
Inês/Ines	Female
Romanian (ro-RO)	
Carmen	Female
Russian (ru-RU)	
Maxim	Male
Tatyana	Female
Spanish (Castilian) (es-ES)	
Conchita	Female
Enrique	Male
Spanish (Latin American) (es-US)	
Miguel	Male
Penélope/Penelope	Female
Swedish (sv-SE)	
Astrid	Female
Turkish (tr-TR)	
Filiz	Female
Welsh (cy-GB)	
Gwyneth	Female

Languages Supported by Amazon Polly

Language	Language Code
Danish	da-DK

Language	Language Code
Dutch	nl-NL
English (Australian)	en-AU
English (British)	en-GB
English (Indian)	en-IN
English (US)	en-US
English (Welsh)	en-GB-WLS
French	fr-FR
French (Canadian)	fr-CA
German	de-DE
Icelandic	is-IS
Italian	it-IT
Japanese	ja-JP
Norwegian	nb-NO
Polish	pl-PL
Portuguese (Brazilian)	pt-BR
Portuguese (European)	pt-PT
Romanian	ro-RO
Russian	ru-RU
Spanish	es-ES
Spanish (Latin American)	es-US
Swedish	sv-SE
Turkish	tr-TR
Welsh	cy-GB

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 AWS Regions and Endpoints in the *Amazon Web Services General Reference*.

Throttling

- Throttle rate per account: 100 transactions (requests) 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`	80 tps with a burst limit of 100 tps

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. 90).

SynthesizeSpeech API Operation

Note the following limits related to using the SynthesizeSpeech API operation:

- The size of the input text can be up to 1500 billed characters (3000 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 5 minutes, after which, any remaining speech is cut off.

For more information, see SynthesizeSpeech (p. 155).

Note
Some limitations of the `SynthesizeSpeech` API operation can be bypassed using AWS Batch or other services. For more information on AWS Batch, see What Is AWS Batch?

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 Using SSML (p. 75).

Logging Amazon Polly API Calls with AWS CloudTrail

Amazon Polly is integrated with CloudTrail, a service that captures all of the Amazon Polly API calls and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures API calls from the Amazon Polly console or from your code to the Amazon Polly APIs. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Polly, the source IP address from which the request was made, who made the request, when it was made, and so on.

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

Amazon Polly Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to Amazon Polly actions are tracked in CloudTrail log files, where they are written with other AWS service records. CloudTrail determines when to create and write to a new file based on a time period and file size.

All Amazon Polly actions are logged by CloudTrail and are documented in the Amazon Polly API Reference (p. 144). The following actions are supported.

- DeleteLexicon (p. 145)
- DescribeVoices (p. 147)
- GetLexicon (p. 149)
- ListLexicons (p. 151)
- PutLexicon (p. 153)
- SynthesizeSpeech (p. 155)

Every log entry contains information about who generated the request. The user identity information in the log entry helps you determine the following:

- Whether the request was made with root or 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.

You can store your log files in your Amazon S3 bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted with Amazon S3 server-side encryption (SSE).

If you want to be notified upon log file delivery, you can configure CloudTrail to publish Amazon SNS notifications when new log files are delivered. For more information, see Configuring Amazon SNS Notifications for CloudTrail.

You can also aggregate Amazon Polly log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket.

For more information, see Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts.

Understanding Amazon Polly Log File Entries

CloudTrail log files can contain one or more log entries. Each entry lists multiple JSON-formatted events. A log entry 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. Log entries are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

Because of potential confidentiality issues, log entries do not contain the synthesized text. Instead, this text is redacted in the log entry.

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": "tts.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"
            }
        }

    ]
}
```

The `eventName` element identifies the action that occurred and may include date and version information, such as `"SynthesizeSpeech20161128"`, nevertheless it is still referring to the same public API.

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

Metric	Description
`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.

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

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.

Name (p. 145)

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. 151)) 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 V2

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?LanguageCode=LanguageCode&NextToken=NextToken HTTP/1.1
```

URI Request Parameters

The request requires the following URI parameters.

LanguageCode (p. 147)

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: `cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR`

NextToken (p. 147)

An opaque pagination token returned from the previous `DescribeVoices` operation. If present, this indicates where to continue the listing.

Request Body

The request does not have a request body.

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Voices": [
      {
         "Gender": "string",
         "Id": "string",
```

```
            "LanguageCode": "string",
            "LanguageName": "string",
            "Name": "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. 147)

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

Voices (p. 147)

A list of voices with their properties.

Type: Array of Voice (p. 163) 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 V2

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.

Name (p. 149)

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. 149)

Lexicon object that provides name and the string content of the lexicon.

Type: Lexicon (p. 159) object

LexiconAttributes (p. 149)

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. 160) 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. 151)) 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 V2

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. 151)

An opaque pagination token returned from previous `ListLexicons` operation. If present, indicates where to continue the list of lexicons.

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. 151)

A list of lexicon names and attributes.

Type: Array of LexiconDescription (p. 162) objects

NextToken (p. 151)

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

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 V2

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.

Name (p. 153)

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. 153)

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 V2

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

{
   "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.

LexiconNames (p. 155)

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. 155)

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

SampleRate (p. 155)

The audio frequency specified in Hz.

The valid values for `mp3` and `ogg_vorbis` are "8000", "16000", and "22050". The default value is "22050".

Valid values for `pcm` are "8000" and "16000" The default value is "16000".

Type: String

Required: No

SpeechMarkTypes (p. 155)

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. 155)

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. 155)

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. 155)

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: `Geraint | Gwyneth | Mads | Naja | Hans | Marlene | Nicole | Russell | Amy | Brian | Emma | Raveena | Ivy | Joanna | Joey | Justin | Kendra | Kimberly | Salli | Conchita | Enrique | Miguel | Penelope | Chantal | Celine | Mathieu | Dora | Karl | Carla | Giorgio | Mizuki | Liv | Lotte | Ruben | Ewa | Jacek | Jan | Maja | Ricardo | Vitoria | Cristiano | Ines | Carmen | Maxim | Tatyana | Astrid | Filiz`

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. 156)

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. 156)

Number of characters synthesized.

The response returns the following as the HTTP body.
<varlistentry> **AudioStream (p. 156)**

Stream containing the synthesized speech.
</varlistentry>

Errors

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

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. 151)) 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. The limit for input text is a maximum of 3000 characters total, of which no more than 1500 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 V2

Data Types

The following data types are supported:

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 V2

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: `cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | 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 V2

LexiconDescription

Describes the content of the lexicon.

Contents

Attributes

Provides lexicon metadata.

Type: LexiconAttributes (p. 160) 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 V2

Voice

Description of the voice.

Contents

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: `Geraint | Gwyneth | Mads | Naja | Hans | Marlene | Nicole | Russell | Amy | Brian | Emma | Raveena | Ivy | Joanna | Joey | Justin | Kendra | Kimberly | Salli | Conchita | Enrique | Miguel | Penelope | Chantal | Celine | Mathieu | Dora | Karl | Carla | Giorgio | Mizuki | Liv | Lotte | Ruben | Ewa | Jacek | Jan | Maja | Ricardo | Vitoria | Cristiano | Ines | Carmen | Maxim | Tatyana | Astrid | Filiz`

Required: No

LanguageCode

Language code of the voice.

Type: String

Valid Values: `cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | 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

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 V2

Authentication and Access Control for Amazon Polly

Access to Amazon Polly requires credentials. Those credentials must have permissions to access AWS resources, such as an Amazon Polly lexicon or an Amazon Elastic Compute Cloud (Amazon EC2) instance. The following sections provide details on how you can use AWS Identity and Access Management (IAM) and Amazon Polly to help secure access to your resources.

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. This is your *AWS account root user*. Its credentials provide complete access to all of your AWS resources.

  Important
  For security reasons, we recommend that you use the root user only to create an *administrator*, which is an *IAM user* with full permissions to your AWS account. You can then use this administrator user to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the *IAM User Guide*.

- **IAM user** – An IAM user is simply an identity within your AWS account that has specific custom permissions (for example, permissions to create a lexicon in Amazon Polly). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (CLI). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use the AWS tools, you must sign the request yourself. Amazon Polly supports *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*.

- **IAM role** – An IAM role is another IAM identity that you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:

  - **Federated user access** – Instead of creating an IAM user, you can use preexisting user 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 in your account to grant another AWS account permissions to access your account's resources. For an example, see Tutorial: Delegate Access Across AWS Accounts Using IAM Roles in the *IAM User Guide*.

  - **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. 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 running on an EC2 instance and making 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 running on the EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Amazon Polly resources. For example, you must have permissions to create an Amazon Polly lexicon.

The following sections describe how to manage permissions for Amazon Polly. We recommend that you read the overview first.

- Amazon Polly API Permissions: Actions, Permissions, and Resources Reference (p. 174)

Overview of Managing Access Permissions to Your Amazon Polly Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

> **Note**
> An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

- Amazon Polly Resources and Operations (p. 167)
- Understanding Resource Ownership (p. 167)
- Managing Access to Resources (p. 168)
- Specifying Policy Elements: Actions, Effects, and Principals (p. 169)
- Specifying Conditions in a Policy (p. 170)

Amazon Polly Resources and Operations

In Amazon Polly, the primary resource is *a lexicon*. In a policy, you use an Amazon Resource Name (ARN) to identify the resource that the policy applies to.

These resources and subresources have unique Amazon Resource Names (ARNs) associated with them, as shown in the following table.

Resource Type	ARN Format
Lexicon	`arn:aws:polly:region:account-id:lexicon/LexiconName`

Amazon Polly provides a set of operations to work with Amazon Polly resources. For a list of available operations, see Amazon Polly Amazon Polly API Reference (p. 144).

Understanding Resource Ownership

The AWS account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the AWS account of the principal entity (that is, the root account, an IAM user, or an IAM role) that authenticates the resource creation request. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create a lexicon, your AWS account is the owner of the resource (in Amazon Polly, the resource is a lexicon).
- If you create an IAM user in your AWS account and grant permissions to create a lexicon to that user, the user can create a lexicon. However, your AWS account, to which the user belongs, owns the lexicon resource.

- If you create an IAM role in your AWS account with permissions to create a lexicon, anyone who can assume the role can create a lexicon. Your AWS account, to which the user belongs, owns the lexicon resource.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

> **Note**
> This section discusses using IAM in the context of Amazon Polly. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is IAM? in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM polices) and policies attached to a resource are referred to as *resource-based* policies. Amazon Polly supports identity-based policies.

Topics

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to create a Amazon Polly resource, such as a lexicon, you can attach a permissions policy to a user or group that the user belongs to.
- **Attach a permissions policy to a role (grant cross-account permissions)** – 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. 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.

```
{
```

```
   "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/*"
      }
   ]
}
```

For more information about using identity-based policies with Amazon Polly, see Using Identity-Based Policies (IAM Policies) for Amazon Polly (p. 170). For more information about users, groups, roles, and permissions, see Identities (Users, Groups, and Roles) 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.

Specifying Policy Elements: Actions, Effects, and Principals

For each Amazon Polly resource, the service defines a set of API operations. To grant permissions for these API operations, Amazon Polly defines a set of actions that you can specify in a policy. Some API operations can require permissions for more than one action in order to perform the API operation. For more information about resources and API operations, see Amazon Polly Resources and Operations (p. 167) and Amazon Polly API Reference (p. 144).

The following are the most basic policy elements:

- **Resource** – You use an Amazon Resource Name (ARN) to identify the resource that the Identity-based policy applies to. For more information, see Amazon Polly Resources and Operations (p. 167).
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, you can use `polly:PutLexicon` to add a lexicon to the region.
- **Effect** – You specify the effect, either allow or deny, when the user requests the specific action. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). Amazon Polly doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

For a list showing all of the Amazon Polly API operations and the resources that they apply to, see Amazon Polly API Permissions: Actions, Permissions, and Resources Reference (p. 174).

Specifying Conditions in a Policy

When you grant permissions, you can use the access policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see Condition in the *IAM User Guide*.

To express conditions, you use predefined condition keys. There are no condition keys specific to Amazon Polly. However, there are AWS-wide condition keys that you can use as appropriate. For a complete list of AWS-wide keys, see Available Keys for Conditions in the *IAM User Guide*.

Using Identity-Based Policies (IAM Policies) for Amazon Polly

This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) and thereby grant permissions to perform operations on Amazon Polly resources.

> **Important**
> We recommend that you first review the introductory topics that explain the basic concepts and options available to manage access to your Amazon Polly resources. For more information, see Overview of Managing Access Permissions to Your Amazon Polly Resources (p. 167).

Topics

The following shows an example of a permissions policy.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "AllowGet-Delete-ListActions",
      "Effect": "Allow",
      "Action": [
         "polly:GetLexicon",
         "polly:DeleteLexicon",
         "polly:ListLexicons"],
      "Resource": "*"
      }
   ],
    "Statement": [{
      "Sid": "NoOverrideMyLexicons",
      "Effect": "Deny",
      "Action": [
         "polly:PutLexicon"],
      "Resource": "arn:aws:polly:us-east-2:123456789012:lexicon/my*"
      }
   ]
}
```

The policy has two statements:

- The first statement grants permission for three Polly actions (`polly:GetLexicon`, `polly:DeleteLexicon`, and `polly:ListLexicons` on any lexicon. Use of the wildcard character (*) as the resource grants universal permissions for these actions across all regions and lexicons owned by this account.
- The second statement explicitly denies permission for one Polly action (`polly:PutLexicon`). The ARN shown as the resource specifically applies this permission all lexicons that begin with the letters "my" that are in the region `us-east-2`.

For a table showing all of the Amazon Polly API operations and the resources that they apply to, see Amazon Polly API Permissions: Actions, Permissions, and Resources Reference (p. 174).

Permissions Required to Use the Amazon Polly Console

For a user to work with the Amazon Polly console, that user must have a minimum set of permissions that allows users 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 won'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, you need to grant permissions to all the Amazon Polly APIs. There are no additional permissions needed. The following permissions policy is all that is needed to use the Amazon Polly console.

```
}
"Version": "2012-10-17",
   "Statement": [{
      "Sid": "Console-AllowAllPollyActions",
      "Effect": "Allow",
      "Action": [
         "polly:*"],
      "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, you need to grant permissions to all the Amazon Polly APIs. This is discussed in Permissions Required to Use the Amazon Polly Console (p. 171).

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. 6)) you create an administrator user to manage your account, including creating users and managing their permissions.

You might choose to 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 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 Access Control (p. 166) and writing 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. 145)

Required Permissions (API Action): `polly:DeleteLexicon`

Resources: `arn:aws:polly:`*`region`*`:`*`account-id`*`:lexicon/`*`LexiconName`*

API Operation: DescribeVoices (p. 147)

Required Permissions (API Action): `polly:DescribeVoices`

Resources: `arn:aws:polly:`*`region`*`:`*`account-id`*`:lexicon/`*`voice-name`*

API Operation: GetLexicon (p. 149)

Required Permissions (API Action): `polly:GetLexicon`

Resources: `arn:aws:polly:`*`region`*`:`*`account-id`*`:lexicon/`*`voice-name`*

API Operation: ListLexicons (p. 151)

Required Permissions (API Action): `polly:ListLexicons`

Resources: `arn:aws:polly:region:account-id:lexicon/*`

API Operation: PutLexicon (p. 153)

Required Permissions (API Action): `polly:ListLexicons`

Resources: `*`

API Operation: SynthesizeSpeech (p. 155)

Required Permissions (API Action): `polly:SynthesizeSpeech`

Resources: `*`

Document History for Amazon Polly

The following table describes the documentation for this release of Amazon Polly.

- **Latest documentation update:** April 19, 2017

Change	Description	Date
New service and guide	This is the initial release of the AWS Text-to-Speech service, Amazon Polly, and the *Amazon Polly Developer Guide*.	November 30, 2016
New feature and expanded documentation	This is an update to Amazon Polly, and includes the new Speech Marks feature as well as an expansion of SSML capabilities.	April 19, 2017

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.