



开发人员指南

# Amazon Polly



# Amazon Polly: 开发人员指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

# Table of Contents

什么是 Amazon Polly ? .....	1
工作原理 .....	1
优势 .....	2
您是 新用户吗 ? .....	2
与 AWS SDKs .....	3
入门 .....	4
报名参加 AWS .....	4
注册获取 AWS 账户 .....	4
设置 AWS CLI .....	5
重新配置 AWS CLI .....	6
语音合成示例 .....	8
双向流媒体 .....	12
SynthesizeSpeech 并 StartSpeechSynthesisStream 进行了比较 .....	12
发送文本和接收音频 .....	13
打开直播 .....	14
发送短信 .....	14
接收音频 .....	15
关闭直播 .....	15
代码示例 .....	15
Amazon Polly 中的语音 .....	25
可用语音 .....	25
品牌语音 .....	33
双语语音 .....	33
重音双语语音 .....	33
完全双语语音 .....	34
应用新闻播音员的语音 .....	35
收听语音 .....	37
对语音速度进行计时 .....	37
更改语音速度 .....	38
Amazon Polly 中的语言 .....	40
阿拉伯语 (arb) .....	44
阿拉伯语 (海湾) (ar-AE) .....	47
加泰罗尼亚语 (ca-ES) .....	51
中文 (粤语) (yue-CN) .....	54

中文 ( 普通话 ) (cmn-CN) .....	58
捷克语 ( cs-CZ ) .....	61
丹麦语 (da-DK) .....	64
荷兰语 ( 比利时 ) (nl-BE) .....	67
荷兰语 (nl-NL) .....	69
英语 ( 美国 ) (en-US) .....	72
英语 ( 澳大利亚 ) (en-AU) .....	75
英语 ( 英国 ) (en-GB) .....	77
英语 ( 印度 ) (en-IN) .....	80
英语 ( 爱尔兰 ) (en-IE) .....	83
英语 ( 新西兰 ) (en-NZ) .....	86
英语 ( 新加坡 ) ( en-SG ) .....	91
英语 ( 南非 ) (en-ZA) .....	93
英语 ( 威尔士 ) (en-GB-WLS) .....	97
芬兰语 (fi-FI) .....	100
法语 (fr-FR) .....	103
法语 ( 比利时 ) (fr-BE) .....	106
法语 ( 加拿大 ) (fr-CA) .....	108
德语 (de-DE) .....	110
德语 ( 奥地利 ) (de-AT) .....	113
德语 ( 瑞士标准 ) ( de-CH ) .....	116
印地语 (hi-IN) .....	119
冰岛语 (is-IS) .....	122
意大利语 (it-IT) .....	125
日语 (ja-JP) .....	127
韩语 (ko-KR) .....	130
挪威语 (nb-NO) .....	132
波兰语 (pl-PL) .....	134
葡萄牙语 (pt-PT) .....	137
葡萄牙语 ( 巴西 ) (pt-BR) .....	139
罗马尼亚语 (ro-RO) .....	141
俄语 (ru-RU) .....	143
西班牙语 (es-ES) .....	146
西班牙语 ( 墨西哥 ) (es-MX) .....	148
西班牙语 ( 美国 ) (es-US) .....	150
瑞典语 (sv-SE) .....	152

土耳其语 (tr-TR) .....	155
威尔士语 (cy-GB) .....	158
语音引擎 .....	162
生成式引擎 .....	162
可用的生成式语音 .....	163
特征和区域兼容性 .....	165
Long-form 发动机 .....	166
可用的长篇语音 .....	167
特征和区域兼容性 .....	167
神经引擎 .....	168
可用的神经语音 .....	169
特征和区域兼容性 .....	172
标准引擎 .....	173
可用的标准语音 .....	173
特征和区域兼容性 .....	177
选择语音引擎 .....	178
语音标记 .....	179
语音标记类型 .....	179
语音视位和 Amazon Polly .....	180
语音标记输出 .....	180
请求语音标记 .....	181
没有 SSML 的语音标记示例 .....	183
有 SSML 的语音标记示例 .....	184
使用 SSML .....	186
预留字符 .....	187
在控制台上使用 SSML .....	189
通过合成语音命令使用 SSML .....	190
合成 SSML 增强文档 .....	192
支持的 SSML 标签 .....	193
标识 SSML 增强文本 .....	194
添加停顿 .....	195
强调词语 .....	196
为特定词语指定另一种语言 .....	196
在您的文本中插入自定义标签 .....	198
在段落之间添加停顿 .....	198
使用语音发音 .....	199

控制音量、语速和音高 .....	200
为合成语音设置最长持续时间 .....	203
在句子之间添加停顿 .....	206
控制如何朗读特殊的词语类型 .....	207
首字母缩略词和缩写的发音 .....	210
指定词性以改善发音 .....	210
添加呼吸音 .....	212
新闻播音员风格 .....	216
添加动态范围压缩 .....	216
柔和地朗读 .....	219
控制音质 .....	219
轻读 .....	221
管理词典 .....	222
使用多个词典 .....	223
上传词典 .....	224
应用词典 ( 合成语音 ) .....	229
在控制台上筛选词典列表 .....	232
在控制台上下载词典 .....	234
删除词典 .....	235
长音频文件 .....	237
为异步合成设置 IAM 策略 .....	238
创建长音频文件 .....	238
配额 .....	242
支持的区域 .....	242
配额和节流率 .....	243
并发请求 .....	244
减少节流的最佳实践 .....	244
发音词典 .....	244
SynthesizeSpeech API 操作 .....	245
SpeechSynthesisTask API 操作 .....	245
语音合成标记语言 (SSML) .....	246
示例代码和应用程序 .....	247
Java 示例 .....	247
DeleteLexicon .....	248
DescribeVoices .....	250
GetLexicon .....	251

ListLexicons .....	253
PutLexicon .....	254
StartSpeechSynthesisTask .....	256
语音标记 .....	260
SynthesizeSpeech .....	263
Python 示例 .....	264
DeleteLexicon .....	265
GetLexicon .....	266
ListLexicon .....	267
PutLexicon .....	268
StartSpeechSynthesisTask .....	269
SynthesizeSpeech .....	270
Java 示例 .....	270
Python 示例 .....	275
Python 示例 : index.html .....	277
Python 示例 : server.py .....	281
iOS 示例 .....	288
Android 示例 .....	290
代码示例 .....	293
基本功能 .....	293
操作 .....	294
场景 .....	344
将文本转换为语音以及将语音转换回文本 .....	344
创建口型同步应用程序 .....	345
创建用于分析客户反馈的应用程序 .....	346
Amazon Polly 入门 .....	352
安全性 .....	359
数据保护 .....	359
静态加密 .....	360
传输中加密 .....	360
互连网络流量保密性 .....	360
身份和访问管理 .....	360
受众 .....	361
使用身份进行身份验证 .....	361
使用策略管理访问 .....	362
Amazon Polly 如何与 IAM 配合使用 .....	363

Identity-based 策略示例 .....	369
Amazon Polly API 权限参考 .....	375
问题排查 .....	376
日志记录和监控 .....	378
合规性验证 .....	378
恢复能力 .....	379
基础设施安全性 .....	379
安全最佳实践 .....	379
使用接口 VPC 终端节点 .....	380
可用性 .....	380
为 Amazon Polly 创建 VPC 终端节点 .....	381
测试您的 VPC 和 Amazon Polly 之间的连接 .....	381
控制对 Amazon Polly 终端节点的访问 .....	381
对 VPC 上下文键的支持 .....	382
使用 AWS CloudTrail 记录 Amazon Polly API 调用 .....	383
CloudTrail 中的 Amazon Polly 信息 .....	383
示例：Amazon Polly 日志文件条目 .....	384
CloudWatch 集成 .....	386
获取 CloudWatch 指标（控制台） .....	386
在 AWS CLI 上获取 CloudWatch 指标 .....	386
Amazon Polly 指标 .....	387
Amazon Polly 指标的维度 .....	388
API 参考 .....	389
操作 .....	389
DeleteLexicon .....	390
DescribeVoices .....	392
GetLexicon .....	396
GetSpeechSynthesisTask .....	399
ListLexicons .....	402
ListSpeechSynthesisTasks .....	405
PutLexicon .....	408
StartSpeechSynthesisStream .....	411
StartSpeechSynthesisTask .....	417
SynthesizeSpeech .....	425
数据类型 .....	431
AudioEvent .....	432

---

CloseStreamEvent .....	433
FlushStreamConfiguration .....	434
Lexicon .....	435
LexiconAttributes .....	436
LexiconDescription .....	438
StartSpeechSynthesisStreamActionStream .....	439
StartSpeechSynthesisStreamEventStream .....	440
StreamClosedEvent .....	442
SynthesisTask .....	443
TextEvent .....	448
ThrottlingReason .....	449
ValidationExceptionField .....	450
Voice .....	451
常见错误类型 .....	453
常见参数 .....	455
文档历史记录 .....	458
.....	cdlxxii

# 什么是 Amazon Polly ?

Amazon Polly 云服务可以将文本转化为逼真的语音。可以使用 Amazon Polly 开发能提高参与度和可用性的应用程序。Amazon Polly 支持多种语言，并包含各种逼真的语音。借助 Amazon Polly，您可以构建支持语音并能用于很多位置的应用程序，并使用适合客户的语音。此外，您只需为合成的文本付费。您也可以免费缓存和重放 Amazon Polly 生成的语音。

Amazon Polly 提供许多语音选项，包括生成式、长式、神经和标准 text-to-speech (TTS) 选项。这些声音使用新的机器学习技术突破性地提高了语音质量，从而尽可能提供最自然、最像人的 text-to-speech 声音。神经 TTS 技术还支持新闻播音员风格，专为新闻播报使用案例量身定制。

Amazon Polly 的常用案例包括但不限于移动应用程序（如新闻阅读器、游戏、电子学习平台）、视障人士辅助功能应用程序以及快速增长的物联网（IoT）细分市场。

Amazon Polly 经认证可用于 1996 年《健康保险可携性与责任法 (HIPAA)》和支付卡行业数据安全标准 (PCI DSS) 的受监管工作负载。

## Amazon Polly 工作原理

Amazon Polly 可以将输入文本转化为逼真的语音。要使用一种 Amazon Polly 语音，请选择[语音引擎](#)，调用语音合成方法，提供要合成的文本，然后指定音频输出格式。然后，Amazon Polly 将提供的合成文本合成为高质量语音音频流。

- 输入文本 — 提供要合成的文本，然后 Amazon Polly 返回音频流。您可以提供纯文本格式或语音合成标记语言 (SSML) 格式的输入。借助 SSML，您可以控制语音的各个方面，如发音、音量、音高和语速。有关更多信息，请参阅[由 SSML 文档生成语音](#)。
- 可用语音 — Amazon Polly 提供了语言和各种语音的组合，包括双语语音（适用于英语和印地语）。对于大多数语言，您可以从多种男性和女性语音中选择。在启动语音合成任务时，您可以指定语音 ID，然后 Amazon Polly 将使用此语音将文本转换为语音。Amazon Polly 不是翻译服务，即，合成的语音采用文本的语言。数位形式的数字（例如，53 而不是五十三）采用语音的语言而不是文本合成。有关更多信息，请参阅[Amazon Polly 中的语音](#)。
- 输出格式 — Amazon Polly 可以提供多种格式的合成语音。您可以选择适合您需求的音频格式。例如，您可以请求 MP3 或 Ogg Vorbis 格式的语音以供 Web 和移动应用程序使用。或者，您可以请求 PCM 输出格式以供 AWS IoT 设备和电话解决方案使用。对于电话应用程序，您可以使用 mu-law 或 a-law 格式。

**Note**

要在您的浏览器中收听 Amazon Polly 语音示例，请参阅 [Amazon Polly 产品概述](#)。

## 优势

使用 Amazon Polly 的一些好处包括：

- **高品质** — Amazon Polly 提供高性能的生成式、长型、神经和高质量 (TTS) 的声音。text-to-speech 这些技术可以合成发音精度非常高的自然语音（包括缩写、首字母缩略词扩展、日期/时间解释和同义词消歧）。
- **低延迟** – Amazon Polly 可实现快速响应，这使其能适用于低延迟使用案例（如对话系统）。
- **支持很多种语言和语音** – Amazon Polly 支持数十种语音和语言，并为大多数语言提供男性和女性语音选项。随着我们推出的神经语音越来越多，这个数字将继续增加。美国英语语音 Matthew 和 Joanna 也可以使用神经新闻播音员风格，与专业新闻主播的发音十分相似。
- **经济实惠** — Amazon Polly 的 pay-per-use 模型意味着没有安装成本。可以从小规模开始，然后根据应用程序的发展情况进行扩展。
- **基于云的解决方案** — 设备上的 TTS 解决方案需要大量的计算资源，特别是 CPU 功率、RAM 和磁盘空间。这些可能增加平板电脑、智能手机等设备的开发成本和功耗。相比之下，在中完成的 TTS 转换 AWS Cloud 大大降低了本地资源需求。这样就能够以一流的质量支持所有可用的语言和语音。此外，还可以立即向所有最终用户提供语音改进，而不需要进行额外的设备更新。

**Note**

要在您的浏览器中收听 Amazon Polly 语音示例，请参阅 [Amazon Polly 产品概述](#)。

## 您是新用户吗？

如果您是 Amazon Polly 服务的新用户，我们建议您按列出的顺序阅读以下各节：

1. [Amazon Polly 工作原理](#) – 本节介绍了您可用来创造简单体验的各种 Amazon Polly 输入和选项。
2. [Amazon Polly 入门](#) — 在本部分中，您将设置账户并测试 Amazon Polly 语音合成。
3. [Amazon Polly 的示例代码和应用程序](#) — 本节提供了可供您探索 Amazon Polly 的更多示例。

## 将 Amazon Polly 与软件开发工具包配合使用 AWS

AWS 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档	代码示例
<a href="#">适用于 C++ 的 AWS SDK</a>	<a href="#">适用于 C++ 的 AWS SDK 代码示例</a>
<a href="#">AWS CLI</a>	<a href="#">AWS CLI 代码示例</a>
<a href="#">适用于 Go 的 AWS SDK</a>	<a href="#">适用于 Go 的 AWS SDK 代码示例</a>
<a href="#">适用于 Java 的 AWS SDK</a>	<a href="#">适用于 Java 的 AWS SDK 代码示例</a>
<a href="#">适用于 JavaScript 的 AWS SDK</a>	<a href="#">适用于 JavaScript 的 AWS SDK 代码示例</a>
<a href="#">适用于 Kotlin 的 AWS SDK</a>	<a href="#">适用于 Kotlin 的 AWS SDK 代码示例</a>
<a href="#">适用于 .NET 的 AWS SDK</a>	<a href="#">适用于 .NET 的 AWS SDK 代码示例</a>
<a href="#">适用于 PHP 的 AWS SDK</a>	<a href="#">适用于 PHP 的 AWS SDK 代码示例</a>
<a href="#">AWS Tools for PowerShell</a>	<a href="#">AWS Tools for PowerShell 代码示例</a>
<a href="#">适用于 Python (Boto3) 的 AWS SDK</a>	<a href="#">适用于 Python (Boto3) 的 AWS SDK 代码示例</a>
<a href="#">适用于 Ruby 的 AWS SDK</a>	<a href="#">适用于 Ruby 的 AWS SDK 代码示例</a>
<a href="#">适用于 Rust 的 AWS SDK</a>	<a href="#">适用于 Rust 的 AWS SDK 代码示例</a>
<a href="#">适用于 SAP ABAP 的 AWS SDK</a>	<a href="#">适用于 SAP ABAP 的 AWS SDK 代码示例</a>
<a href="#">适用于 Swift 的 AWS SDK</a>	<a href="#">适用于 Swift 的 AWS SDK 代码示例</a>

### 示例可用性

找不到所需的内容？通过使用此页面底部的提供反馈链接请求代码示例。

# Amazon Polly 入门

Amazon Polly 提供了多个 API 操作，您可以轻松地将这些操作与现有应用程序集成。有关支持的操作的列表，请参阅 [操作](#)。

可以在 Amazon Polly 控制台和 AWS CLI 上执行几乎所有相同的操作。但是，您无法在上收听合成语音。AWS CLI 要在 AWS CLI 上处理音频，请将您的文本保存到文件中。然后，在您选择的音频应用程序中打开该文件。

您可以使用以下任一选项。

- AWS 软件开发工具包 — 使用软件开发工具包时，系统会自动使用您提供的凭证对您向 Amazon Polly 提出的请求进行签名和身份验证。这是用于构建应用程序的推荐选择。
- AWS CLI — 无需编写任何代码 AWS CLI 即可使用 Amazon Polly。

首次使用 Amazon Polly 之前，您必须先注册。当您注册 Amazon Web Services (AWS) 时，您的 AWS 账户将自动注册使用中的所有服务 AWS，包括 Amazon Polly。您只需为使用的服务和资源付费。如果您是新的 AWS 客户，则可以免费开始使用 Amazon Polly。有关更多信息，请参阅 [AWS 免费使用套餐](#)。

下面几个部分描述了如何开始使用 Amazon Polly。

## 主题

- [报名参加 AWS](#)
- [设置 AWS CLI](#)
- [重新配置 AWS CLI](#)

## 报名参加 AWS

在使用任何 AWS 服务（包括 Amazon Polly）之前，您必须先注册。AWS

## 注册获取 AWS 账户

首先 AWS，你需要一个 AWS 账户。有关创建的信息 AWS 账户，请参阅《AWS 账户管理 参考指南》AWS 账户中的 [入门](#) 指南。

有关 IAM 的更多信息，请参阅以下文档：

- [AWS Identity and Access Management \(IAM\)](#)
- [IAM 入门](#)
- [IAM 用户指南](#)

#### Note

记下您的 AWS 账户 ID。您在后续步骤中需要使用此值。

## 设置 AWS CLI

请按照以下步骤下载并配置 AWS CLI 以与 Amazon Polly 配合使用。

要设置 AWS Command Line Interface

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：
  - [开始设置 AWS Command Line Interface](#)
  - [配置 AWS Command Line Interface](#)
2. 在 Config 文件中为管理员用户添加已命名的 AWS CLI AWS 配置文件。您可以在运行 AWS CLI 命令时使用此配置文件。有关命名配置文件的更多信息，请参阅 AWS Command Line Interface 用户指南中的[命名配置文件](#)。

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

有关可用 AWS 区域以及 Amazon Polly 支持的区域列表，请参阅中的[区域和终端节点](#)。Amazon Web Services 一般参考

#### Note

如果您使用的是您在配置时指定的 Amazon Polly 支持的区域 AWS CLI，请从代码示例中省略以下行。AWS CLI

```
--region aws-region
```

3. 在命令提示符处键入以下帮助命令来验证设置。

```
aws help
```

AWS CLI 窗口中应显示有效 AWS 命令的列表。

## 重新配置 AWS CLI

如果您之前下载并配置过，则除非您重新配置 AWS CLI，否则 Amazon Polly 可能不可用。AWS CLI 以下过程检查是否需要重新进行配置。

要从中重新激活 Amazon Polly AWS CLI

1. 在命令提示符下键入以下帮助命令，以验证 Amazon Polly 的 AWS CLI 可用性。

```
aws polly help
```

如果您看到 Amazon Polly 的描述并且 AWS CLI 窗口中显示了有效命令列表，则可以立即使用 Amazon Polly。AWS CLI 在这种情况下，您可以跳过此过程的剩余部分。如果未显示，请继续执行步骤 2。

2. 使用以下两个选项之一激活 Amazon Polly：

- a. 卸载并重新安装。AWS CLI

有关说明，请参阅 AWS Command Line Interface 用户指南中的[安装 AWS Command Line Interface](#)。

或者

- b. 下载文件 [service-2.json](#)。

在命令提示符下，运行以下命令。

```
aws configure add-model --service-model file://service-2.json --service-name polly
```

### 3. 重新验证 Amazon Polly 的可用性。

```
aws polly help
```

Amazon Polly 的描述应可见。

# 使用 Amazon Polly 合成语音示例

本页提供了在控制台、和 Python 中执行的 AWS CLI 简短语音合成示例。本示例使用纯文本执行语音合成，而不是使用 SSML。

## Console

在控制台上合成语音

1. 登录 AWS 管理控制台 并打开 Amazon Polly 控制台，网址为。<https://console.aws.amazon.com/polly/>
2. 选择文本到语音转换选项卡。文本字段将加载示例文本，以便您可以快速试用 Amazon Polly。
3. 关闭 SSML。
4. 在输入框中键入或粘贴此文本。

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

5. 在引擎下，选择生成式、长篇、神经或标准。
6. 选择一种语言和 AWS 地区，然后选择一种声音。（如果您为引擎选择神经，则只有支持 NTTs 的语言和语音才可用。所有标准和长篇语音都会被禁用。）
7. 要立即收听语音，请选择收听。
8. 要将语音保存到文件中，请执行以下操作之一：
  - a. 选择下载。
  - b. 要更改为其他文件格式，请展开“其他设置”，打开“语音文件格式设置”，选择所需的文件格式（例如 OGG、PCM MP3、mu-law 或 a-law），然后选择“下载”。

## AWS CLI

在本练习中，将通过传送输入文本来调用 SynthesizeSpeech 操作。您可以将生成的音频保存为文件并验证其内容。

1. 运行 synthesize-speech AWS CLI 命令将示例文本合成到音频文件 (hello.mp3)。

以下 AWS CLI 示例是针对 Unix、Linux 和 macOS 进行格式化的。对于 Windows，请将每行末尾的反斜杠 (\) Unix 行继续符替换为脱字号 (^) 并在输入文本周围使用全角引号 (""), 内部标签使用单引号 (')。

```
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
```

在对 `synthesize-speech` 的调用中，您提供样本文本，该文本将与您选择的语音进行合成。您必须提供语音 ID（将在后续步骤中进行说明）和输出格式。该命令会将生成的音频保存为 `hello.mp3` 文件。除 MP3 文件外，该操作还向控制台发送以下输出。

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

2. 播放生成的 `hello.mp3` 文件以验证合成的语音。

## Python

要测试 Python 示例代码，您需要 AWS SDK for Python (Boto)。有关说明，请参阅 [适用于 Python \(Boto3\) 的 AWS SDK](#)。

本示例中的 Python 代码将执行以下操作：

- 调用，AWS SDK for Python (Boto) 向 Amazon Polly 发送 `SynthesizeSpeech` 请求（通过提供一些文本作为输入）。
- 访问在响应中生成的音频流并将音频保存为您本地磁盘上的文件 (`speech.mp3`)。
- 使用您的本地系统的默认音频播放器播放音频文件。

将代码保存为一个文件 (`example.py`) 并运行。

```
"""Getting Started Example for Python 2.7+/3.3+"""  
from boto3 import Session
```

```
from botocore.exceptions import BotoCoreError, ClientError
from contextlib import closing
import os
import sys
import subprocess
from tempfile import gettempdir

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

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

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

        try:
            # Open a file for writing the output as a binary stream
            with open(output, "wb") as file:
                file.write(stream.read())
        except IOError as error:
            # Could not write to file, exit gracefully
            print(error)
            sys.exit(-1)
else:
    # The response didn't contain audio data, exit gracefully
    print("Could not stream audio")
    sys.exit(-1)
```

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

有关更深入的示例，请参阅以下主题：

- [在控制台上使用 SSML](#)
- [应用词典 \(合成语音\)](#)
- [Amazon Polly 的示例代码和应用程序](#)

# 使用双向流媒体合成语音

Amazon Polly 提供的 `StartSpeechSynthesisStream` 操作通过您的应用程序和服务之间的双向通信建立 HTTP/2 连接。文本从您的应用程序流向 Amazon Polly，而合成音频则流回原处。您可以在文本可用时发送文本，Amazon Polly 会在合成时返回音频，而无需任何一方等待另一方完成。

当文本是逐步生成而不是一次全部生成时，这很有用。例如，由 Amazon Bedrock 基础模型提供支持的客户服务聊天机器人通过令牌生成其响应令牌。通过双向流式传输，您的应用程序可以在模型生成每个文本块时将其转发到 Amazon Polly，并在模型仍在生成其余响应时开始向调用者播放音频。

此操作需要生成引擎和支持 HTTP/2 事件流的 AWS SDK。音频以一系列块的形式到达，您的应用程序会将其累积到完整的音频输出中。此操作不支持语音标记。

## Note

不支持 AWS CLI ( v1 和 v2 )、PowerShell ( v4 和 v5 ) AWS 工具、Python 和 .NET v3。你可以将双向直播 API 与以下 SDK 一起使用：AWS 适用于 Java 的 SDK 2.x、v JavaScript 3、.NET v4、C++、Go v2、Kotlin、PHP v3、Rust 和 Swift。

## 主题

- [SynthesizeSpeech 并 StartSpeechSynthesisStream 进行了比较](#)
- [发送文本和接收音频](#)
- [代码示例](#)

## SynthesizeSpeech 并 StartSpeechSynthesisStream 进行了比较

[SynthesizeSpeech](#) 是一个请求-响应操作。您可以在单个请求中提供完整的文本，并在单个响应中接收完整的合成音频。它支持所有引擎（标准、神经、长格式、生成式）、包括语音标记在内的所有输出格式，并且每个请求的文本限制为 6,000 个总字符（其中计费字符不能超过 3,000 个）。一旦第一个字节可用，响应就会将音频流回去。当您预先准备好所有文本时，请使用此操作。

[StartSpeechSynthesisStream](#) 是一种双向流媒体操作。它会打开一个 HTTP/2 连接，您可以通过该连接增量发送文本，并在合成时接收音频。由于文本是连续流式传输的，因此没有每个请求的文本限制。它需要生成引擎，并且不支持语音标记。当文本以增量方式到达，并且您希望在所有输入都可用之前开始音频输出时，请使用此操作。常见场景包括：

- 对话式 AI 和语音助手。大型语言模型以小块 ( 标记 ) 的形式生成响应文本。当每个文本块到达时将其转发到 Amazon Polly，这样用户就可以在模型仍在生成时听到语音。
- Real-time 翻译。翻译系统逐段生成翻译后的文本。流式传输每个片段进行合成，无需等待完整翻译完成。
- Long-form 内容超出 SynthesizeSpeech 限制。长度超过 6,000 个字符的文本可以连续流式传输，而无需拆分为多个请求或管理区块边界。

## SynthesizeSpeech 和的比较 StartSpeechSynthesisStream

方面	SynthesizeSpeech	StartSpeechSynthesisStream
协议	Request-response	双向事件流 () HTTP/2
文字传送	请求正文中的全文	通过 TextEvent 消息流式传输输入文本
音频传输	通过 HTTP 响应正文流式传输音频响应	通过 AudioEvent 消息流式传输音频响应
发动机支架	标准、神经、长形、生成式	仅限生成式
SSML 支持	是 ( 所有引擎 ; <a href="#">支持的标签因引擎而异</a> )	是 ( <a href="#">仅限生成引擎标签</a> )
词典	支持	是
语音标记	是	否
文字限制	每个请求总共有 6,000 个字符 ( 计费 3,000 个 )	每个 6,000 个字符 ( 计费 3,000 个 ) TextEvent
AWS CLI 支持	是	否 ( 双向直播需要 SDK )

## 发送文本和接收音频

双向流媒体会话包括打开连接、同时发送文本和接收音频，然后在输入完成后关闭直播。以下各节详细描述了每个阶段。

## 打开直播

您的应用程序通过 SDK 调用该[StartSpeechSynthesisStream](#)操作，指定合成参数 ( EngineVoiceId、OutputFormat、以及可选的LanguageCodeLexiconNames、SampleRate )。SDK 建立 HTTP/2 连接，双向流已准备好接受输入事件。

## 发送短信

客户端在输入流上发送一条或多条[TextEvent](#)消息。每个事件都可以在文本可用后立即发送，无需等待完整输入准备就绪。文本事件不需要与句子或标点符号边界对齐。Amazon Polly 会在内部重新组装文本，并生成听起来很自然的语音，无论输入如何按事件分配。

### Note

使用 [SSML](#) 时，每个 SSML 文档都必须独立包含在单个文档中。TextEvent 您不能在多个事件中拆分 SSML 标签。但是，您可以在同一个流中混合使用纯文本事件和 SSML 事件。

直播有以下时间限制：

- 最长直播时长：10 分钟。无论活动如何，Amazon Polly 都会在 10 分钟后关闭直播。如果您的内容需要更多时间，请为剩余的文本打开一个新的视频流。
- 连续事件之间的空闲超时：5 秒。如果在 5 秒钟内没有发送任何输入事件，Amazon Polly 将关闭直播。如果您的文本源暂停时间超过 5 秒，请发送TextEvent带有空字符串或空白字符串的 keep-alive 以防止超时。

## 使用刷新强制合成缓冲文本

默认情况下，Amazon Polly 会根据自然语言边界决定何时合成缓冲文本。这样可以产生最佳的音频质量，但这意味着在您发送后可能不会立即返回音频TextEvent。

Flushing 可以让你控制何时发生合成。刷新时，Amazon Polly 会立即合成到目前为止缓冲过的所有文本，无论文本是否以自然边界结尾。当您的文本源在逻辑部分之间暂停并且您想为到目前为止发送的内容提供音频时，这很有用。

要冲洗，请设置[FlushStreamConfiguration](#)。Force a true 上的参数为TextEvent。你也可以发送一个设置了 flush 标志TextEvent的空白，这样就可以在不添加新内容的情况下触发合成。

冲洗是一种权衡。设置Force为true句子中间可能会影响发音和语调，因为合成器缺乏关于后面内容的上下文。为了获得最佳效果，请尽可能让 Amazon Polly 缓冲到自然边界，并且仅在延迟要求需要时才强制合成。

## 接收音频

当 Amazon Polly 合成文本时，它会在输出流上返回[AudioEvent](#)消息。每个事件都包含一块音频数据。您的应用程序必须累积这些块（例如，将它们按顺序写入文件或音频缓冲区），才能生成完整的音频输出。音频事件可以在您仍在发送文本事件时到达。

## 关闭直播

当所有输入文本都发送完毕后，客户端会发送[CloseStreamEvent](#)。Amazon Polly 完成对所有剩余缓冲文本的处理，发送最终的音频事件，并返回[StreamClosedEvent](#)包含合成字符总数的 a。请务必发送，CloseStreamEvent而不是依靠刷新来结束直播。关闭可确保合成并返回所有缓冲文本。

有关请求参数、事件类型和错误的完整详细信息，请参阅 [StartSpeechSynthesisStreamAPI 参考](#)。

## 代码示例

以下示例演示了一个完整的双向流媒体会话。该程序创建异步 Amazon Polly 客户端，打开配置有生成引擎和 MP3 输出的流，以一系列消息TextEvent的形式发送文本，将AudioEvent返回的区块累积到输出文件中，然后使用关闭流。CloseStreamEvent由于输入和输出是同时进行的，因此音频数据在发送所有文本之前就开始到达。

### Java

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyAsyncClient;
import software.amazon.awssdk.services.polly.model.*;

import java.io.FileOutputStream;
import java.io.OutputStream;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.Semaphore;
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscription;

public class BidirectionalStreamExample {

    public static void main(String[] args) throws Exception {
```

```

try (PollyAsyncClient pollyClient = PollyAsyncClient.builder()
    .region(Region.US_EAST_1)
    .build()) {

    StartSpeechSynthesisStreamRequest request =
StartSpeechSynthesisStreamRequest.builder()
    .engine(Engine.GENERATIVE)
    .voiceId(VoiceId.TIFFANY)
    .outputFormat(OutputFormat.MP3)
    .sampleRate("24000")
    .build();

    try (OutputStream audioOutput = new FileOutputStream("output.mp3")) {

        StartSpeechSynthesisStreamResponseHandler responseHandler =
            StartSpeechSynthesisStreamResponseHandler.builder()

                .subscriber(StartSpeechSynthesisStreamResponseHandler.Visitor.builder()
                    .onAudioEvent(audioEvent -> {
                        try {
                            byte[] audio =
audioEvent.audioChunk().asByteArray();
                            System.out.println("Received AudioEvent: " +
audio.length + " bytes");

                            audioOutput.write(audio);
                        } catch (Exception e) {
                            throw new RuntimeException(e);
                        }
                    })
                    .onStreamClosedEvent(closedEvent -> {
                        System.out.println("Stream closed. Characters
synthesized: "
                            + closedEvent.requestCharacters());
                    })
                    .onDefault(event -> {})
                    .build())
                .onError(error -> {
                    System.err.println("Stream error: " +
error.getMessage());
                })
                .build();

        String[] textChunks = {
            "The weather forecast for today shows clear skies ",

```

```

        "with temperatures reaching twenty five degrees. ",
        "Tomorrow we expect some cloud cover in the morning ",
        "but it should clear up by the afternoon. ",
        "The rest of the week looks mostly sunny ",
        "with a slight chance of rain on Friday. ",
        "Overall a great week to spend time outdoors."
    };

    Publisher<StartSpeechSynthesisStreamActionStream> inputPublisher =
subscriber -> {
    subscriber.onSubscribe(new Subscription() {
        private final Semaphore permits = new Semaphore(0);
        private volatile boolean cancelled = false;

        @Override
        public void request(long n) {
            permits.release((int) Math.min(n, Integer.MAX_VALUE));
        }

        @Override
        public void cancel() {
            cancelled = true;
            permits.release();
        }

        {
            new Thread(() -> {
                for (String chunk : textChunks) {
                    try { permits.acquire(); } catch (InterruptedException
e) { return; }

                    if (cancelled) return;
                    System.out.println("Sending TextEvent: " +
chunk.trim());

                    subscriber.onNext(StartSpeechSynthesisStreamActionStream.textEventBuilder()
                        .text(chunk).textType(TextType.TEXT).build());
                    // Simulate delay between chunks (e.g. waiting for LLM
tokens)
                    try { Thread.sleep(300); } catch (InterruptedException
e) { return; }
                }
                if (!cancelled) {
                    subscriber.onNext(StartSpeechSynthesisStreamActionStream
                        .closeStreamEventBuilder().build());

```

```

        subscriber.onComplete();
    }
    }).start();
}
});
};

CompletableFuture<Void> future = pollyClient.startSpeechSynthesisStream(
    request, inputPublisher, responseHandler);

future.join();
} // audioOutput closed
} // pollyClient closed
}
}

```

## JavaScript

```

import { PollyClient, StartSpeechSynthesisStreamCommand } from "@aws-sdk/client-polly";
import { createWriteStream } from "fs";

const client = new PollyClient({ region: "us-east-1" });

const textChunks = [
    "The weather forecast for today shows clear skies ",
    "with temperatures reaching twenty five degrees. ",
    "Tomorrow we expect some cloud cover in the morning ",
    "but it should clear up by the afternoon. ",
    "The rest of the week looks mostly sunny ",
    "with a slight chance of rain on Friday. ",
    "Overall a great week to spend time outdoors."
];

const sleep = (ms) => new Promise((resolve) => setTimeout(resolve, ms));

async function* createInputEvents() {
    for (const chunk of textChunks) {
        console.log(`Sending TextEvent: ${chunk.trim()}`);
        yield {
            TextEvent: {
                Text: chunk,
                TextType: "text",
            }
        };
    }
}

```

```
    },
  };
  // Simulate delay between chunks (e.g. waiting for LLM tokens)
  await sleep(300);
}

yield { CloseStreamEvent: {} };
}

async function synthesizeStream() {
  const command = new StartSpeechSynthesisStreamCommand({
    Engine: "generative",
    VoiceId: "Tiffany",
    OutputFormat: "mp3",
    SampleRate: "24000",
    ActionStream: createInputEvents(),
  });

  const response = await client.send(command);
  const outputStream = createWriteStream("output.mp3");

  for await (const event of response.EventStream) {
    if (event.AudioEvent) {
      console.log(`Received AudioEvent: ${event.AudioEvent.AudioChunk.length}
bytes`);
      outputStream.write(event.AudioEvent.AudioChunk);
    } else if (event.StreamClosedEvent) {
      console.log(
        `Stream closed. Characters synthesized:
${event.StreamClosedEvent.RequestCharacters}`
      );
    }
  }

  outputStream.end();
}

synthesizeStream().catch(console.error);
```

Go

```
package main
```

```
import (
    "context"
    "fmt"
    "os"
    "strings"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/polly"
    "github.com/aws/aws-sdk-go-v2/service/polly/types"
)

func main() {
    ctx := context.Background()

    cfg, err := config.LoadDefaultConfig(ctx, config.WithRegion("us-east-1"))
    if err != nil {
        fmt.Fprintf(os.Stderr, "Failed to load config: %v\n", err)
        os.Exit(1)
    }

    client := polly.NewFromConfig(cfg)

    output, err := client.StartSpeechSynthesisStream(ctx,
        &polly.StartSpeechSynthesisStreamInput{
            Engine:          types.EngineGenerative,
            VoiceId:        types.VoiceIdTiffany,
            OutputFormat:  types.OutputFormatMp3,
            SampleRate:    aws.String("24000"),
        })
    if err != nil {
        fmt.Fprintf(os.Stderr, "Failed to start stream: %v\n", err)
        os.Exit(1)
    }

    stream := output.GetStream()
    defer stream.Close()

    audioFile, err := os.Create("output.mp3")
    if err != nil {
        fmt.Fprintf(os.Stderr, "Failed to create output file: %v\n", err)
        os.Exit(1)
    }
}
```

```
defer audioFile.Close()

textChunks := []string{
    "The weather forecast for today shows clear skies ",
    "with temperatures reaching twenty five degrees. ",
    "Tomorrow we expect some cloud cover in the morning ",
    "but it should clear up by the afternoon. ",
    "The rest of the week looks mostly sunny ",
    "with a slight chance of rain on Friday. ",
    "Overall a great week to spend time outdoors.",
}

// Send text events in a goroutine
go func() {
    for _, chunk := range textChunks {
        fmt.Printf("Sending TextEvent: %s\n", strings.TrimSpace(chunk))
        err := stream.Send(ctx,
&types.StartSpeechSynthesisStreamActionStreamMemberTextEvent{
            Value: types.TextEvent{
                Text:      aws.String(chunk),
                TextType: types.TextTypeText,
            },
        })
        if err != nil {
            fmt.Fprintf(os.Stderr, "Failed to send text event: %v\n", err)
            return
        }
        // Simulate delay between chunks (e.g. waiting for LLM tokens)
        time.Sleep(300 * time.Millisecond)
    }

    // Signal end of input
    stream.Send(ctx,
&types.StartSpeechSynthesisStreamActionStreamMemberCloseStreamEvent{
        Value: types.CloseStreamEvent{}},
    )
}()

// Receive audio events
for event := range stream.Events() {
    switch v := event.(type) {
    case *types.StartSpeechSynthesisStreamEventStreamMemberAudioEvent:
        fmt.Printf("Received AudioEvent: %d bytes\n", len(v.Value.AudioChunk))
        audioFile.Write(v.Value.AudioChunk)
    }
}
```

```

    case *types.StartSpeechSynthesisStreamEventStreamMemberStreamClosedEvent:
        fmt.Printf("Stream closed. Characters synthesized: %d\n",
            v.Value.RequestCharacters)
    }
}

if err := stream.Err(); err != nil {
    fmt.Fprintf(os.Stderr, "Stream error: %v\n", err)
    os.Exit(1)
}
}
}

```

## Rust

```

use aws_sdk_polly::types::{
    CloseStreamEvent, Engine, OutputFormat,
    StartSpeechSynthesisStreamActionStream,
    StartSpeechSynthesisStreamEventStream, TextEvent, TextType, VoiceId,
};
use aws_sdk_polly::Client;
use std::fs::File;
use std::io::Write;
use tokio::time::{sleep, Duration};

#[tokio::main]
async fn main() {
    let config = aws_config::defaults(aws_config::BehaviorVersion::latest())
        .region("us-east-1")
        .load()
        .await;

    let client = Client::new(&config);

    let text_chunks = vec![
        "The weather forecast for today shows clear skies ",
        "with temperatures reaching twenty five degrees. ",
        "Tomorrow we expect some cloud cover in the morning ",
        "but it should clear up by the afternoon. ",
        "The rest of the week looks mostly sunny ",
        "with a slight chance of rain on Friday. ",
        "Overall a great week to spend time outdoors.",
    ];
}

```

```

let input_stream = async_stream::stream! {
    for chunk in &text_chunks {
        println!("Sending TextEvent: {}", chunk.trim());
        yield Ok(StartSpeechSynthesisStreamActionStream::TextEvent(
TextEvent::builder().text(*chunk).text_type(TextType::Text).build().unwrap(),
        ));
        // Simulate delay between chunks (e.g. waiting for LLM tokens)
        sleep(Duration::from_millis(300)).await;
    }
    yield Ok(StartSpeechSynthesisStreamActionStream::CloseStreamEvent(
        CloseStreamEvent::builder().build(),
    ));
};

let mut output = client
    .start_speech_synthesis_stream()
    .engine(Engine::Generative)
    .voice_id(VoiceId::Tiffany)
    .output_format(OutputFormat::Mp3)
    .sample_rate("24000")
    .action_stream(input_stream.into())
    .send()
    .await
    .expect("Failed to start stream");

let mut audio_file = File::create("output.mp3").expect("Failed to create output
file");

while let Ok(Some(event)) = output.event_stream.recv().await {
    match event {
        StartSpeechSynthesisStreamEventStream::AudioEvent(audio_event) => {
            if let Some(chunk) = audio_event.audio_chunk() {
                let bytes = chunk.as_ref();
                println!("Received AudioEvent: {} bytes", bytes.len());
                audio_file.write_all(bytes).unwrap();
            }
        }
        StartSpeechSynthesisStreamEventStream::StreamClosedEvent(closed_event)
=> {
            println!(
                "Stream closed. Characters synthesized: {}",
                closed_event.request_characters()
            );
        }
    }
}

```

```
    }  
    _ => {}  
  }  
}
```

# Amazon Polly 中的语音

Amazon Polly 提供了数十种逼真的语音并支持多种语言。每个语音都是用母语人士创造的，因此，语音与语音之间存在变化，甚至在同一种语言中也是如此。您还可以使用 AWS 管理控制台用您选择的文本测试每个语音。对于大多数语言，至少有一个男性语音和一个女性语音，通常每个语音都不止一个。少数语言只有一个语音。

语音目录和包含的语言的数量正在不断更新以包括更多选择。要建议一种新的语言或语音，请在此页上提供反馈。遗憾的是，我们无法在特定新语言计划发布之前对其发表评论。

## Note

要在您的浏览器中收听 Amazon Polly 语音示例，请参阅 [Amazon Polly 产品概述](#)。

## 主题

- [可用语音](#)
- [双语语音](#)
- [应用新闻播音员的语音](#)
- [收听语音](#)
- [对语音速度进行计时](#)
- [更改语音速度](#)

## 可用语音

Amazon Polly 以多种语言提供各种逼真的语音，以使用文本合成语音。下表显示了 Amazon Polly 提供的所有语音。

	语言和语言变体	语言代码	名称/ID	性别	生成式语音	长篇语音	神经语音	标准语音
1	阿拉伯语	arb	Zeina	女	否	否	否	是

	语言和语言变体	语言代码	名称/ID	性别	生成式语音	长篇语音	神经语音	标准语音
2	阿拉伯语 (海湾)	ar-AE	Hala*	女	否	否	是	否
			Zayd*	男	否	否	是	否
3	荷兰语 (比利时)	nl-BE	Lisa	女	是	否	是	否
4	加泰罗尼亚语	ca-ES	Arlet	女	否	否	是	否
5	捷克语	cs-CZ	Jitka	女	否	否	是	否
6	中文 (粤语)	yue-CN	Hiujin	女	否	否	是	否
7	中文 (普通话)	cmn-CN	知语	女	否	否	是	是
8	丹麦语	da-DK	Naja	女	否	否	否	是
			Mads	男	否	否	否	是
			Sofie	女	否	否	是	否
9	荷兰语	nl-NL	Laura	女	是	否	是	否
			Lotte	女	否	否	否	是
			Ruben	男	否	否	否	是

	语言和语言变体	语言代码	名称/ID	性别	生成式语音	长篇语音	神经语音	标准语音
10	英语 (澳大利亚)	en-AU	Nicole	女	否	否	否	是
			Olivia	女	是	否	是	否
			Russell	男	否	否	否	是
11	英语 (英国)	en-GB	Amy**	女	是	否	是	是
			Emma	女	否	否	是	是
			Brian	男	是	否	是	是
			Arthur	男	否	否	是	否
12	英语 (印度)	en-IN	Aditi*	女	否	否	否	是
			Raveena	女	否	否	否	是
			Kajal*	女	是	否	是	否
13	英语 (爱尔兰)	en-IE	Niamh	女	是	否	是	否
14	英语 (新西兰)	en-NZ	Aria	女	是	否	是	否
15	英语 (新加坡)	en-SG	Jasmine	女	是	否	是	否
16	英语 (南非)	en-ZA	Ayanda	女	是	否	是	否

	语言和语言变体	语言代码	名称/ID	性别	生成式语音	长篇语音	神经语音	标准语音
17	英语 (美国)	en-US	Danielle	女	是	是	是	否
			Gregory	男	否	是	是	否
			Ivy	女 ( 孩童 )	否	否	是	是
			Joanna**	女	是	否	是	是
			Kendra	女	否	否	是	是
			Kimberly	女	否	否	是	是
			Salli	女	是	否	是	是
			Joey	女	否	否	是	是
			Justin	男	否	否	是	否
			Kevin	男 ( 孩童 )	否	否	是	是
			Matthew**	男 ( 孩童 )	是	否	是	否
			Ruth	女	是	是	是	否
			Stephen	男	是	否	是	否
			Tiffany	女	是	否	否	否
			Patrick	男	否	是	否	否
			女					
			男					
18	英语 ( 威尔士 )	en-GB-WLS	Geraint	男	否	否	否	是

	语言和语言变体	语言代码	名称/ID	性别	生成式语音	长篇语音	神经语音	标准语音
19	芬兰语	fi-FI	Suvi	女	否	否	是	否
20	法语	fr-FR	Ambre	女	支持	否	否	否
			Céline/ Celine	女	是	否	否	否
			弗洛里安	男	是	否	否	否
			Florian	女	是	否	是	是
			Léa	男	否	否	否	是
			Mathieu	男	是	否	是	否
21	法语 (比利时)	fr-BE	Isabelle	女	是	否	是	否
22	法语 (加拿大)	fr-CA	Chantal	女	否	否	否	是
			Gabrielle	女	是	否	是	否
			Liam	男	是	否	是	否
23	德语	de-DE	Marlene	女	否	否	否	是
			Vicki	女	是	否	是	是
			Hans	男	否	否	否	是
			Daniel	男	是	否	是	否
			伦纳特	男	是	否	否	否

	语言和语言变体	语言代码	名称/ID	性别	生成式语音	长篇语音	神经语音	标准语音
24	德语 (奥地利)	de-AT	Hannah	女	是	否	是	否
25	德语 (瑞士)	de-CH	Sabrina	女	是	否	是	否
26	印地语	hi-IN	Aditi*	女	否	否	否	是
			Kajal*	女	否	否	是	否
27	冰岛语	is-IS	Dóra/ Dora	女	否	否	否	是
				男	否	否	否	是
			Karl					
28	意大利语	it-IT	比阿特丽斯	女	是	否	否	否
				女	否	否	否	是
			Carla	女	是	否	是	是
			Bianca	男	是	否	否	否
			Lorenzo	男	否	否	否	是
			Giorgio	男	否	否	是	否
			Adriano					
29	日语	ja-JP	Mizuki	女	否	否	否	是
			Takumi	男	否	否	是	是
			Kazuha	女	否	否	是	否
			Tomoko	女	否	否	是	否

	语言和语言变体	语言代码	名称/ID	性别	生成式语音	长篇语音	神经语音	标准语音
30	韩语	ko-KR	Seoyeon	女	是	否	是	是
			Jihye	女	否	否	是	否
31	挪威语	nb-NO	Liv	女	否	否	否	是
			Ida	女	否	否	是	否
32	波兰语	pl-PL	Ewa	女	是	否	否	是
			Maja	女	否	否	否	是
			Jacek	男	否	否	否	是
			Jan	男	否	否	否	是
			Ola	女	是	否	是	否
33	葡萄牙语 (巴西)	pt-BR	Camila	女	是	否	是	是
			Vitória/ Vitoria	女	否	否	是	是
				男	否	否	否	是
			Ricardo	男	否	否	是	否
			Thiago					
34	葡萄牙语 (欧洲)	pt-PT	Inês/ Ines	女	否	否	是	是
				男	否	否	否	是
			Cristiano					
35	罗马尼亚语	ro-RO	Carmen	女	否	否	否	是
36	俄语	ru-RU	Tatyana	女	否	否	否	是
			Maxim	男	否	否	否	是

	语言和语言变体	语言代码	名称/ID	性别	生成式语音	长篇语音	神经语音	标准语音
37	西班牙语 (西班牙)	es-ES	Conchita	女	否	否	否	是
			Lucia	女	是	否	是	是
			Alba	女	否	是	否	否
			Enrique	男	否	否	否	是
			Sergio	男	是	否	是	否
			Raúl	男	否	是	否	否
38	西班牙语 (墨西哥)	es-MX	Mia	女	支持	否	是	是
			Andrés	男	是	否	是	否
39	西班牙语 (美国)	es-US	Lupe**	女	是	否	是	是
			Penélope/	女	否	否	否	是
			Penelope	男	否	否	否	是
			Miguel	男	是	否	是	否
40	瑞典语	sv-SE	Astrid	女	否	否	否	是
			Elin	女	否	否	是	否
41	土耳其语	tr-TR	Filiz	女	否	否	否	是
			Burcu	女	否	否	是	否
42	威尔士语	cy-GB	Gwyneth	女	否	否	否	是

\* 该语音说的是双语。有关更多信息，请参阅 [双语语音](#)。

\*\* 这些语音在采用神经格式时，可实现新闻播音员风格。有关更多信息，请参阅 [应用新闻播音员的语音](#)。

每个 Amazon Polly 语音引擎都具备独特的功能。详细了解 Amazon Polly 提供的语音引擎的功能和区域可用性：

- [生成式语音](#)
- [长篇语音](#)
- [神经语音](#)
- [标准语音](#)

## 品牌语音

除了上表中列出的可用语音外，您还可以使用 Amazon Polly 为自己的品牌形象打造自定义语音。借助品牌语音，您可以为客户提供独特而独有的语音。要了解有关 Amazon Polly 品牌语音的更多信息，请参阅 [品牌语音](#)。

## 双语语音

Amazon Polly 有两种生成双语语音的方式：

- [重音双语语音](#)
- [完全双语语音](#)

## 重音双语语音

重音双语语音可以使用任意 Amazon Polly 语音创建，但只有在使用 SSML 标签时才能创建。

通常，输入文本中的所有词语都使用您正在使用的指定语音的默认语言朗读。

例如，如果您正在使用 Joanna 语音（说美国英语），Amazon Polly 会使用 Joanna 的语音朗读以下内容，没有法语腔调：

```
<speak>
  Why didn't she just say, 'Je ne parle pas français?'
</speak>
```

在这种情况下，Je ne parle pas français 单词就会像英语一样朗读。

不过，如果您使用 Joanna 语音时加入 <lang> 标签，Amazon Polly 会使用 Joanna 语音以美国口音的法语朗读这个句子：

```
<speaK>
  Why didn't she just say, <lang xml:lang="fr-FR">'Je ne parle pas français?'</
lang>.
</speaK>
```

因为 Joanna 的母语不是法语，发音会以她的母语为基础，也就是美国英语。例如，虽然完美的法语发音在 français 这个词中有一个小舌颤音 /R/，但 Joanna 的美国英语语音将这个音素发为相应的 /r/。

如果您使用说意大利语的 Giorgio 语音朗读以下文本，Amazon Polly 会以 Giorgio 的语音通过意大利语发音朗读这个句子：

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

## 完全双语语音

Aditi 或 Kajal ( 印度英语和印地语 ) 之类的完全双语语音可以流畅地说两种语言。这使您能够在使用同一语音的单个文本中使用来自这两种语言的单词和短语。

目前，Aditi、Kajal、Hala 和 Zayd是唯一可用的完全双语语音。

使用双语语音 ( 例如 Aditi )

Aditi 可以流畅地说印度英语 (en-IN) 和印地语 (hi-IN)。您可以同时采用英语和印地语合成语音，而且语音甚至可以在同一句子中在两种语言之间切换。

Hindi 可以采用两种不同的形式：

- Devanagari : “उसेन कहाँ, खेल तोह अब शुरु होगा”
- Romanagari ( 使用拉丁字母 ) : “Usne kahan, khel toh ab shuru hoga”

此外，还可以在单个语句中混合使用英语和/或印地语：

- Devanagari + 英语 : “This is the song कभी कभी अदिति”

- Romanagari + 英语：“This is the song from the movie Jaane Tu Ya Jaane Na.”
- Devanagari + Romanagari + 英语：“This is the song कभी कभी अदिति from the movie Jaane Tu Ya Jaane Na.”

由于 Aditi 是双语语音，因此将正确读出所有这些情况下的文本，因为 Amazon Polly 可以区分语言和脚本。

Amazon Polly 还支持同时采用英语（阿拉伯数字）和印地语（Devanagari 数字）的数字、日期、时间和货币扩展名。默认情况下，阿拉伯数字用印度英语读出。要使 Amazon Polly 采用印地语读出它们，您必须使用 hi-IN 语言代码参数。

## 应用新闻播音员的语音

人们会根据上下文使用不同的讲话风格。例如，非正式对话听起来与电视或电台新闻广播有很大不同。由于制作标准语音的方式，这些语音不能产生不同的说话风格。但是，神经语音却可以。神经语音会针对特定的讲话风格进行训练，体现这种风格固有的某些部分语音的语调变化和重音。

除了默认的神经语音外，Amazon Polly 还提供了一种新闻播音员风格。新闻播音员风格使用神经系统来生成电视或电台新闻播音员风格的语音。新闻播音员风格适用于 Matthew 和 Joanna 语音，这些语音提供美国英语（en-US）版；适用于 Lupe 语音，该语音提供美国西班牙语（es-US）版；还适用于 Amy 语音，该语音提供英式英语（en-GB）版。

要使用新闻播音员风格，请先选择神经引擎，然后在输入文本中使用以下步骤中描述的语法。

### Note

- 要使用任何神经网络语言风格，您需要使用支持神经语音的 AWS 区域之一。此选项并非在所有区域中都可用。有关更多信息，请参阅 [特征和区域兼容性](#)。

## Console

### 应用新闻播音员风格

1. 通过以下网址打开 Amazon Polly 控制台：<https://console.aws.amazon.com/polly/>。
2. 请确保您使用的是支持神经语音的 AWS 区域。
3. 在文本到语音转换页面上，对于引擎，选择神经。

4. 选择您想要使用的语言和语音。只有美国英语 ( en-US ) 版的 Matthew 和 Joanna 语音、美国西班牙语 ( es-US ) 版的 Lupe 语音以及英式英语 ( en-GB ) 版的 Amy 语音可用于新闻播音员语音。
5. 打开 SSML。
6. 使用新闻播音员风格 SSML 语法将输入文本添加到文本转语音请求中。

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

例如，您可以按以下所示使用新闻播音员语音标签：

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

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

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

7. 选择收听。

## AWS CLI

### 应用新闻播音员风格

1. 在您的 API 请求中，包含具有 `neural` 值的引擎参数：

```
--engine neural
```

2. 使用新闻播音员风格 SSML 语法将输入文本添加到 API 请求中。

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

例如，您可以按以下所示使用新闻播音员语音标签：

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

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

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

有关 SSML 的更多信息，请参阅[支持的 SSML 标签](#)。

## 收听语音

[设置](#) Amazon Polly 后，您就可以在控制台上使用自定义文本来测试语音。

在控制台上收听 Amazon Polly 语音

1. 登录到 AWS 管理控制台 并打开 Amazon Polly 控制台，网址：<https://console.aws.amazon.com/polly/>。
2. 选择文本到语音转换选项卡。
3. 对于引擎，选择生成式、长篇、神经或标准。
4. 选择语言和区域。然后选择语音。
5. 为要说出语音输入文本或使用默认短语，然后选择收听。

## 对语音速度进行计时

由于语音之间的自然差异，每个可用的语音会以略微不同的速度朗读文本。例如，对于美国英语语音，Ivy 和 Joanna 的语速比 Matthew 的语速略快一点，比 Joey 的语速快很多。由于语音之间有很大

的差异，因此，对于 Amazon Polly 语音来说没有任何标准语速（每分钟单词数）。不过，您可以使用[语音标记](#)来查明用您的语音朗读选定文本需要多长时间。

对口语文本段落长度进行计时

1. 打开 AWS CLI。
2. 运行以下代码（根据需要填充）。

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

3. 打开 LengthOfText.txt。

如果文本为“Mary had a little lamb”，则 Amazon Polly 返回的最后几行将为：

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

最后一个 viseme（本质上是“lamb”的最后几个字母的读音）在语音开始后 1082 毫秒开始。虽然这不是音频的准确长度，但很接近，可用作比较语音的基础。

## 更改语音速度

对于某些应用程序，您可能会发现，您更希望放慢或加快您喜欢的语音。如果需要考虑语音速度，Amazon Polly 会提供使用 SSML 标签进行修改的能力。例如，如果您的组织正在开发一款向移民受众朗读书籍内容的应用程序，则可能需要调整语音速度。这些受众会说英语，但不那么流利。Amazon Polly 可使用 SSML `<prosody>` 标签来帮助您放慢语音速度。

您可以使用百分比：

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

```
</speak>
```

或者使用预设速度：

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

在将 SSML 与 Amazon Polly 结合使用时，您可使用两个速度选项：

- 预设速度：x-slow、slow、medium、fast 和 x-fast。在这些情况下，每个选项的速度都是近似的，具体取决于您的首选语音。medium 选项是正常的语音速度。
- 语音速度的 n%：可使用介于 20% 和 200% 之间的任何语音速度的百分比。在这些情况下，您可以选择自己所需的速度。不过，实际的语音速度都是近似的，具体取决于您选定的语音。100% 被视为正常的语音速度。

#### Note

按照不同的速度测试您选择的语音。每个选项的速度都是近似值，并且取决于您选定的语音。

有关使用 prosody 标签的更多信息，请参阅[控制音量、语速和音高](#)。

# Amazon Polly 中的语言

以下语言受 Amazon Polly 支持，可用于合成语音。每种语言都具有唯一的语言代码。这些语言代码为 [W3C 语言标识标签](#)（语言名称代码为 *ISO 639-3*，国家/地区代码为 *ISO 3166*）。

请从下表中选择一种语言，详细了解 Amazon Polly 提供的音素和语音视位。

语言	语言代码
<a href="#">*阿拉伯语</a>	arb
<a href="#">阿拉伯语 (海湾)</a>	ar-AE
<a href="#">加泰罗尼亚语</a>	ca-ES
<a href="#">中文 (粤语)</a>	yue-CN
<a href="#">中文 (普通话)</a>	cmn-CN
<a href="#">捷克语</a>	cs-CZ
<a href="#">丹麦语</a>	da-DK
<a href="#">荷兰语</a>	nl-BE

语言	语言代码	
<a href="#">(比利时)</a>		
<a href="#">荷兰语</a>	nl-NL	
<a href="#">英语 (澳大利亚)</a>	en-AU	
<a href="#">英语 (英国)</a>	en-GB	
<a href="#">英语 (印度)</a>	en-IN	
<a href="#">英语 (新西兰)</a>	en-NZ	
<a href="#">英语 (新加坡)</a>	en-SG	
<a href="#">英语 (南非)</a>	en-ZA	

语言	语言代码
<a href="#">英语 (美国)</a>	en-US
<a href="#">英语 (威尔士)</a>	en-GB-WLS
<a href="#">芬兰语</a>	fi-FI
<a href="#">法语</a>	fr-FR
<a href="#">法语 (比利时)</a>	fr-BE
<a href="#">法语 (加拿大)</a>	fr-CA
<a href="#">印地语</a>	hi-IN
<a href="#">德语*</a>	de-DE
<a href="#">德语 (奥地利)</a>	de-AT

语言	语言代码
<a href="#">德语 (瑞士标准)</a>	de-CH
<a href="#">冰岛语</a>	is-IS
<a href="#">意大利语</a>	it-IT
<a href="#">日语*</a>	ja-JP
<a href="#">韩语*</a>	ko-KR
<a href="#">挪威语</a>	nb-NO
<a href="#">波兰语</a>	pl-PL
<a href="#">葡萄牙语 (巴西)</a>	pt-BR
<a href="#">葡萄牙语 (欧洲)</a>	pt-PT
<a href="#">罗马尼亚语</a>	ro-RO

语言	语言代码
<a href="#">俄语</a>	ru- RU
<a href="#">西班牙语 (西班牙)</a>	es- ES
<a href="#">西班牙语 (墨西哥)</a>	es- MX
<a href="#">西班牙语 (美国)</a>	es- US
<a href="#">瑞典语</a>	sv- SE
<a href="#">土耳其语</a>	tr-TR
<a href="#">威尔士英语</a>	cy- GB

## 阿拉伯语 (arb)

下表列出了 Amazon Polly 支持的 Zeina 阿拉伯语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

## 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
ʔ	ʔ	声门闭锁音	أنا	
ʕ	ʔ\	浊咽擦音	عُمَرَ	k
b	b	浊双唇塞音	بَلَد	p
d	d	浊齿龈塞音	داري	t
d <sup>ɤ</sup>	d_ʔ\	强调浊齿龈塞音	ضَوْء	t
ɖʒ	dʒ	浊龈后塞擦音	جَمِيل	S
ð	D	浊齿擦音	ذَلِكَ	T
ð <sup>ɤ</sup>	D_ʔ\	强调浊齿擦音	ظَالِم	T
f	f	清唇齿擦音	فَصْل	f
g	g	浊软顎塞音	إنجلترا	k
ɣ	G	浊软顎擦音	عَرَب	k
h	h	清喉擦音	هذا	k
j	j	硬顎近音	يَمَشِي	i
k	k	清软顎塞音	كَالْب	k
l	l	齿龈边音	لأقوى	t
l <sup>ɤ</sup>	l_G	强调齿龈边音	عبدالله	t
m	m	双唇鼻音	ماذا	p
n	n	齿龈鼻音	نور	t
p	p	清双唇塞音	حَبَس	p

IPA	X-SAMPA	描述	示例	语音视位
q	q	清软喉塞音	قَرِيْب	k
r	r	齿龈颤音	رَمَل	r
s	s	清齿龈摩擦音	سُؤَال	s
s <sup>f</sup>	s_?\ 	强调清齿龈摩擦音	صَاحِب	s
ʃ	S	清龈后擦音	شُكْر	S
t	t	清齿龈塞音	تَمَر	t
t <sup>f</sup>	t_?\ 	强调清齿龈塞音	طَالِب	t
θ	T	清齿擦音	ثَلَاث	T
v	v	浊唇齿擦音	فِيْتَامِيْن	f
w	w	浊圆唇软顎近音	وَلَد	u
x	x	清软顎擦音	خَوْف	k
ħ	X\ 	清咽擦音	حَوْلَ	k
z	z	浊齿龈擦音	زُهْر	s
元音				
a	a	开前不圆唇元音	بَرْد	a
a:	a:	长开前不圆唇元音	دَار	a
a <sup>f</sup>	A_?\ 	强调开后不圆唇元音	طَابَل	a
a <sup>f</sup> :	A_?\ :	强调长开后不圆唇元音	طَالِم	a
u	u	闭后圆唇元音	شُرْب	u

IPA	X-SAMPA	描述	示例	语音视位
u:	u:	长闭后圆唇元音	سور	u
u <sup>ɥ</sup>	u_?\ <sup>ɥ</sup>	强调闭后圆唇元音	بُدّ	u
u <sup>ɥ</sup> :	u_?\ <sup>ɥ</sup> :	强调长闭后圆唇元音	طول	u
i	i	闭前不圆唇元音	بنت	i
i:	i:	长闭前不圆唇元音	حَزِين	i
i <sup>ɥ</sup>	i_?\ <sup>ɥ</sup>	强调闭前不圆唇元音	ضدّ	i
i <sup>ɥ</sup> :	i_?\ <sup>ɥ</sup> :	强调长闭前不圆唇元音	ماضي	i
e	e	半闭前不圆唇元音	ماركت	e
e:	e:	长半闭前不圆唇元音	موديل	e
ɔ	O	半开后圆唇元音	تكنولوجيا	O
ɔ:	O:	长半开后圆唇元音	تلفزيون	O

## 阿拉伯语 ( 海湾 ) (ar-AE)

下表列出了 Amazon Polly 支持的 Hala 阿拉伯语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	发音	语音视位
辅音					
b	b	浊双唇塞音	بلا	/" b a . l a d /	b

IPA	X-SAMPA	描述	示例	发音	语音视位
d	d	浊齿龈塞音	رد	/" r a d d /	d
d <sup>ɸ</sup>	d_? <sup>\</sup>	咽化浊齿龈塞音	ضوء	/" d_? <sup>\</sup> a w ? /	D
f	f	清唇齿擦音	فرن	/" f l . r l n /	f
g	g	浊软顎塞音	قال	/" g a : l /	k
j	j	浊硬顎近音	يَمشي	/" j l m . S i : /	i
k	k	清软顎塞音	كامل	/" k a : . m i l /	k
l	l	浊齿龈边音	ليل	/" l e : l /	t
l <sup>ɸ</sup>	l_G	咽化浊齿龈边音	عبدالله	/? <sup>\</sup> a b . " d A_? <sup>\</sup> l_G . l_G A_? <sup>\</sup> /	t
m	m	双唇鼻塞音	مئة	/" m l j . j a /	p
n	n	齿龈鼻塞音	نور	/" n u : r /	t
p	p	清双唇塞音	أوبرا	/" ? O . p e . r a : /	p
q	q	清软喉塞音	قصر	/" q A_? <sup>\</sup> s_? <sup>\</sup> r /	k
r	r	齿龈颤音	رمل	/" r a . m l l /	r
s	s	清齿龈摩擦音	سمسم	/" s l m . s l m /	s
s <sup>ɸ</sup>	s_? <sup>\</sup>	咽化清齿龈摩擦音	صاحب	/" s_? <sup>\</sup> A_? <sup>\</sup> : . X l l b /	s
t	t	清齿龈塞音	تمر	/" t a . m a r /	t

IPA	X-SAMPA	描述	示例	发音	语音视位
tʃ	t_ʔ\	咽化清齿龈摩擦音	طالب	/" t_ʔ\ A_ʔ: . l l b /	t
v	v	浊唇齿擦音	فيتامين	/v i: . t A . " m i: n /	f
w	w	浊唇软腭近音	وايد	/" w a: . j l d /	u
x	x	清软颚擦音	خروف	/x a . " r u: f /	k
z	z	清软颚擦音	زهور	/" z h u: r /	s
ð	D	浊齿间擦音	ذلك	/" D a: . l l k /	D
ðʃ	D_ʔ\	咽化浊齿间擦音	ظلام	/D_ʔ\ A_ʔ\ . " l a: m /	D
ħ	X\	清咽擦音	الحيين	/ʔ a l . " X l i: n /	k
ŋ	N	软颚鼻塞音	هونغ كونغ	/h O N . " k O N g /	k
ɣ	G	浊软颚擦音	غريبة	/G l . " r i: . b a /	k
ʃ	S	清龈后擦音	شمس	/" S a m s /	S
ʒ	Z	浊龈后擦音	حالكيت	/Z a . " k e: t /	S
ʔ	ʔ	声门闭锁音	مؤسسة	/m u . " ʔ a s . s a . s a /	
ʕ	ʔ\	浊咽擦音	عام	/" ʔ\ a: m m /	k
dʒ	dZ	浊龈后塞擦音	جامعة	/" dZ a: m . ʔ\ a /	S

IPA	X-SAMPA	描述	示例	发音	语音视位
θ	T	浊齿间擦音	ثلاثة	/T a . " l a : . T a /	T
ħ	h	浊喉擦音	هال	/ " h l a : l /	k
元音					
æ	a	中前不圆唇开短元音	سفر	/ " s a . f a r /	a
ɑ̣	A_? \	咽化开后不圆唇短元音	صلب	/ " s_? \ A_? \ b /	a
æ:	a:	中前不圆唇开元音	باب	/ " b a : b /	a
ɑ̣:	A_?:	咽化开后不圆唇长元音	ناضح	/ " n A_?: . D_? \ i_? \ dZ /	a
a	A	开央不圆唇短元音	wifi	/ " w A j . f A j /	a
i	i	紧音闭前不圆唇短元音 (MSA)	إسحاق	/ ? i s . " X \ A_? \ : q /	i
ɪ	我	松音闭前不圆唇短元音	بنت	/ " b l n t /	i
ị	i_? \	咽化闭前不圆唇短元音	طفل	/ " t_? \ i_? \ f l l /	i
i:	i:	闭前不圆唇长元音	سبيل	/ s a . " b i : l /	i
ị:	i_?:	咽化闭前不圆唇长元音	رطب	/ r A_? \ . " t_? \ i_?: b /	i

IPA	X-SAMPA	描述	示例	发音	语音视位
u	u	紧音闭后圆唇短元音 (MSA)	مختار	/"m u x . t a . r i ? \ /	u
ʊ	U	松音闭后圆唇短元音	رسوم	/r U . " s u : m /	u
u <sup>ʕ</sup>	u_?\ ʕ	咽化闭后圆唇短元音	عصفور	/ ? \ u _ ? \ s _ ? \ . " f u : r /	u
u:	u:	闭后圆唇长元音	توت	/" t u : t /	u
u <sup>ʕ</sup> :	u_?\ ʕ:	咽化闭后圆唇长元音	صور	/" s _ ? \ u _ ? \ : r /	u
e	e	中前不圆唇短元音	إِنْتَرْنِت	/" s e n t /	e
e:	e:	中前不圆唇长元音	إيش	/" ? e : S /	e
ɔ	O	半开后圆唇短元音	دولار	/d O . " l A r /	O
ɔ:	O:	半开后圆唇长元音	لون	/" l O : n /	O

## 加泰罗尼亚语 (ca-ES)

下表列出了 Amazon Polly 支持的 Arlet 加泰罗尼亚语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				

IPA	X-SAMPA	描述	示例	语音视位
p	p	清双唇塞音	ploure	p
t	t	清齿龈塞音	Tarragona	t
k	k	清软顎塞音	com	k
b	b	浊双唇塞音	bata	p
d	d	浊齿龈塞音	endoll	t
g	g	浊软顎塞音	gros	k
m	m	浊双唇鼻音	manera	p
n	n	浊齿龈鼻音	donar	t
ɲ	J	浊硬顎鼻音	any	J
ŋ	N	浊软顎鼻音	pingüí	k
ɫ	5	浊软顎化齿龈边音 (含糊舌边音 //)	albercoc	l
ʎ	L	浊硬顎边音	llop	J
r	r	浊齿龈颤音	arra	r
ɾ	4	浊齿龈轻拍音	para	t
f	f	清唇齿擦音	èmfasi	f
s	s	清齿龈摩擦音	sac	s
z	z	浊齿龈擦音	calzes	s
ʃ	S	清龈后擦音	guix	S
ʒ	Z	浊龈后擦音	col·legi	S
tʃ	tS	清龈后塞擦音	cotxe	S

IPA	X-SAMPA	描述	示例	语音视位
ɖʒ	dʒ	浊龈后塞擦音	platja	S
β	B	浊双唇近音	obert	B
ð	D	浊齿近音	bedoll	T
j	j	浊硬腭近音	noia	i
ɣ	G	浊软腭近音	pega	k
v	v	浊唇齿擦音	afgà	f
w	w	浊唇软腭近音	aigua	u
x	x	清软腭擦音	Jiménez	k
ʝ	j\	浊硬腭擦音	yeso	J
l	l	浊齿龈边音	alondra	t
θ	T	清齿擦音	González	T
元音				
a	a	开后元音	casa	a
e	e	半闭前不圆唇元音	llenya	e
ɛ	E	中前不圆唇开元音	xec	E
i	i	闭前不圆唇元音	visca	i
o	o	半闭后圆唇元音	gos	o
ɔ	O	半开后圆唇元音	joc	O
u	u	闭后圆唇元音	un	u
ə	@	中央元音	casa	@

IPA	X-SAMPA	描述	示例	语音视位
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 中文 ( 粤语 ) (yue-CN)

下表列出了 Amazon Polly 支持的粤语语音的粤拼和国际音标 (IPA) 音素。粤拼是粤语的拼音系统，在学术界和讲粤语人群中常用。IPA 和 X-SAMPA 不常用，但可用于英语支持。表中的 IPA 和 X-SAMPA 符号仅供参考，不应用于中文转录。还将显示粤拼示例和相应的语音视位。

要让 Amazon Polly 使用粤拼语音发音，请使用 `phoneme alphabet="x-amazon-jyutping"` 标签。

以下示例展示了每种标准的情况。

粤拼：

```
<speak>
  ## <phoneme alphabet="x-amazon-jyutping" ph="sing2">#</phoneme>#
  ## <phoneme alphabet="x-amazon-jyutping" ph="seng2">#</phoneme>#
</speak>
```

IPA：

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

X-SAMPA：

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

&lt;/speak&gt;

**Note**

Amazon Polly 仅接受 UTF-8 编码的粤语输入。

## 音素/语音视位表

粤拼	IPA	X-SAMPA	描述	粤拼示例	语音视位
<b>辅音</b>					
b	p	p	清双唇塞音	巴, baa1	p
c	ts <sup>h</sup>	ts_h	送气清齿龈擦音	叉, caa1	s
d	t	t	清齿龈塞音	打, daa2	t
f	f	f	清唇齿擦音	花, faa1	f
g	k	k	清软顎塞音	家, gaa1	k
gw	k <sup>w</sup>	k_w	唇音化清软顎塞音	瓜, gwaa1	u
h	h	h	清喉擦音	哈, haa1	k
k	k <sup>h</sup>	k_h	送气清软顎塞音	卡, kaa1	k
kw	k <sup>wh</sup>	k_wh	唇音化送气清软顎塞音	誇, kwaa1	u
l	l	l	齿龈边音	啦, laa1	t
m	m	m	双唇鼻音	媽, maa1	p
m	m	m=	音节双唇鼻音	唔, m4	p
ng	ŋ	N	软顎鼻音	牙, ngaa4	k
ng	ŋ	N=	音节软顎鼻音	吳, ng4	k

粤拼	IPA	X-SAMPA	描述	粤拼示例	语音视位
n	n	n	齿龈鼻音	拿, naa4	t
p	p <sup>h</sup>	p_h	送气清双唇塞音	趴, paa1	p
s	s	s	清齿龈摩擦音	沙, saa1	s
t	t <sup>h</sup>	t_h	送气清齿龈塞音	他, taa1	t
w	w	w	浊圆唇软颚近音	娃, waa1	u
y	j	j	硬颚近音	也, jaa5	i
z	ts	ts	清齿龈塞擦音	渣, zaa1	s
元音					
a	ɐ	6	次开央元音	吉, gat1	a
aa	ɑ	A	开后不圆唇元音	家, gaa1	a
aaɪ	aɪ	Ai	双元音	街, gaai1	a
aaʊ	aʊ	Au	双元音	交, gaau1	a
ai	ei	6i	双元音	雞, gai1	a
au	eu	6u-	双元音	溝, kau1	a
e	ɛ	E	中前不圆唇开元音	爹, de1	E
ei	ei	ei	双元音	基, gei1	e
eo	ɵ	8	半闭央圆唇元音	春, ceon1	o
eoɪ	ɵy	8y	双元音	居, geoi1	o
eu	ɛʊ	Eu	双元音	掉 in 掉垃圾, deu6	E

粤拼	IPA	X-SAMPA	描述	粤拼示例	语音视位
i	i	i	闭前不圆唇元音	斯, si1	i
i	我	l	近闭近前不圆唇元音	激, gik1	i
iu	iu	iu	双元音	驕, giu1	i
o	ɔ	O	半开后圆唇元音	哥, go1	O
oe	œ	9	半开前圆唇元音	鋸, goe3	O
oi	ɔi	Oi	双元音	該, goi1	O
ou	ou	ou	双元音	高, gou1	o
u	u	u	闭后圆唇元音	姑, gu1	u
u	ʊ	U	次闭次后圆唇元音	谷, guk5	u
ui	ui	ui	双元音	刼, gui6	u
yu	y	y	闭前圆唇元音	於, jyu1	u
音调标记和其他符号					
1			高平	詩, si1	
2			中升	史, si2	
3			中平	試, si3	
4			很低平	時, si4	
5			低升	市, si5	
6			低平	是, si6	
-	.	.	音节划分	語音 jyu5-jam1	

## 中文 ( 普通话 ) (cmn-CN)

下表列出了 Amazon Polly 支持的中文普通话语音的拼音和国际音标 (IPA) 音素。拼音是标准汉语拼音的国际标准。IPA 和 X-SAMPA 不常用，但可用于英语支持。表中的 IPA 和 X-SAMPA 符号仅供参考，不应用于中文转录。还将显示拼音示例和相应的语音视位。

要让 Amazon Polly 使用拼音语音发音，请使用 `phoneme alphabet="x-amazon-phonetic standard used"` 标签。

以下示例展示了每种标准的情况。

拼音：

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

IPA：

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

X-SAMPA：

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

### Note

Amazon Polly 仅接受 UTF-8 编码的中文普通话输入。Amazon Polly 目前不支持 GB 18030 编码标准。

## 音素/语音视位表

拼音	IPA	X-SAMPA	描述	拼音示例	语音视位
辅音					
f	f	f	清唇齿擦音	发, fa1	f
h	h	h	清喉擦音	和, he2	k
g	k	k	清软顎塞音	古, gu3	k
k	k <sup>h</sup>	k_h	送气清软顎塞音	苦, ku3	k
l	l	l	齿龈边音	拉, la1	t
m	m	m	双唇鼻音	骂, ma4	p
n	n	n	齿龈鼻音	那, na4	t
ng	ŋ	N	软顎鼻音	正, zheng4	k
b	p	p	清双唇塞音	爸, ba4	p
p	p <sup>h</sup>	p_h	送气清双唇塞音	怕, pa4	p
s	s	s	清齿龈摩擦音	四, si4	s
x	ç	s\	清龈顎擦音	西, xi1	J
sh	ʃ	s`	清卷舌擦音	是, shi4	S
d	t	t	清齿龈塞音	打, da3	t
t	t <sup>h</sup>	t_h	送气清齿龈塞音	他, ta1	t
zh	ʈʂ	t`s`	清卷舌塞擦音	之, zhi1	S
ch	ʈʂ <sup>h</sup>	t`s`_h	送气清卷舌塞擦音	吃, chi1	S
s	ʈʂ	ts	清齿龈塞擦音	字, zi4	s

拼音	IPA	X-SAMPA	描述	拼音示例	语音视位
j	$\text{tɕ}$	ts\	清龈腭塞擦音	鸡, ji1	J
q	$\text{tɕ}^h$	ts\_h	送气清龈腭塞擦音	七, qi1	J
c	$\text{ts}^h$	ts_h	送气清齿龈擦音	次, ci4	s
w	w	w	浊圆唇软腭近音	我, wo3	u
r	$\text{ʐ}$	z`	浊卷舌擦音	日, ri4	S
“er”和“r”色彩音节					
er	$\text{ə}$	@`	r 色彩中央元音	二, er4	@
-r			r 色彩音节	馅儿, xianr4	@
元音					
e	$\text{ɤ}$	7	半闭后不圆唇元音	恶, e4	e
e	$\text{ə}$	@	中央元音	恩, en1	@
a	a	a	开前不圆唇元音	安, an1	a
ai	aɪ	aɪ	双元音	爱, ai4	a
ao	aʊ	aʊ	双元音	奥, ao4	a
ei	eɪ	e	双元音	诶, ei4	e
e	$\text{ɛ}$	E	中前不圆唇开元音	姐, jie3	E
i	i	i	闭前不圆唇元音	鸡, ji1	i
ou	oʊ	oʊ	双元音	欧, ou1	o
o	ɔ	O	半开后圆唇元音	哦, o4	o
u	u	u	闭后圆唇元音	主, zhu3	u

拼音	IPA	X-SAMPA	描述	拼音示例	语音视位
yu	y	y	闭前圆唇元音	于, yu2	u
音调标记和其他符号					
1			高平调	淤, yu1	
2			升调	鱼, yu2	
3			低 (降-升) 调	语, yu3	
4			降调	育, yu4	
0			轻音调	的, de0	
-	.	.	音节划分	语音 yu3-yin1	

## 捷克语 (cs-CZ)

下表列出了 Amazon Polly 支持的捷克语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
p	p	清双唇塞音	pes	p
t	t	清齿龈塞音	tok	t
c	c	清硬顎塞音	t'uk	J
k	k	清软顎塞音	kos	k
b	b	浊双唇塞音	bez	p

IPA	X-SAMPA	描述	示例	语音视位
d	d	浊齿龈塞音	dok	t
ɟ	ɟ\	浊硬腭塞音	d'as	J
g	g	浊软腭塞音	gum	k
f	f	清唇齿擦音	film	f
v	v	浊唇齿擦音	ves	f
s	s	清齿龈摩擦音	sen	s
z	z	浊齿龈擦音	zel	s
ʃ	S	清龈后擦音	šel	S
ʒ	Z	浊龈后擦音	žen	S
x	x	清软腭擦音	chat	k
ɦ	h	浊喉擦音	hus	k
ts	ts	清齿龈塞擦音	co	s
tʃ	tS	清龈后塞擦音	čin	S
dz	dz	浊齿龈塞擦音	špicberský	s
dʒ	dZ	清齿龈塞擦音	džin	S
m	m	双唇鼻音	mor	p
n	n	齿龈鼻音	nos	t
ɲ	J	硬腭鼻音	ňader	J
ŋ	N	软腭鼻音	banka	k
r	r	浊齿龈颤音	rys	r

IPA	X-SAMPA	描述	示例	语音视位
ɹ	r_r	浊音抬高的齿槽擦音颤音	řez	r
ɹ̥	r_0_r	清音抬高的齿槽擦音颤音	keř	r
l	l	齿龈边音	les	t
j	j	硬颚近音	jen	i
w	w	浊圆唇软颚近音	Watson	u
ɾ	r_=	音节浊齿槽颤音	krk	r
ɺ	l_=	音节齿龈边音	vlna	t
<b>元音</b>				
a	a	开前不圆唇元音	lan	a
a:	a:	长开前不圆唇元音	lán	a
ɛ	E	中前不圆唇开元音	let	E
ɛ:	E:	长中前不圆唇开元音	lét	E
ɪ	ɪ	近闭近前不圆唇元音	bit	i
i:	i:	长闭前不圆唇元音	bít	i
o	o	半闭后圆唇元音	hol	o
o:	o:	长半闭后圆唇元音	gól	o
u	u	闭后圆唇元音	pul	u
u:	u:	长闭后圆唇元音	půl	u

IPA	X-SAMPA	描述	示例	语音视位
ɑ̃u	au	双元音	自动	a
ɛ̃u	Eu	双元音	euro	E
ɔ̃u	ou	双元音	mouk	o
其他符号				
ˈ	ˈ	主重音		
·	·	音节划分		

## 丹麦语 (da-DK)

下表列出了 Amazon Polly 支持的丹麦语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	bat	p
d	d	浊齿龈塞音	da	t
ð	D	浊齿擦音	mad、thriller	T
f	f	清唇齿擦音	fat	f
g	g	浊软顎塞音	gat	k
h	h	清喉擦音	hat	k
j	j	硬顎近音	jo	i
k	k	清软顎塞音	kat	k

IPA	X-SAMPA	描述	示例	语音视位
l	l	齿龈边音	ladt	t
m	m	双唇鼻音	mat	p
n	n	齿龈鼻音	nay	t
ŋ	N	软顎鼻音	lang	k
p	p	清双唇塞音	pande	p
r	r	齿龈颤音	thriller、story	r
ʁ	R	浊小舌擦音	rat	k
s	s	清齿龈摩擦音	sat	s
t	t	清齿龈塞音	tal	t
v	v	浊唇齿擦音	vat	f
w	w	浊圆唇软顎近音	hav、weekend	u
元音				
ø	2	半闭前圆唇元音	øst	o
ø:	2:	长半闭前圆唇元音	øse	o
e	6	次开央元音	mor	a
œ	9	半开前圆唇元音	skøn、grønt	O
œ:	9:	长半开前圆唇元音	høne、gøre	O
ə	@	中央元音	ane	@
æ:	{:	长次开前不圆唇元音	male	a
a	a	开前不圆唇元音	man	a

IPA	X-SAMPA	描述	示例	语音视位
æ	{	次开前不圆唇元音	adresse	a
ɑ	A	开后不圆唇元音	lak、tak	a
ɑ:	A:	长开后不圆唇元音	rase	a
e	e	半闭前不圆唇元音	midt	e
e:	e:	长半闭前不圆唇元音	mele	e
ɛ	E	中前不圆唇开元音	mæt	E
ɛ:	E:	长中前不圆唇开元音	mæle	E
i	i	闭前不圆唇元音	mit	i
i:	i:	长闭前不圆唇元音	mile	i
o	o	半闭后圆唇元音	foto	o
o:	o:	长半闭后圆唇元音	mole	o
ɔ	O	半开后圆唇元音	mund	O
ɔ:	O:	长半开后圆唇元音	måle	O
ɒ:	Q:	长开后圆唇元音	morse	O
u	u	闭后圆唇元音	lusk	u
u:	u:	长闭后圆唇元音	mule	u
ʌ	V	半开后不圆唇	kører	E
y	y	闭前圆唇元音	yt	u
y:	y:	长闭前圆唇元音	hyle	u

IPA	X-SAMPA	描述	示例	语音视位
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 荷兰语 ( 比利时 ) (nl-BE)

下表列出了 Amazon Polly 支持的比利时荷兰语 ( 佛兰芒语 ) 语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	bak	p
d	d	浊齿龈塞音	dak	t
ɖʒ	dʒ	浊龈后塞擦音	manager	S
f	f	清唇齿擦音	fel	f
g	g	浊软顎塞音	goal	k
ɣ	G	浊软顎擦音	hoed	k
ɦ	h\	浊喉擦音	hand	k
j	j	硬顎近音	ja	i
k	k	清软顎塞音	kap	k
l	l	齿龈边音	land	t

IPA	X-SAMPA	描述	示例	语音视位
m	m	双唇鼻音	met	p
n	n	齿龈鼻音	net	t
ŋ	N	软顎鼻音	bang	k
p	p	清双唇塞音	pak	p
r	r	齿龈颤音	rand	r
s	s	清齿龈摩擦音	sein	s
ʃ	S	清龈后擦音	show	S
t	t	清齿龈塞音	tak	t
v	v	浊唇齿擦音	vel	f
ʋ	v\	唇齿近音	wit	f
x	x	清软顎擦音	toch	k
z	z	浊齿龈擦音	ziin	s
ʒ	Z	浊龈后擦音	bagage	S
元音				
ø:	2:	长半闭前圆唇元音	neus	o
œy	9y	双元音	buit	O
ə	@	中央元音	de	@
a:	a:	长开前不圆唇元音	baad	a
ɑ:	A	开后不圆唇元音	bad	a
e:	e:	长半闭前不圆唇元音	beet	e

IPA	X-SAMPA	描述	示例	语音视位
ɜ:	3:	长半开央不圆唇元音	barrière	E
ɛ	E	中前不圆唇开元音	bed	E
ɛi	Ei	双元音	beet	E
i	i	闭前不圆唇元音	vier	i
ɪ	我	近闭近前不圆唇元音	pit	i
o:	o:	长半闭后圆唇元音	boot	o
ɔ	O	半开后圆唇元音	pot	O
u	u	闭后圆唇元音	hoed	u
ʌu	Vu	双元音	fout	E
y:	y:	长闭前圆唇元音	fuut	u
ʏ	Y	次闭次前圆唇元音	hut	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 荷兰语 (nl-NL)

下表列出了 Amazon Polly 支持的荷兰语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

## 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	bak	p
d	d	浊齿龈塞音	dak	t
ɖʒ	dʒ	浊龈后塞擦音	manager	S
f	f	清唇齿擦音	fel	f
g	g	浊软顎塞音	goal	k
ɣ	G	浊软顎擦音	hoed	k
ɦ	h\	浊喉擦音	hand	k
j	j	硬顎近音	ja	i
k	k	清软顎塞音	kap	k
l	l	齿龈边音	land	t
m	m	双唇鼻音	met	p
n	n	齿龈鼻音	net	t
ŋ	N	软顎鼻音	bang	k
p	p	清双唇塞音	pak	p
r	r	齿龈颤音	rand	r
s	s	清齿龈摩擦音	sein	s
ʃ	S	清龈后擦音	show	S
t	t	清齿龈塞音	tak	t
v	v	浊唇齿擦音	vel	f

IPA	X-SAMPA	描述	示例	语音视位
ʋ	v\	唇齿近音	wit	f
x	x	清软顎擦音	toch	k
z	z	浊齿龈擦音	ziin	s
ʒ	Z	浊龈后擦音	bagage	S
元音				
ø:	2:	长半闭前圆唇元音	neus	o
œy	9y	双元音	buit	O
ə	@	中央元音	de	@
a:	a:	长开前不圆唇元音	baad	a
ɑ:	A	开后不圆唇元音	bad	a
e:	e:	长半闭前不圆唇元音	beet	e
ɜ:	3:	长半开央不圆唇元音	barrière	E
ɛ	E	中前不圆唇开元音	bed	E
ɛi	Ei	双元音	beet	E
i	i	闭前不圆唇元音	vier	i
ɪ	我	近闭近前不圆唇元音	pit	i
o:	o:	长半闭后圆唇元音	boot	o
ɔ	O	半开后圆唇元音	pot	O
u	u	闭后圆唇元音	hoed	u

IPA	X-SAMPA	描述	示例	语音视位
ʌu	Vu	双元音	fout	E
y:	y:	长闭前圆唇元音	fuut	u
ɻ	Y	次闭次前圆唇元音	hut	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 英语 ( 美国 ) (en-US)

下表列出了 Amazon Polly 支持的美式英语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɹ̄ʒ	dʒ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k

IPA	X-SAMPA	描述	示例	语音视位
j	j	硬顎近音	yes	i
k	k	清软顎塞音	cat	k
l	l	齿龈边音	lay	l
m	m	双唇鼻音	mouse	p
n	n	齿龈鼻音	nap	t
ŋ	N	软顎鼻音	thing	k
p	p	清双唇塞音	speak	p
r	r\	齿龈近音	red	r
s	s	清齿龈摩擦音	seem	s
ʃ	S	清龈后擦音	ship	S
t	t	清齿龈塞音	trap	t
tʃ	tS	清龈后塞擦音	chart	S
θ	T	清齿擦音	thin	T
v	v	浊唇齿擦音	vest	f
w	w	浊圆唇软顎近音	west	u
z	z	浊齿龈擦音	zero	s
ʒ	Z	浊龈后擦音	vision	S
<b>元音</b>				
ə	@	中央元音	arena	@
ə̃	@`	中央 r 色彩元音	reader	@

IPA	X-SAMPA	描述	示例	语音视位
æ	{	次开前不圆唇元音	trap	a
aɪ	al	双元音	price	a
aʊ	aU	双元音	mouth	a
ɑ	A	长开后不圆唇元音	father	a
eɪ	el	双元音	face	e
ɜ̃	3`	中央不圆唇的 r 色彩开元音	nurse	E
ɛ	E	中前不圆唇开元音	dress	E
i	i	长闭前不圆唇元音	fleece	i
ɪ	我	近闭近前不圆唇元音	kit	i
oʊ	oU	双元音	goat	o
ɔ	O	长开中后圆唇元音	thought	O
ɔɪ	Oɪ	双元音	choice	O
u	u	长闭后圆唇元音	goose	u
ʊ	U	次闭次后圆唇元音	foot	u
ʌ	V	半开后不圆唇元音	strut	E
<b>其他符号</b>				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 英语 ( 澳大利亚 ) (en-AU)

下表列出了 Amazon Polly 支持的澳大利亚英语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɹ̥ʒ	dʒ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k
j	j	硬顎近音	yes	i
k	k	清软顎塞音	cat	k
l	l	齿龈边音	lay	t
ɹ̥	l̥=	音节齿龈边音	battle	t
m	m	双唇鼻音	mouse	p
ɱ	m=	音节双唇鼻音	anthem	p
n	n	齿龈鼻音	nap	t
ɳ	n=	音节齿龈鼻音	button	t
ŋ	N	软顎鼻音	thing	k

IPA	X-SAMPA	描述	示例	语音视位
p	p	清双唇塞音	pin	p
r	r\	齿龈近音	red	r
s	s	清齿龈摩擦音	seem	s
ʃ	S	清龈后擦音	ship	S
t	t	清齿龈塞音	task	t
tʃ	tS	清龈后塞擦音	chart	S
θ	T	清齿擦音	thin	T
v	v	浊唇齿擦音	vest	f
w	w	浊圆唇软腭近音	west	u
z	z	浊齿龈擦音	zero	s
ʒ	Z	浊龈后擦音	vision	S
元音				
ə	@	中央元音	arena	@
əʊ	@U	双元音	goat	@
æ	{	次开前不圆唇元音	trap	a
aɪ	al	双元音	price	a
aʊ	aU	双元音	mouth	a
ɑ:	A:	长开后不圆唇元音	father	a
eɪ	el	双元音	face	e
ɜ:	3:	长半开央不圆唇元音	nurse	E

IPA	X-SAMPA	描述	示例	语音视位
ɛ	E	中前不圆唇开元音	dress	E
ɛə	E@	双元音	square	E
i:	i	长闭前不圆唇元音	fleece	i
ɪ	我	近闭近前不圆唇元音	kit	i
ɪə	l@	双元音	near	i
ɔ:	OI	长半开后圆唇元音	thought	O
ɔɪ	OI	双元音	choice	O
ɒ	Q	开后圆唇元音	lot	O
u:	u:	长闭后圆唇元音	goose	u
ʊ	U	次闭次后圆唇元音	foot	u
ʊə	U@	双元音	cure	u
ʌ	V	半开后不圆唇元音	strut	E
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 英语 ( 英国 ) (en-GB)

下表列出了 Amazon Polly 支持的英式英语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

## 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɹ̥d͡ʒ	dʒ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k
j	j	硬顎近音	yes	i
k	k	清软顎塞音	cat	k
l	l	齿龈边音	lay	t
ɫ	l=	音节齿龈边音	battle	t
m	m	双唇鼻音	mouse	p
ᵻ	m=	音节双唇鼻音	anthem	p
n	n	齿龈鼻音	nap	t
ɳ	n=	音节齿龈鼻音	button	t
ŋ	N	软顎鼻音	thing	k
p	p	清双唇塞音	pin	p
r	r\	齿龈近音	red	r
s	s	清齿龈摩擦音	seem	s

IPA	X-SAMPA	描述	示例	语音视位
ʃ	S	清龈后擦音	ship	S
t	t	清齿龈塞音	task	t
tʃ	tS	清龈后塞擦音	chart	S
θ	T	清齿擦音	thin	T
v	v	浊唇齿擦音	vest	f
w	w	浊圆唇软顎近音	west	u
z	z	浊齿龈擦音	zero	s
ʒ	Z	浊龈后擦音	vision	S
元音				
ə	@	中央元音	arena	@
əʊ	@U	双元音	goat	@
æ	{	次开前不圆唇元音	trap	a
aɪ	al	双元音	price	a
aʊ	aU	双元音	mouth	a
ɑ:	A:	长开后不圆唇元音	father	a
eɪ	el	双元音	face	e
ɜ:	3:	长半开央不圆唇元音	nurse	E
ɛ	E	中前不圆唇开元音	dress	E
ɛə	E@	双元音	square	E
i:	i	长闭前不圆唇元音	fleece	i

IPA	X-SAMPA	描述	示例	语音视位
ɪ	我	近闭近前不圆唇元音	kit	i
ɪə	l@	双元音	near	i
ɔ:	O:	长半开后圆唇元音	thought	O
ɔɪ	Oɪ	双元音	choice	O
ɒ	Q	开后圆唇元音	lot	O
u:	u:	长闭后圆唇元音	goose	u
ʊ	U	次闭次后圆唇元音	foot	u
ʊə	U@	双元音	cure	u
ʌ	V	半开后不圆唇元音	strut	E
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 英语 ( 印度 ) (en-IN)

下表列出了 Amazon Polly 支持的印度英语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

有关与印度英语结合使用的其他发音，请参阅 [印地语 \(hi-IN\)](#)。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				

IPA	X-SAMPA	描述	示例	语音视位
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɖ	dʒ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k
j	j	硬顎近音	yes	i
k	k	清软顎塞音	cat	k
l	l	齿龈边音	lay	t
ɭ	l=	音节齿龈边音	battle	t
m	m	双唇鼻音	mouse	p
ɱ	m=	音节双唇鼻音	anthem	p
n	n	齿龈鼻音	nap	t
ɳ	n=	音节齿龈鼻音	nap	t
ŋ	N	软顎鼻音	thing	k
p	p	清双唇塞音	pin	p
r	r\	齿龈近音	red	r
s	s	清齿龈摩擦音	seem	s
ʃ	S	清龈后擦音	ship	S

IPA	X-SAMPA	描述	示例	语音视位
t	t	清齿龈塞音	task	t
tʃ	tS	清龈后塞擦音	chart	S
θ	T	清齿擦音	thin	T
v	v	浊唇齿擦音	vest	f
w	w	浊圆唇软顎近音	west	u
z	z	浊齿龈擦音	zero	s
ʒ	Z	浊龈后擦音	vision	S
元音				
ə	@	中央元音	arena	@
əʊ	@U	双元音	goat	@
æ	{	次开前不圆唇元音	trap	a
aɪ	al	双元音	price	a
aʊ	aU	双元音	mouth	a
ɑ:	A:	长开后不圆唇元音	father	a
eɪ	el	双元音	face	e
ɜ:	3:	长半开央不圆唇元音	nurse	E
ɛ	E	中前不圆唇开元音	dress	E
ɛə	E@	双元音	square	E
i:	i	长闭前不圆唇元音	fleece	i

IPA	X-SAMPA	描述	示例	语音视位
ɪ	我	近闭近前不圆唇元音	kit	i
ɪə	l@	双元音	near	i
ɔ:	Ol	长半开后圆唇元音	thought	O
ɔɪ	Ol	双元音	choice	O
ɒ	Q	开后圆唇元音	lot	O
u:	u:	长闭后圆唇元音	goose	u
ʊ	U	次闭次后圆唇元音	foot	u
ʊə	U@	双元音	cure	u
ʌ	V	半开后不圆唇元音	strut	E
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 英语 (爱尔兰) (en-IE)

下表列出了 Amazon Polly 支持的爱尔兰英语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				

IPA	X-SAMPA	描述	示例	语音视位
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɟ	dʒ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k
j	j	硬顎近音	yes	i
k	k	清软顎塞音	cat	k
l	l	齿龈边音	lay	t
m	m	双唇鼻音	mouse	p
n	n	齿龈鼻音	nap	t
ŋ	N	软顎鼻音	thing	k
p	p	清双唇塞音	speak	p
r	r\	齿龈近音	red	r
s	s	清齿龈摩擦音	seem	s
ʃ	S	清龈后擦音	ship	S
t	t	清齿龈塞音	trap	t
tʃ	tS	清龈后塞擦音	chart	S
θ	T	清齿擦音	thin	T

IPA	X-SAMPA	描述	示例	语音视位
v	v	浊唇齿擦音	vest	f
w	w	浊圆唇软顎近音	west	u
z	z	浊齿龈擦音	zero	s
ʒ	Z	浊龈后擦音	vision	S
<b>元音</b>				
ə	@	中央元音	arena	@
ə̃	@`	中央 r 色彩元音	reader	@
æ	{	次开前不圆唇元音	trap	a
aɪ	al	双元音	price	a
aʊ	aU	双元音	mouth	a
ɑ	A	长开后不圆唇元音	father	a
eɪ	el	双元音	face	e
ɜ̃	3`	中央不圆唇的 r 色彩开元音	nurse	E
ɛ	E	中前不圆唇开元音	dress	E
i	i	长闭前不圆唇元音	fleece	i
ɪ	我	近闭近前不圆唇元音	kit	i
oʊ	oU	双元音	goat	o
ɔ	O	长开中后圆唇元音	thought	O
ɔɪ	OI	双元音	choice	O

IPA	X-SAMPA	描述	示例	语音视位
u	u	长闭后圆唇元音	goose	u
ʊ	U	次闭次后圆唇元音	foot	u
ʌ	V	半开后不圆唇元音	strut	E
<b>其他符号</b>				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 英语 ( 新西兰 ) (en-NZ)

下表列出了 Amazon Polly 支持的新西兰英语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
<b>辅音</b>				
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɹ̥ʒ	dZ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k

IPA	X-SAMPA	描述	示例	语音视位
j	j	硬颚近音	yes	i
k	k	清软颚塞音	cat	k
l	l	齿龈边音	lay	t
ɹ	l=	音节齿龈边音	battle	t
m	m	双唇鼻音	mouse	p
ɱ	m=	音节双唇鼻音	anthem	p
n	n	齿龈鼻音	nap	t
ɳ	n=	音节齿龈鼻音	button	t
ŋ	N	软颚鼻音	thing	k
p	p	清双唇塞音	pin	p
ɹ	r\	齿龈近音	red	r
s	s	清齿龈摩擦音	seem	s
ʃ	S	清龈后擦音	ship	S
t	t	清齿龈塞音	task	t
tʃ	tS	清龈后塞擦音	chart	S
θ	T	清齿擦音	thin	T
v	v	浊唇齿擦音	vest	f
w	w	浊圆唇软颚近音	west	u
z	z	浊齿龈擦音	zero	s
ʒ	Z	浊龈后擦音	vision	S

IPA	X-SAMPA	描述	示例	语音视位
元音				
ə	@	中央元音	arena	@
əʊ	@U	双元音	goat	@
æ	{	次开前不圆唇元音	trap	a
aɪ	al	双元音	price	a
aʊ	aU	双元音	mouth	a
ɑ:	A:	长开后不圆唇元音	father	a
eɪ	el	双元音	face	e
ɜ:	3:	长半开央不圆唇元音	nurse	E
ɛ	E	中前不圆唇开元音	dress	E
ɛə	E@	双元音	square	E
i:	i	长闭前不圆唇元音	fleece	i
ɪ	我	近闭近前不圆唇元音	kit	i
ɪə	l@	双元音	near	i
ɔ:	O:	长半开后圆唇元音	thought	O
ɔɪ	Oɪ	双元音	choice	O
ɒ	Q	开后圆唇元音	lot	O
u:	u:	长闭后圆唇元音	goose	u
ʊ	U	次闭次后圆唇元音	foot	u

IPA	X-SAMPA	描述	示例	语音视位
ʊə	U@	双元音	cure	u
ʌ	V	半开后不圆唇元音	strut	E
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

Aria 语音讲的是新西兰英语，对毛利语的支持有限。可以进行以下毛利语单词和短语的发音。毛利语短语区分大小写。

English	毛利语
Hello/cheers	Kia ora
Welcome (to)	Nau mai (ki)
Hello (one person)/thank you	Tēnā koe
Hello (three or more people)/thank you	Tēnā koutou
Good morning	Ata mārie
Good morning	Mōrena
Thank you	Ngā mihi
Take care	Ngā manaakitanga
See you	Ka kite
See you later	Mā te wā
Have a good day	Kia pai tō rā

English	毛利语
Merry Christmas	Meri Kirihimete
Maori	Māori
Maori language	te reo Māori
Maori language week	Te wiki o te reo Māori
New Zealand	Aotearoa
Maori New Year	Matariki
Town in New Zealand/Waitangi Day is the national day of New Zealand	Waitangi
One	tahi
Two	rua
Three	toru
Four	whā
Five	rima
Six	ono
Seven	whitu
Eight	waru
Nine	iwa
Ten	tekau
Twenty	rua tekau
Thirty	Toru tekau

## 英语 ( 新加坡 ) ( en-SG )

下表列出了 Amazon Polly 支持的新加坡英语语音的国际音标 ( IPA ) 音素、拓展音标字母评估法 ( X-SAMPA ) 符号和对应的语音视位。

音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɹ̥ʒ	dʒ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k
j	j	硬顎近音	yes	i
k	k	清软顎塞音	cat	k
l	l	齿龈边音	lay	t
ɹ̥	l̥=	音节齿龈边音	battle	t
m	m	双唇鼻音	mouse	p
ṁ	m=	音节双唇鼻音	anthem	p
n	n	齿龈鼻音	nap	t
ṅ	n=	音节齿龈鼻音	button	t
ŋ	N	软顎鼻音	thing	k

IPA	X-SAMPA	描述	示例	语音视位
p	p	清双唇塞音	pin	p
r	r\	齿龈近音	red	r
s	s	清齿龈摩擦音	seem	s
ʃ	S	清龈后擦音	ship	S
t	t	清齿龈塞音	task	t
tʃ	tS	清龈后塞擦音	chart	S
θ	T	清齿擦音	thin	T
v	v	浊唇齿擦音	vest	f
w	w	浊圆唇软顎近音	west	u
z	z	浊齿龈擦音	zero	s
ʒ	Z	浊龈后擦音	vision	S
元音				
ə	@	中央元音	arena	@
əʊ	@U	双元音	goat	@
æ	{	次开前不圆唇元音	trap	a
aɪ	al	双元音	price	a
aʊ	aU	双元音	mouth	a
ɑ:	A:	长开后不圆唇元音	father	a
eɪ	el	双元音	face	e
ɜ:	3:	长半开央不圆唇元音	nurse	E

IPA	X-SAMPA	描述	示例	语音视位
ɛ	E	中前不圆唇开元音	dress	E
ɛə	E@	双元音	square	E
i:	i	长闭前不圆唇元音	fleece	i
ɪ	我	近闭近前不圆唇元音	kit	i
ɪə	l@	双元音	near	i
ɔ:	O:	长半开后圆唇元音	thought	O
ɔɪ	Oɪ	双元音	choice	O
ɒ	Q	开后圆唇元音	lot	O
u:	u:	长闭后圆唇元音	goose	u
ʊ	U	次闭次后圆唇元音	foot	u
ʊə	U@	双元音	cure	u
ʌ	V	半开后不圆唇元音	strut	E
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 英语 ( 南非 ) (en-ZA)

下表列出了 Amazon Polly 支持的南非英语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

## 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɹ̥d͡ʒ	dʒ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k
j	j	硬顎近音	yes	i
k	k	清软顎塞音	cat	k
l	l	齿龈边音	lay	t
ɫ	l=	音节齿龈边音	battle	t
ɬ	K	清边擦音	umhlanga	t
m	m	双唇鼻音	mouse	p
ɱ	m=	音节双唇鼻音	anthem	p
n	n	齿龈鼻音	nap	t
ɳ	n=	音节齿龈鼻音	button	t
ŋ	N	软顎鼻音	thing	k
p	p	清双唇塞音	pin	p
r	r\	齿龈近音	red	r

IPA	X-SAMPA	描述	示例	语音视位
r	r	齿龈颤音	pareis	r
s	s	清齿龈摩擦音	seem	s
ʃ	S	清龈后擦音	ship	S
t	t	清齿龈塞音	task	t
ʧ	tS	清龈后塞擦音	chart	S
θ	T	清齿擦音	thin	T
v	v	浊唇齿擦音	vest	f
w	w	浊圆唇软顎近音	west	u
x	x	清软顎擦音	gauteng	k
z	z	浊齿龈擦音	zero	s
!	!\	后齿龈吸气音	gqeberha	k
	\	齿吸气音	ncube	t
	\	边吸气音	xhosa	t
元音				
ə	@	中央元音	arena	@
əi	@i	双元音	nelspruit	i
əʊ	@U	双元音	goat	@
æ	{	次开前不圆唇元音	trap	a
aɪ	al	双元音	price	a
aʊ	aU	双元音	mouth	a

IPA	X-SAMPA	描述	示例	语音视位
a:	A:	长开后不圆唇元音	father	a
eɪ	eɪ	双元音	face	e
ɜ:	3:	长半开央不圆唇元音	nurse	E
ɛ	E	中前不圆唇开元音	dress	E
ɛə	E@	双元音	square	E
i:	i	长闭前不圆唇元音	fleece	i
iə	ɪ@	双元音	du preez	i
ɪ	我	近闭近前不圆唇元音	kit	i
ɪə	ɪ@	双元音	near	i
ɔ:	O:	长半开后圆唇元音	thought	O
ɔɪ	Oɪ	双元音	choice	O
ɒ	Q	开后圆唇元音	lot	O
u:	u:	长闭后圆唇元音	goose	u
ʊ	U	次闭次后圆唇元音	foot	u
ʊə	U@	双元音	cure	u
ʌ	V	半开后不圆唇元音	strut	E
y	y	闭前圆唇元音	van vuuren	u
其他符号				
'	"	主重音	Alabama	

IPA	X-SAMPA	描述	示例	语音视位
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 英语 ( 威尔士 ) (en-GB-WLS)

下表列出了 Amazon Polly 支持的威尔士英语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
<b>辅音</b>				
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɟ	dʒ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k
j	j	硬顎近音	yes	i
k	k	清软顎塞音	cat	k
l	l	齿龈边音	lay	t
ɭ	l=	音节齿龈边音	battle	t
m	m	双唇鼻音	mouse	p

IPA	X-SAMPA	描述	示例	语音视位
m̥	m=	音节双唇鼻音	anthem	p
n	n	齿龈鼻音	nap	t
ɱ	n=	音节齿龈鼻音	nap	t
ŋ	N	软顎鼻音	thing	k
p	p	清双唇塞音	pin	p
ɹ	r\	齿龈近音	red	r
s	s	清齿龈摩擦音	seem	s
ʃ	S	清龈后擦音	ship	S
t	t	清齿龈塞音	task	t
tʃ	tS	清龈后塞擦音	chart	S
θ	T	清齿擦音	thin	T
v	v	浊唇齿擦音	vest	f
w	w	浊圆唇软顎近音	west	u
z	z	浊齿龈擦音	zero	s
ʒ	Z	浊龈后擦音	vision	S
元音				
ə	@	中央元音	arena	@
əʊ	@U	双元音	goat	@
æ	{	次开前不圆唇元音	trap	a
aɪ	al	双元音	price	a

IPA	X-SAMPA	描述	示例	语音视位
aʊ	aU	双元音	mouth	a
ɑ:	A:	长开后不圆唇元音	father	a
eɪ	eɪ	双元音	face	e
ɜ:	3:	长半开央不圆唇元音	nurse	E
ɛ	E	中前不圆唇开元音	dress	E
ɛə	E@	双元音	square	E
i:	i	长闭前不圆唇元音	fleece	i
ɪ	我	近闭近前不圆唇元音	kit	i
ɪə	l@	双元音	near	i
ɔ:	Oɪ	长半开后圆唇元音	thought	O
ɔɪ	Oɪ	双元音	choice	O
ɒ	Q	开后圆唇元音	lot	O
u:	u:	长闭后圆唇元音	goose	u
ʊ	U	次闭次后圆唇元音	foot	u
ʊə	U@	双元音	cure	u
ʌ	V	半开后不圆唇元音	strut	E
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	

IPA	X-SAMPA	描述	示例	语音视位
.	.	音节划分	A.la.ba.ma	

## 芬兰语 (fi-FI)

下表列出了 Amazon Polly 支持的芬兰语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
芬兰语辅音				
p	p	清双唇塞音	[p]ankki	p
t	t	清齿龈塞音	[t]alo	t
k	k	清软腭塞音	[k]aali	k
d	d	浊齿龈塞音	[d]ata	t
s	s	清齿龈摩擦音	[s]ali	s
h	h	清喉擦音	[h]attu	k
u	v\	浊唇齿近音	[v]aivā	v
j	j	硬腭近音	[j]oki	i
l	l	齿龈边音	[l]oma	t
r	r	浊齿龈颤音	[r]iita	r
m	m	双唇鼻音	[m]ato	p
n	n	齿龈鼻音	[n]enää	t
ŋ	N	软腭鼻音	he[n]ki	k

IPA	X-SAMPA	描述	示例	语音视位
外来词中的辅音				
b	b	浊双唇塞音	[b]ussi	p
f	f	清唇齿擦音	[f]irma	v
w	w	浊圆唇软顎近音	[w]iki	u
z	z	浊齿龈擦音	[z]ulu	s
g	g	浊软顎塞音	[g]aala	k
ʃ	S	清龈后擦音	[sh]akki	S
ʒ	Z	浊龈后擦音	[g]enre	S
θ	T	清齿擦音	ear[th]	T
ð	D	浊齿擦音	ei[th]er	T
短元音				
i	i	闭前不圆唇元音	k[i]lo	i
ɛ	E	中前不圆唇开元音	k[e]sä	E
æ	{	次开前不圆唇元音	k[ä]ly	A
y	y	闭前圆唇元音	k[y]lä	u
ø	2	半闭前圆唇元音	p[ö]ly	O
u	u	闭后圆唇元音	k[u]lo	u
ɔ	O	半开后圆唇元音	k[o]lo	O
a	A	开后不圆唇元音	k[a]la	A
长元音				

IPA	X-SAMPA	描述	示例	语音视位
i:	i:	长闭前不圆唇元音	s[ii]li	i
ɛ:	E:	长中前不圆唇开元音	[ee]tu	E
æ:	{:	长次开前不圆唇元音	t[ää]llä	A
y:	y:	长闭前不圆唇元音	t[yy]li	u
ø:	2:	长半闭前圆唇元音	t[öö]lö	O
u:	u:	长闭后圆唇元音	t[uu]li	u
ɔ:	O:	长开中后圆唇元音	r[oo]li	O
ɑ:	A:	长开后不圆唇元音	k[aa]su	A
双元音				
ɛi	Ei	双元音	l[ei]pä	E
æi	{i	双元音	[äi]ti	A
ui	ui	双元音	k[ui]n	u
ai	Ai	双元音	k[ai]kki	A
ɔi	Oi	双元音	p[oi]ka	O
øi	2i	双元音	s[öi]n	O
yi	yi	双元音	l[yi]jy	u
au	Au	双元音	s[au]na	A
ɔu	Ou	双元音	k[ou]lu	O
ɛu	Eu	双元音	r[eu]na	E

IPA	X-SAMPA	描述	示例	语音视位
iu	iu	双元音	v[iu]lu	i
æy	{y	双元音	t[äy]nnä	A
øy	2y	双元音	k[öy]hä	O
ɛy	Ey	双元音	pes[ey]tyä	E
iy	iy	双元音	käär[iy]tyä	i
iɛ	iE	双元音	t[ie]	i
yø	y2	双元音	[yö]	u
uo	uO	双元音	t[uo]	u
英语外来词中的元音				
ɪ	我	近闭近前不圆唇元音	b[i]t	i
ʊ	U	次闭次后圆唇元音	b[oo]k	u
ə	@	中央元音	[a]bout	@
ʌ	V	半开后不圆唇元音	c[u]t	E

## 法语 (fr-FR)

下表列出了 Amazon Polly 支持的法语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				

IPA	X-SAMPA	描述	示例	语音视位
b	b	浊双唇塞音	boire	p
d	d	浊齿龈塞音	madame	t
f	f	清唇齿擦音	femme	f
g	g	浊软顎塞音	grand	k
ɥ	H	唇硬顎近音	bruit	u
j	j	硬顎近音	meilleur	i
k	k	清软顎塞音	quatre	k
l	l	齿龈边音	malade	t
m	m	双唇鼻音	maison	p
n	n	齿龈鼻音	astronome	t
ɲ	J	硬顎鼻音	baigner	J
ŋ	N	软顎鼻音	parking	k
p	p	清双唇塞音	pomme	p
ʁ	R	浊小舌擦音	amoureux	k
s	s	清齿龈摩擦音	santé	s
ʃ	S	清龈后擦音	chat	S
t	t	清齿龈塞音	téléphone	t
v	v	浊唇齿擦音	vrai	f
w	w	浊圆唇软顎近音	soir	u
z	z	浊齿龈擦音	raison	s

IPA	X-SAMPA	描述	示例	语音视位
ʒ	Z	浊龈后擦音	aubergine	S
元音				
ø	2	半闭前圆唇元音	deux	o
œ	9	半开前圆唇元音	neuf	O
œ̃	9~	半开前圆唇鼻元音	brun	O
ə	@	中央元音	je	@
a	a	开前不圆唇元音	table	a
ɑ̃	A~	开后不圆唇鼻元音	camembert	a
e	e	半闭前不圆唇元音	marché	e
ɛ	E	中前不圆唇开元音	neige	E
ɛ̃	E~	半开前不圆唇鼻元音	sapin	E
i	i	闭前不圆唇元音	mille	i
o	o	半闭后圆唇元音	hôpital	o
ɔ	O	半开后圆唇元音	homme	O
ɔ̃	O~	半开后圆唇鼻元音	bon	O
u	u	闭后圆唇元音	sous	u
y	y	闭前圆唇元音	dur	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	

IPA	X-SAMPA	描述	示例	语音视位
.	.	音节划分	A.la.ba.ma	

## 法语 (比利时) (fr-BE)

下表列出了 Amazon Polly 支持的比利时法语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	boire	p
d	d	浊齿龈塞音	madame	t
f	f	清唇齿擦音	femme	f
g	g	浊软顎塞音	grand	k
ɥ	H	唇硬顎近音	bruit	u
j	j	硬顎近音	meilleur	i
k	k	清软顎塞音	quatre	k
l	l	齿龈边音	malade	t
m	m	双唇鼻音	maison	p
n	n	齿龈鼻音	astronome	t
ɲ	J	硬顎鼻音	baigner	J
ŋ	N	软顎鼻音	parking	k
p	p	清双唇塞音	pomme	p

IPA	X-SAMPA	描述	示例	语音视位
ʁ	R	浊小舌擦音	amoureux	k
s	s	清齿龈摩擦音	santé	s
ʃ	S	清龈后擦音	chat	S
t	t	清齿龈塞音	téléphone	t
v	v	浊唇齿擦音	vrai	f
w	w	浊圆唇软顎近音	soir	u
z	z	浊齿龈擦音	raison	s
ʒ	Z	浊龈后擦音	aubergine	S
元音				
ø	2	半闭前圆唇元音	deux	o
œ	9	半开前圆唇元音	neuf	O
œ̃	9~	半开前圆唇鼻元音	brun	O
ə	@	中央元音	je	@
a	a	开前不圆唇元音	table	a
ɑ̃	A~	开后不圆唇鼻元音	camembert	a
e	e	半闭前不圆唇元音	marché	e
ɛ	E	中前不圆唇开元音	neige	E
ɛ̃	E~	半开前不圆唇鼻元音	sapin	E
i	i	闭前不圆唇元音	mille	i
o	o	半闭后圆唇元音	hôpital	o

IPA	X-SAMPA	描述	示例	语音视位
ɔ	O	半开后圆唇元音	homme	O
õ	O~	半开后圆唇鼻元音	bon	O
u	u	闭后圆唇元音	sous	u
y	y	闭前圆唇元音	dur	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 法语 ( 加拿大 ) (fr-CA)

下表列出了 Amazon Polly 支持的加拿大法语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	boire	p
d	d	浊齿龈塞音	madame	t
f	f	清唇齿擦音	femme	f
g	g	浊软顎塞音	grand	k
ɥ	H	唇硬顎近音	bruit	u
j	j	硬顎近音	meilleur	i

IPA	X-SAMPA	描述	示例	语音视位
k	k	清软顎塞音	quatre	k
l	l	齿龈边音	malade	t
m	m	双唇鼻音	maison	p
n	n	齿龈鼻音	astronome	t
ɲ	J	硬顎鼻音	baigner	J
ŋ	N	软顎鼻音	parking	k
p	p	清双唇塞音	pomme	p
ʁ	R	浊小舌擦音	amoureux	k
s	s	清齿龈摩擦音	santé	s
ʃ	S	清龈后擦音	chat	S
t	t	清齿龈塞音	téléphone	t
v	v	浊唇齿擦音	vrai	f
w	w	浊圆唇软顎近音	soir	u
z	z	浊齿龈擦音	raison	s
ʒ	Z	浊龈后擦音	aubergine	S
元音				
ø	2	半闭前圆唇元音	deux	o
œ	9	半开前圆唇元音	neuf	O
œ̃	9~	半开前圆唇鼻元音	brun	O
ə	@	中央元音	je	@

IPA	X-SAMPA	描述	示例	语音视位
a	a	开前不圆唇元音	table	a
ã	A~	开后不圆唇鼻元音	camembert	a
e	e	半闭前不圆唇元音	marché	e
ɛ	E	中前不圆唇开元音	neige	E
ɛ̃	E~	半开前不圆唇鼻元音	sapin	E
i	i	闭前不圆唇元音	mille	i
o	o	半闭后圆唇元音	hôpital	o
ɔ	O	半开后圆唇元音	homme	O
õ	O~	半开后圆唇鼻元音	bon	O
u	u	闭后圆唇元音	sous	u
y	y	闭前圆唇元音	dur	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 德语 (de-DE)

下表列出了 Amazon Polly 支持的德语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

## 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
ʔ	ʔ	声门闭锁音		
b	b	浊双唇塞音	Bier	p
d	d	浊齿龈塞音	Dach	t
ç	C	清硬腭擦音	ich	k
ɖʒ	dZ	浊龈后塞擦音	Dschungel	S
f	f	清唇齿擦音	Vogel	f
g	g	浊软腭塞音	Gabel	k
h	h	清喉擦音	Haus	k
j	j	清喉擦音	jemand	i
k	k	清软腭塞音	Kleid	k
l	l	齿龈边音	Loch	t
m	m	双唇鼻音	Milch	p
n	n	齿龈鼻音	Natur	t
ŋ	N	软腭鼻音	klingen	k
p	p	清双唇塞音	Park	p
ɸf	pf	清唇齿塞擦音	Apfel	
ʀ	R	小舌颤音	Regen	
s	s	清齿龈摩擦音	Messer	s
ʃ	S	清龈后擦音	Fischer	S

IPA	X-SAMPA	描述	示例	语音视位
t	t	清齿龈塞音	Topf	T
$\text{t}^{\text{h}}$	Ts	清齿龈塞擦音	Zahl	
$\text{t}^{\text{f}}$	tS	清龈后塞擦音	deutsch	S
v	v	浊唇齿擦音	Wasser	f
x	x	清软顎擦音	kochen	k
z	z	浊齿龈擦音	See	s
ʒ	Z	浊龈后擦音	Orange	S
<b>元音</b>				
ø:	2:	长半闭前圆唇元音	böse	o
e	6	次开央元音	besser	a
ɐ	6_^	非音节次开央元音	Klar	a
œ	9	半开前圆唇元音	können	O
ə	@	中央元音	Rede	@
a	a	开前不圆唇元音	Salz	a
a:	a:	长开前不圆唇元音	Sahne	a
aɪ	al	双元音	nein	a
aʊ	aU	双元音	Augen	a
ã	A~	开后不圆唇鼻元音	Restaurant	a
e:	e:	长半闭前不圆唇元音	Rede	e
ɛ	E	中前不圆唇开元音	Keller	E

IPA	X-SAMPA	描述	示例	语音视位
ɛ̃	E~	半开前不圆唇鼻元音	Terrain	E
i:	i:	长闭前不圆唇元音	Lied	i
ɪ	我	近闭前不圆唇元音	bitte	i
o:	o:	长半闭后圆唇元音	Kohl	o
ɔ	O	半开后圆唇元音	Koffer	O
õ	O~	半开后圆唇鼻元音	Annonce	O
ɔʏ	OY	双元音	neu	O
u:	u:	长闭后圆唇元音	Bruder	u
ʊ	U	次闭次后圆唇元音	Wunder	u
y:	y:	长闭前圆唇元音	kühl	u
ʏ	Y	次闭次前圆唇元音	Küche	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 德语 ( 奥地利 ) (de-AT)

下表列出了 Amazon Polly 支持的奥地利德语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

## 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
ʔ	ʔ	声门闭锁音		
b	b	浊双唇塞音	Bier	p
d	d	浊齿龈塞音	Dach	t
ç	C	清硬腭擦音	ich	k
ɖʒ	dZ	浊龈后塞擦音	Dschungel	S
f	f	清唇齿擦音	Vogel	f
g	g	浊软腭塞音	Gabel	k
h	h	清喉擦音	Haus	k
j	j	清喉擦音	jemand	i
k	k	清软腭塞音	Kleid	k
l	l	齿龈边音	Loch	t
m	m	双唇鼻音	Milch	p
n	n	齿龈鼻音	Natur	t
ŋ	N	软腭鼻音	klingen	k
p	p	清双唇塞音	Park	p
ɸf	pf	清唇齿塞擦音	Apfel	
ʀ	R	小舌颤音	Regen	
s	s	清齿龈摩擦音	Messer	s
ʃ	S	清龈后擦音	Fischer	S

IPA	X-SAMPA	描述	示例	语音视位
t	t	清齿龈塞音	Topf	T
ʈs	Ts	清齿龈塞擦音	Zahl	
ʈʃ	tS	清龈后塞擦音	deutsch	S
v	v	浊唇齿擦音	Wasser	f
x	x	清软顎擦音	kochen	k
z	z	浊齿龈擦音	See	s
ʒ	Z	浊龈后擦音	Orange	S
<b>元音</b>				
ø:	2:	长半闭前圆唇元音	böse	o
e	6	次开央元音	besser	a
ɐ	6_^	非音节次开央元音	Klar	a
œ	9	半开前圆唇元音	können	O
ə	@	中央元音	Rede	@
a	a	开前不圆唇元音	Salz	a
a:	a:	长开前不圆唇元音	Sahne	a
aɪ	al	双元音	nein	a
aʊ	aU	双元音	Augen	a
ã	A~	开后不圆唇鼻元音	Restaurant	a
e:	e:	长半闭前不圆唇元音	Rede	e
ɛ	E	中前不圆唇开元音	Keller	E

IPA	X-SAMPA	描述	示例	语音视位
ɛ̃	E~	半开前不圆唇鼻元音	Terrain	E
i:	i:	长闭前不圆唇元音	Lied	i
ɪ	我	近闭近前不圆唇元音	bitte	i
o:	o:	长半闭后圆唇元音	Kohl	o
ɔ	O	半开后圆唇元音	Koffer	O
õ	O~	半开后圆唇鼻元音	Annonce	O
ɔʏ	OY	双元音	neu	O
u:	u:	长闭后圆唇元音	Bruder	u
ʊ	U	次闭次后圆唇元音	Wunder	u
y:	y:	长闭前圆唇元音	kühl	u
ʏ	Y	次闭次前圆唇元音	Küche	u
<b>其他符号</b>				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 德语 ( 瑞士标准 ) ( de-CH )

下表列出了 Amazon Polly 支持的德语 ( 瑞士标准 ) 语音的国际音标 ( IPA ) 音素、拓展音标字母评估法 ( X-SAMPA ) 符号和对应的语音视位。

## 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
ʔ	ʔ	声门闭锁音		
b	b	浊双唇塞音	Bier	p
d	d	浊齿龈塞音	Dach	t
ç	C	清硬腭擦音	ich	k
ɖʒ	dZ	浊龈后塞擦音	Dschungel	S
f	f	清唇齿擦音	Vogel	f
g	g	浊软腭塞音	Gabel	k
h	h	清喉擦音	Haus	k
j	j	清喉擦音	jemand	i
k	k	清软腭塞音	Kleid	k
l	l	齿龈边音	Loch	t
m	m	双唇鼻音	Milch	p
n	n	齿龈鼻音	Natur	t
ŋ	N	软腭鼻音	klingen	k
p	p	清双唇塞音	Park	p
pf	pf	清唇齿塞擦音	Apfel	
ʀ	R	小舌颤音	Regen	
s	s	清齿龈摩擦音	Messer	s
ʃ	S	清龈后擦音	Fischer	S

IPA	X-SAMPA	描述	示例	语音视位
t	t	清齿龈塞音	Topf	T
$\text{t}^{\text{h}}$	Ts	清齿龈塞擦音	Zahl	
$\text{t}^{\text{f}}$	tS	清龈后塞擦音	deutsch	S
v	v	浊唇齿擦音	Wasser	f
x	x	清软顎擦音	kochen	k
z	z	浊齿龈擦音	See	s
ʒ	Z	浊龈后擦音	Orange	S
元音				
ø:	2:	长半闭前圆唇元音	böse	o
e	6	次开央元音	besser	a
ɐ	6_^	非音节次开央元音	Klar	a
œ	9	半开前圆唇元音	können	O
ə	@	中央元音	Rede	@
a	a	开前不圆唇元音	Salz	a
a:	a:	长开前不圆唇元音	Sahne	a
aɪ	al	双元音	nein	a
aʊ	aU	双元音	Augen	a
ã	A~	开后不圆唇鼻元音	Restaurant	a
e:	e:	长半闭前不圆唇元音	Rede	e
ɛ	E	中前不圆唇开元音	Keller	E

IPA	X-SAMPA	描述	示例	语音视位
ɛ̃	E~	半开前不圆唇鼻元音	Terrain	E
i:	i:	长闭前不圆唇元音	Lied	i
ɪ	我	近闭近前不圆唇元音	bitte	i
o:	o:	长半闭后圆唇元音	Kohl	o
ɔ	O	半开后圆唇元音	Koffer	O
õ	O~	半开后圆唇鼻元音	Annonce	O
ɔʏ	OY	双元音	neu	O
u:	u:	长闭后圆唇元音	Bruder	u
ʊ	U	次闭次后圆唇元音	Wunder	u
y:	y:	长闭前圆唇元音	kühl	u
ʏ	Y	次闭次前圆唇元音	Küche	u
<b>其他符号</b>				
'	“	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 印地语 (hi-IN)

下表列出了 Amazon Polly 支持的印地语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和音素的声音类型。

有关与印地语结合使用的其他发音，请参阅 [英语 \(印度\) \(en-IN\)](#)。

## 音素/语音视位表

IPA	X-SAMPA	描述	示例
辅音			
p <sup>h</sup>	p_h	送气清双唇塞音	फूल (phool)
b <sup>h</sup>	b_h	送气浊双唇塞音	भारी (bhaari)
t̪	t_d	清齿塞音	तापमान (taapmaan)
t̪ <sup>h</sup>	t_d_h	送气清齿塞音	थोड़ा (thoda)
d̪	d_d	浊齿塞音	दिल्ली (dilli)
d̪ <sup>h</sup>	d_d_h	送气浊齿塞音	धोबी (dhobi)
t̪	t̪`	清卷舌塞音	कटोरा (katora)
t̪ <sup>h</sup>	t̪`_h	送气清卷舌塞音	ठंड (thand)
d̪	d̪`	浊卷舌塞音	डर (darr)
d̪ <sup>h</sup>	d̪`_h	送气浊卷舌塞音	ढाल (dhal)
tʃ <sup>h</sup>	tS_h	送气清硬颚塞擦音	छाल (chaal)
dʒ <sup>h</sup>	dZ_h	送气浊硬颚塞擦音	झाल (jhaal)
k <sup>h</sup>	k_h	送气清软颚塞音	खान (khan)
g <sup>h</sup>	g_h	送气浊软颚塞音	घान (ghaan)
ŋ	n̠`	卷舌鼻音	क्षण (kshan)
r	ɽ	齿龈闪音	राम (ram)
ɽ	r̠`	平卷舌闪音	बड़ा (bada)
ɽ <sup>h</sup>	r̠`_h	送气浊卷舌闪音	बढ़ी (barhi)
ʋ	v\	双唇近音	वसूल (wasool)

IPA	X-SAMPA	描述	示例
元音			
ə	@_o	中央元音	अच्छा (achhaa)
ẽ	@~	鼻腔中央元音	हँसना (hansnaa)
a	A_o	开前不圆唇元音	आग (aag)
ã	A~	鼻腔开前不圆唇元音	घड़ियाँ (ghariyaan)
ɪ	l_o	近闭近前不圆唇元音	इक्कीस (ikkees)
ĩ	l~	鼻腔近闭近前不圆唇元音	संचिाई (sinchai)
i	i_o	闭前不圆唇元音	बिल्ली (billee)
ĩ	i~	鼻腔闭前不圆唇元音	नही (nahin)
ʊ	U_o	次闭次后圆唇元音	उलूल (ullu)
ũ	U~	鼻腔近闭近后圆唇元音	मुँह (munh)
u	u_o	闭后圆唇元音	फूल (phool)
ũ	u~	鼻腔近后圆唇元音	ऊँट (oont)
ɔ	O_o	半开后圆唇元音	कौन (kaun)
õ	O~	鼻腔开中后圆唇元音	भौ (bhaun)
o	o	半闭后圆唇元音	सोना (sona)
õ	o~	鼻腔半闭后圆唇元音	क्यो (kyon)
ɛ	E_o	中前不圆唇开元音	पैसा (paisa)
ẽ	E~	鼻腔半开前不圆唇元音	मैं (main)
e	e	半闭前不圆唇元音	एक (ek)

IPA	X-SAMPA	描述	示例
ẽ	e~	鼻腔半闭前不圆唇元音	कतिबे (kitabein)

## 冰岛语 (is-IS)

下表列出了 Amazon Polly 支持的冰岛语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	grasbakkanum	0
c	c	清硬腭塞音	pakkin	k
c <sup>h</sup>	c_h	送气清硬腭塞音	anarkistai	k
ç	C	清硬腭擦音	héðan	k
d	d	浊齿龈塞音	bónði	t
ð	D	浊齿擦音	borð	T
f	f	清唇齿擦音	duft	f
g	g	浊软腭塞音	holgóma	k
ɣ	G	浊软腭擦音	hugur	k
h	h	清喉擦音	heili	k
j	j	硬腭近音	jökull	i
k <sup>h</sup>	k_h	送气清软腭塞音	ósköpunum	k
l	l	齿龈边音	gólf	t

IPA	X-SAMPA	描述	示例	语音视位
l	l_0	清齿龈边音	fólk	t
m	m	双唇鼻音	september	p
m̥	m_0	清双唇鼻音	kompa	p
n	n	齿龈鼻音	númer	t
n̥	n_0	清齿龈鼻音	pöntun	t
ɲ	J	硬腭鼻音	pælingar	J
ŋ	N	软腭鼻音	söngvarann	k
ŋ̥	N_0	清软腭鼻音	frænka	k
p <sup>h</sup>	p_h	送气清双唇塞音	afplánun	p
r	r	齿龈颤音	afskrifta	r
r̥	r_0	清齿龈颤音	andvörpum	r
s	s	清齿龈摩擦音	baðhús	s
t <sup>h</sup>	t_h	送气清齿龈塞音	tanki	t
θ	T	清齿擦音	þeldökki	T
v	v	浊唇齿擦音	silfur	f
w	w	浊圆唇软腭近音		u
x	x	清软腭擦音	samfélags	k
<b>元音</b>				
œ	9	半开前圆唇元音	þröskuldinum	O
œ:	9:	长半开前圆唇元音	tvö	O

IPA	X-SAMPA	描述	示例	语音视位
a	a	开前不圆唇元音	nefna	a
a:	a:	长开前不圆唇元音	fara	a
au	au	双元音	átta	a
au:	au:	双元音	átján	a
ɛ	E	中前不圆唇开元音	kennari	E
ɛ:	E:	长中前不圆唇开元音	dreka	E
i	i	闭前不圆唇元音	Gúliver	i
i:	i:	长闭前不圆唇元音	þír	i
ɪ	我	近闭近前不圆唇元音	samspil	i
ɪ:	l:	长近闭近前不圆唇元音	stig	i
ɔ	O	半开后圆唇元音	regndropar	O
ɔ:	O:	长半开后圆唇元音	ullarbolur	O
ou	Ou	双元音	tólf	O
ou:	Ou:	双元音	fjórir	O
u	u	闭后圆唇元音	stúlkan	u
u:	u:	长闭后圆唇元音	frú	u
ʏ	Y	次闭次前圆唇元音	tíu	u
ʏ:	Y	长次闭次前圆唇元音	gruninn	u

IPA	X-SAMPA	描述	示例	语音视位
其他符号				
ˈ	"	主重音	Alabama	
ˌ	%	辅重音	Alabama	
·	.	音节划分	A.la.ba.ma	

## 意大利语 (it-IT)

下表列出了 Amazon Polly 支持的意大利语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	bacca	p
d	d	浊齿龈塞音	dama	t
ɸdz	dz	浊齿龈塞擦音	zero	s
ɸdʒ	dʒ	浊龈后塞擦音	giro	S
f	f	清唇齿擦音	famiglia	f
g	g	浊软腭塞音	gatto	k
h	h	清喉擦音	horror	k
j	j	硬腭近音	dieci	i
k	k	清软腭塞音	campo	k
l	l	齿龈边音	lido	t

IPA	X-SAMPA	描述	示例	语音视位
ʎ	L	硬颚边音	aglio	J
m	m	双唇鼻音	mille	p
n	n	齿龈鼻音	nove	t
ɲ	J	硬颚鼻音	lasagne	J
p	p	清双唇塞音	pizza	p
r	r	齿龈颤音	risata	r
s	s	清齿龈摩擦音	sei	s
ʃ	S	清龈后擦音	scienza	S
t	t	清齿龈塞音	tavola	t
ʦ	ts	清齿龈塞擦音	forza	s
ʧ	tS	清龈后塞擦音	cielo	S
v	v	浊唇齿擦音	venti	f
w	w	浊圆唇软颚近音	quattro	u
z	z	浊齿龈擦音	bisogno	s
ʒ	Z	浊龈后擦音	bijou	S
元音				
a	a	开前不圆唇元音	arco	a
e	e	半闭前不圆唇元音	tre	e
ɛ	E	中前不圆唇开元音	ettaro	E
i	i	闭前不圆唇元音	impero	i

IPA	X-SAMPA	描述	示例	语音视位
o	o	半闭后圆唇元音	centoq	o
ɔ	O	半开后圆唇元音	otto	O
u	u	闭后圆唇元音	uno	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 日语 (ja-JP)

Amazon Polly 支持日语发音假名和读音假名字母。要让 Amazon Polly 使用此类字母语音发音，请使用音素 `alphabet="x-amazon-phonetic standard used"` 属性。

- `x-amazon-pron-kana` 表示使用发音假名。发音假名是用于语音转录的特殊片假名字符，可以对音高重音进行编码。
- `x-amazon-yomigana` 表示使用读音假名。读音假名可以是传统的片假名、平假名和拉丁字母，可以解读为 Hepburn 罗马拼音。

以下示例展示如何使用这些字母。

### 发音假名

```
<speak>
  ###<phoneme alphabet="x-amazon-pron-kana" ph="###'#">##</phoneme>###
</speak>
```

### 读音假名

```
<speak>
  ###<phoneme alphabet="x-amazon-yomigana" ph="####">##</phoneme>###
  ###<phoneme alphabet="x-amazon-yomigana" ph="####">##</phoneme>###
```

```
###<phoneme alphabet="x-amazon-yomigana" ph="Hirokazu">##</phoneme>###
</speak>
```

下表列出了 Amazon Polly 支持的日语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

IPA	X-SAMPA	描述	示例	语音视位
辅音				
r	4	齿龈闪音	練習, renshuu	t
ʔ	ʔ	声门闭锁音	あつっ、atsu'	
b	b	浊双唇塞音	舞踊, buyou	p
β	B	浊双唇擦音	ヴィンテー ジ, vinteeji	B
c	c	清硬腭塞音	ききょう, kikyou	k
ç	C	清硬腭擦音	人, hito	k
d	d	浊齿龈塞音	濁点, dakuten	t
ɽ	dz\	浊齿龈擦音	純, jun	J
g	g	浊软腭塞音	ご飯, gohan	k
h	h	清喉擦音	本, hon	k
j	j	硬腭近音	屋根, yane	i
ɟ	J\	浊硬腭塞音	行儀, gyougi	J
k	k	清软腭塞音	漢字, kanji	k
ʀ	l\	齿龈边闪音	釣り, tsuri	r
ʎ	lj	齿龈边闪音, 硬腭 近音	流行, ryuukou	r

IPA	X-SAMPA	描述	示例	语音视位
m	m	双唇鼻音	飯, meshi	p
n	n	齿龈鼻音	猫, neko	t
ɲ	J	硬腭鼻音	日本, nippon	J
ɳ	N\	小舌鼻音	缶, kan	k
p	p	清双唇塞音	パン, pan	p
ɸ	p\	清双唇擦音	福, huku	f
s	s	清齿龈摩擦音	層, sou	s
ɕ	s\	清龈腭擦音	書簡, shokan	J
t	t	清齿龈塞音	手紙, tegami	t
ʦ	ts	清齿龈塞擦音	釣り, tsuri	s
ʧ	ts\	清龈腭塞擦音	吉, kichi	J
w	w	浊圆唇软腭近音	電話, denwa	u
z	z	浊齿龈擦音	座敷, zashiki	s
元音				
ä:	a:_"	长开央不圆唇元音	羽蟻, haari	a
ä	a_"	开央不圆唇元音	仮名, kana	a
e:	e:_o	长中前不圆唇元音	学生, gakusei	@
e	e_o	中前不圆唇元音	歴, reki	@
i	i	闭前不圆唇元音	気, ki	i
i:	i:	长闭前不圆唇元音	詩歌, shiika	i

IPA	X-SAMPA	描述	示例	语音视位
ʊ	M	闭后不圆唇元音	運, un	i
ʊ:	M:	长闭后不圆唇元音	宗教, shuukyou	i
o:	o:_o	长中后圆唇元音	購読, koodoku	o
o	o_o	中后圆唇元音	読者, dokusha	o

## 韩语 (ko-KR)

下表列出了 Amazon Polly 支持的韩语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

IPA	X-SAMPA	描述	示例	语音视位
辅音				
k	k	清软顎塞音	강, [g]ang	k
k#	k_t	紧软顎塞音	깨, [kk]e	k
n	n	齿龈鼻音	남, [n]am	t
t	t	清齿龈塞音	도, [d]o	t
t#	t_t	紧齿龈塞音	때, [tt]e	t
r	4	齿龈闪音	사랑, sa[r]ang	t
l	l	齿龈边音	돌, do[l]	t
m	m	双唇鼻音	무, [m]u	p
p	p	清双唇塞音	봄, [b]om	p
p#	p_t	紧双唇塞音	빨, [pp]eol	p
s	s	清齿龈摩擦音	새, [s]e	s

IPA	X-SAMPA	描述	示例	语音视位
s#	s_t	紧齿龈摩擦音	씨, [ss]i	s
ŋ	N	软顎鼻音	방, ba[ng]	k
ʦ̥	ts\	清齿龈塞擦音	조, [j]o	J
ʦ̥#	ts\_t	紧齿龈塞擦音	찌, [jj]i	J
ʦ̥ʰ	ts\_h	送气清齿龈塞擦音	차, [ch]a	J
kʰ	k_h	送气清软顎塞音	코, [k]o	k
tʰ	t_h	送气清齿龈塞音	통, [t]ong	t
pʰ	p_h	送气清双唇塞音	패, [p]e	p
h	h	清喉擦音	힘, [h]im	k
j	j	硬顎近音	양, [y]ang	i
w	w	浊圆唇软顎近音	왕, [w]ang	u
ɥ	M\	圆唇软顎近音>	의, [wj]i	i
元音				
a	a	开前不圆唇元音	밥, b[a]b	a
ʌ	V	半开后不圆唇元音	정, j[eo]ng	E
ɛ	E	中前不圆唇开元音	배, b[e]	E
o	o	半闭后圆唇元音	노, n[o]	o
u	u	闭后圆唇元音	둘, d[u]l	u
ɯ	M	闭后不圆唇元音	은, [eu]n	i
i	i	闭前不圆唇元音	김, k[i]m	i

## 挪威语 (nb-NO)

下图列出了 Amazon Polly 支持的挪威语语音的全套国际音标 (IPA) 音素和拓展音标字母评估法 (X-SAMPA) 符号以及对应的语音视位。

IPA	X-SAMPA	描述	示例	语音视位
<b>辅音</b>				
r	ʀ	齿龈闪音	prøv	t
b	b	浊双唇塞音	labb	p
ç	C	清硬颚擦音	kino	k
d	d	浊齿龈塞音	ladd	t
ɖ	d`	浊卷舌塞音	verdi	t
f	f	清唇齿擦音	fot	f
g	g	浊软颚塞音	tagg	k
h	h	清喉擦音	ha	k
j	j	硬颚近音	gi	i
k	k	清软颚塞音	takk	k
l	l	齿龈边音	fall、ball	t
ɭ	l`	卷舌边音	ærlig	t
m	m	双唇鼻音	lam	p
n	n	齿龈鼻音	vann	t
ɳ	n`	卷舌鼻音	garn	t
ŋ	N	软颚鼻音	sang	k

IPA	X-SAMPA	描述	示例	语音视位
p	p	清双唇塞音	hopp	p
s	s	清齿龈摩擦音	lass	s
ʃ	s`	清卷舌擦音	års	S
ʃ	S	清龈后擦音	skyt	S
t	t	清齿龈塞音	lat	t
t̥	t`	清卷舌塞音	hardt	t
ʋ	v\	唇齿近音	vin	f
w	w	浊圆唇软腭近音	will	x
<b>元音</b>				
ø:	2:	长半闭前圆唇元音	søt	o
œ	9	半开前圆唇元音	søtt	O
ə	@	中央元音	ape	@
æ:	{:	长次开前不圆唇元音	vær	a
ʊ	}	闭央圆唇元音	lund	u
ʊ:	}::	长闭央圆唇元音	lun	u
æ	{	次开前不圆唇元音	vært	a
ɑ	A	开后不圆唇元音	hatt	a
ɑ:	A:	长开后不圆唇元音	hat	a
e:	e:	长半闭前不圆唇元音	sen	e

IPA	X-SAMPA	描述	示例	语音视位
ɛ	E	中前不圆唇开元音	send	E
i:	i:	长闭前不圆唇元音	vin	i
ɪ	我	近闭近前不圆唇元音	vind	i
o:	o:	长半闭后圆唇元音	våt	o
ɔ	O	半开后圆唇元音	vått	O
u:	u:	长闭后圆唇元音	bok	u
ʊ	U	次闭次后圆唇元音	bukk	u
y:	y:	长闭前圆唇元音	lyn	u
ɤ	Y	次闭次前圆唇元音	lynne	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 波兰语 (pl-PL)

下表列出了 Amazon Polly 支持的波兰语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				

IPA	X-SAMPA	描述	示例	语音视位
b	b	浊双唇塞音	bobas、belka	p
d	d	浊齿龈塞音	dar、do	t
ḏ	dz	浊齿龈塞擦音	dzwon、wizowie	s
ḏ̥	dz\	浊龈腭塞擦音	dźwięk	J
ḏ̥̥̥	dz`	浊卷舌塞擦音	dżem、dżungla	S
f	f	清唇齿擦音	furtka、film	f
g	g	浊软腭塞音	gazeta、waga	k
h	h	清喉擦音	chleb、handel	k
j	j	硬腭近音	jak、maja	i
k	k	清软腭塞音	kura、marek	k
l	l	齿龈边音	lipa、alicja	t
m	m	双唇鼻音	matka、molo	p
n	n	齿龈鼻音	norka	t
ɲ	J	硬腭鼻音	koń、toruń	J
p	p	清双唇塞音	pora、stop	p
r	r	齿龈颤音	rok、park	r
s	s	清齿龈摩擦音	sum、pas	s
ʃ	s\	清龈腭擦音	śruba、śnieg	J
ʂ	s`	清卷舌擦音	szum、masz	S
t	t	清齿龈塞音	tok、stół	t

IPA	X-SAMPA	描述	示例	语音视位
$\text{t}^{\text{h}}\text{s}$	ts	清齿龈塞擦音	car、co	s
$\text{t}^{\text{h}}\text{ɕ}$	ts\	清龈腭塞擦音	ćma、mieć	J
$\text{t}^{\text{h}}\text{ʂ}$	ts`	清卷舌塞擦音	czas、raczej	S
v	v	浊唇齿擦音	worek、mewa	f
w	w	浊圆唇软腭近音	łaska、mało	u
z	z	浊齿龈擦音	zero	s
ʐ	z\	浊龈腭擦音	źrebię、bieliznie	J
ʐ̥	z`	浊卷舌擦音	zar、żona	S
<b>元音</b>				
a	a	开前不圆唇元音	ja	a
ɛ	E	中前不圆唇开元音	echo	E
ɛ̃	E~	半开前不圆唇鼻元音	węże	E
i	i	闭前不圆唇元音	ile	i
ɔ	O	半开后圆唇元音	oczy	O
ɔ̃	O~	半开后圆唇鼻元音	wąż	O
u	u	闭后圆唇元音	uczta	u
ɨ	ɨ	闭央不圆唇元音	byk	i
<b>其他符号</b>				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	

IPA	X-SAMPA	描述	示例	语音视位
.	.	音节划分	A.la.ba.ma	

## 葡萄牙语 (pt-PT)

下表列出了 Amazon Polly 支持的葡萄牙语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
<b>辅音</b>				
r	4	齿龈闪音	pira	t
b	b	浊双唇塞音	dato	p
d	d	浊齿龈塞音	dato	t
f	f	清唇齿擦音	facto	f
g	g	浊软顎塞音	gato	k
j	j	硬顎近音	paraguay	i
k	k	清软顎塞音	cacto	k
l	l	齿龈边音	galo	t
ʎ	L	硬顎边音	galho	J
m	m	双唇鼻音	mato	p
n	n	齿龈鼻音	nato	t
ɲ	J	硬顎鼻音	pinha	J
p	p	清双唇塞音	pato	p

IPA	X-SAMPA	描述	示例	语音视位
r	R\	小舌颤音	barroso	k
s	s	清齿龈摩擦音	saca	s
ʃ	S	清龈后擦音	chato	S
t	t	清齿龈塞音	tacto	t
v	v	浊唇齿擦音	vaca	f
w	w	浊圆唇软颚近音	mau	u
z	z	浊齿龈擦音	zaca	s
ʒ	Z	浊龈后擦音	jacto	S
元音				
a	a	开前不圆唇元音	parto	a
ã	a~	开前不圆唇鼻元音	pega	a
e	e	半闭前不圆唇元音	pega	e
环	环~	半闭前不圆唇鼻元音	movem	e
ɛ	E	中前不圆唇开元音	café	E
i	i	闭前不圆唇元音	lingueta	i
ĩ	i~	闭前不圆唇鼻元音	cinto	i
o	o	半闭后圆唇元音	poder	o
õ	o~	半闭后圆唇鼻元音	compra	o
ɔ	O	半开后圆唇元音	cotó	O
u	u	闭后圆唇元音	fui	u

IPA	X-SAMPA	描述	示例	语音视位
ũ	u~	闭后圆唇鼻元音	sunto	u
<b>其他符号</b>				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 葡萄牙语 ( 巴西 ) (pt-BR)

下表列出了 Amazon Polly 支持的巴西葡萄牙语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
<b>辅音</b>				
r	4	齿龈闪音	pira	t
b	b	浊双唇塞音	bato	p
d	d	浊齿龈塞音	dato	t
ɾ̃	dZ	浊龈后塞擦音	idade	S
f	f	清唇齿擦音	facto	f
g	g	浊软顎塞音	gato	k
j	j	硬顎近音	paraguay	i
k	k	清软顎塞音	cacto	k
l	l	齿龈边音	galo	t

IPA	X-SAMPA	描述	示例	语音视位
ʎ	L	硬颚边音	galho	J
m	m	双唇鼻音	mato	p
n	n	齿龈鼻音	nato	t
ɲ	J	硬颚鼻音	pinha	J
p	p	清双唇塞音	pato	p
s	s	清齿龈摩擦音	saca	s
ʃ	S	清龈后擦音	chato	S
t	t	清齿龈塞音	tacto	t
tʃ	tS	清龈后塞擦音	noite	S
v	v	浊唇齿擦音	vaca	f
w	w	浊圆唇软颚近音	mau	u
x	X 形	清小舌擦音	carro	k
z	z	浊齿龈擦音	zaca	s
ʒ	Z	浊龈后擦音	jacto	S
元音				
a	a	开前不圆唇元音	parto	a
ã	a~	开前不圆唇鼻元音	pensamos	a
e	e	半闭前不圆唇元音	pega	e
环	环~	半闭前不圆唇鼻元音	movem	e
ɛ	E	中前不圆唇开元音	café	E

IPA	X-SAMPA	描述	示例	语音视位
i	i	闭前不圆唇元音	lingueta	i
ĩ	i~	闭前不圆唇鼻元音	cinto	i
o	o	半闭后圆唇元音	poder	o
õ	o~	半闭后圆唇鼻元音	compra	o
ɔ	O	半开后圆唇元音	cotó	O
u	u	闭后圆唇元音	fui	u
ũ	u~	闭后圆唇鼻元音	sunto	u
<b>其他符号</b>				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 罗马尼亚语 (ro-RO)

下表列出了 Amazon Polly 支持的罗马尼亚语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
<b>辅音</b>				
b	b	浊双唇塞音	bubă	p
d	d	浊齿龈塞音	după	t
ɖʒ	dZ	浊龈后塞擦音	george	S

IPA	X-SAMPA	描述	示例	语音视位
f	f	清唇齿擦音	afacere	f
g	g	浊软顎塞音	agri#	k
h	h	清喉擦音	harpă	k
j	j	硬顎近音	baie	i
k	k	清软顎塞音	co#	k
l	l	齿龈边音	lampa	t
m	m	双唇鼻音	mama	p
n	n	齿龈鼻音	nor	t
p	p	清双唇塞音	pilă	p
r	r	齿龈颤音	rampă	r
s	s	清齿龈摩擦音	soare	s
ʃ	S	清龈后擦音	ma#ină	S
t	t	清齿龈塞音	tata	t
ʦ	ts	清齿龈塞擦音	#ară	s
tʃ	tS	清龈后塞擦音	ceai	S
v	v	浊唇齿擦音	via#ă	f
w	w	浊圆唇软顎近音	beau	u
z	z	浊齿龈擦音	mozol	s
ʒ	Z	浊龈后擦音	joacă	S
元音				

IPA	X-SAMPA	描述	示例	语音视位
ə	@	中央元音	babă	@
a	a	开前不圆唇元音	casa	a
e	e	半闭前不圆唇元音	elan	e
ɛ	e_^	非音节半闭前不圆唇元音	beau	e
i	i	闭前不圆唇元音	mie	i
o	o	半闭后圆唇元音	oră	o
oa	o_^a	双元音	oare	o
u	u	闭后圆唇元音	unde	u
ɨ	1	闭央不圆唇元音	România	i
<b>其他符号</b>				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 俄语 (ru-RU)

下表列出了 Amazon Polly 支持的俄语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
<b>辅音</b>				

IPA	X-SAMPA	描述	示例	语音视位
b	b	浊双唇塞音	борт	p
b <sup>j</sup>	b'	腭化浊双唇塞音	бюро	p
d	d	浊齿龈塞音	дом	t
d <sup>j</sup>	d'	腭化浊齿龈塞音	дядя	t
f	f	清唇齿擦音	флаг	f
f <sup>j</sup>	f'	腭化清唇齿擦音	февраль	f
g	g	浊软顎塞音	нога	k
g <sup>j</sup>	g'	腭化浊软顎塞音	герой	k
j	j	硬顎近音	дизайн、ящик	i
k	k	清软顎塞音	кот	k
k <sup>j</sup>	k'	腭化清软顎塞音	кино	k
l	l	齿龈边音	лампа	t
l <sup>j</sup>	l'	腭化齿龈边音	лес	t
m	m	双唇鼻音	мама	p
m <sup>j</sup>	m'	腭化双唇鼻音	мяч	p
n	n	齿龈鼻音	нос	t
n <sup>j</sup>	n'	腭化齿龈鼻音	няня	t
p	p	清双唇塞音	папа	p
p <sup>j</sup>	p'	腭化清双唇塞音	перо	p
r	r	齿龈颤音	роза	r

IPA	X-SAMPA	描述	示例	语音视位
r <sup>j</sup>	r'	腭化齿龈颤音	рюмка	r
s	s	清齿龈摩擦音	сыр	s
s <sup>j</sup>	s'	腭化清齿龈擦音	сердце、русь	s
ʃ:	s\:	长清齿龈擦音	щека	J
ʃ	s`	清卷舌擦音	шум	S
t	t	清齿龈塞音	точка	t
t <sup>j</sup>	t'	腭化清齿龈塞音	тётя	t
ʈs	ts	清齿龈塞擦音	царь	s
ʈʃ	ts\	清齿龈塞擦音	час	J
v	v	浊唇齿擦音	вор	f
v <sup>j</sup>	v'	腭化浊唇齿擦音	верфь	f
x	x	清软腭擦音	хор	k
x <sup>j</sup>	x'	腭化清软腭擦音	химия	k
z	z	浊齿龈擦音	зуб	s
z <sup>j</sup>	z'	腭化浊齿龈擦音	зима	s
ʒ:	z\:	长浊齿龈擦音	уезжать	J
ʒ	z`	浊卷舌擦音	жена	S
元音				
ə	@	中央元音	канарейка	@
a	a	开前不圆唇元音	два、яблоко	a

IPA	X-SAMPA	描述	示例	语音视位
e	e	半闭前不圆唇元音	печь	e
ɛ	E	中前不圆唇开元音	это	E
i	i	闭前不圆唇元音	один、четыре	i
o	o	半闭后圆唇元音	кот	o
u	u	闭后圆唇元音	муж、вьюга	u
ɨ	ɨ	闭央不圆唇元音	мышь	ɨ

## 西班牙语 (es-ES)

下表列出了 Amazon Polly 支持的西班牙语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
<b>辅音</b>				
r	ɾ	齿龈闪音	pero、bravo、amor、	t
b	b	浊双唇塞音	bestia	p
β	B	浊双唇擦音	bebé	B
d	d	浊齿龈塞音	cuando	t
ð	D	浊齿擦音	arder	T
f	f	清唇齿擦音	fase、café	f
g	g	浊软腭塞音	gato、lengua、guerra	k
ɣ	G	浊软腭擦音	trigo、Argos	k

IPA	X-SAMPA	描述	示例	语音视位
j	j	硬颚近音	hacia、tierra、radio、	i
ɰ	ɰ	浊硬颚擦音	enhielar、sayo、inye syerba	J
k	k	清软颚塞音	caña、laca、quisimo	k
l	l	齿龈边音	lino、calor、princi pal	t
ʎ	ʎ	硬颚边音	llave、pollo	J
m	m	双唇鼻音	madre、comer、anfil	p
n	n	齿龈鼻音	nido、anillo、sin	t
ɲ	ɲ	硬颚鼻音	cabaña、ñoquis	J
ŋ	ŋ	软颚鼻音	cinco、venga	k
p	p	清双唇塞音	pozo、topo	p
r	r	齿龈颤音	perro、enrachado	r
s	s	清齿龈摩擦音	saco、casa、puertas	s
t	t	清齿龈塞音	tamiz、átomo	t
ʎ	tʂ	清龈后塞擦音	chubasco	S
θ	T	清齿擦音	cereza、zorro、lacer	T
w	w	浊圆唇软颚近音	fuego、fuimos、cuot	u
x	x	清软颚擦音	jamón、general、su je、reloj	k
z	z	浊齿龈擦音	rasgo、mismo	s
元音				

IPA	X-SAMPA	描述	示例	语音视位
a	a	开前不圆唇元音	tanque	a
e	e	半闭前不圆唇元音	peso	e
i	i	闭前不圆唇元音	cinco	i
o	o	半闭后圆唇元音	bosque	o
u	u	半闭前不圆唇元音	publicar	u
<b>其他符号</b>				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 西班牙语 ( 墨西哥 ) (es-MX)

下表列出了 Amazon Polly 支持的墨西哥西班牙语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
<b>辅音</b>				
r	4	齿龈闪音	pero、bravo、amor、	t
b	b	浊双唇塞音	bestia	p
β	B	浊双唇擦音	bebé	B
d	d	浊齿龈塞音	cuando	t
ð	D	浊齿擦音	arder	T

IPA	X-SAMPA	描述	示例	语音视位
f	f	清唇齿擦音	fase、café	f
g	g	浊软顎塞音	gato、lengua、guerra	k
ɣ	G	浊软顎擦音	trigo、Argos	k
j	j	硬顎近音	hacia、tierra、radio、	i
ɟ	ɟ\	浊硬顎擦音	enhielar、sayo、inye syerba	J
k	k	清软顎塞音	caña、laca、quisimo	k
l	l	齿龈边音	lino、calor、princi pal	t
m	m	双唇鼻音	madre、comer、anfil	p
n	n	齿龈鼻音	nido、anillo、sin	t
ɲ	J	硬顎鼻音	cabaña、ñoquis	J
ŋ	N	软顎鼻音	angosto、increíble	k
p	p	清双唇塞音	pozo、topo	p
r	r	齿龈颤音	perro、enrachado	r
s	s	清齿龈摩擦音	saco、casa、puertas	s
ʃ	S	清龈后擦音	show、flash	S
t	t	清齿龈塞音	tamiz、átomo	t
ɰ	tS	清龈后塞擦音	chubasco	S
w	w	浊圆唇软顎近音	fuego、fuimos、cuot	u
x	x	清软顎擦音	jamón、general、pe aje、reloj	k

IPA	X-SAMPA	描述	示例	语音视位
z	z	浊齿龈擦音	rasgo、 mismo	s
元音				
a	a	开央不圆唇元音	tanque	a
e	e	半闭前不圆唇元音	peso	e
i	i	闭前不圆唇元音	cinco	i
o	o	半闭后圆唇元音	bosque	o
u	u	闭后圆唇元音	publicar	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 西班牙语 ( 美国 ) (es-US)

下表列出了 Amazon Polly 支持的美国西班牙语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
r	4	齿龈闪音	pero、bravo、amor、	t
b	b	浊双唇塞音	bestia	p
β	B	浊双唇擦音	bebé	B

IPA	X-SAMPA	描述	示例	语音视位
d	d	浊齿龈塞音	cuando	t
ð	D	浊齿擦音	arder	T
f	f	清唇齿擦音	fase、café	f
g	g	浊软顎塞音	gato、lengua、guerra	k
ɣ	G	浊软顎擦音	trigo、Argos	k
j	j	硬顎近音	hacia、tierra、radio、	i
ɟ	ɟ\	浊硬顎擦音	enhielar、sayo、inye syerba	J
k	k	清软顎塞音	caña、laca、quisimo	k
l	l	齿龈边音	lino、calor、princi pal	t
m	m	双唇鼻音	madre、comer、anfil	p
n	n	齿龈鼻音	nido、anillo、sin	t
ɲ	J	硬顎鼻音	cabaña、ñoquis	J
ŋ	N	软顎鼻音	angosto、increíble	k
p	p	清双唇塞音	pozo、topo	p
r	r	齿龈颤音	perro、enrachado	r
s	s	清齿龈摩擦音	saco、casa、puertas	s
ʃ	S	清龈后擦音	show、flash	S
t	t	清齿龈塞音	tamiz、átomo	t
ʧ	tS	清龈后塞擦音	chubasco	S

IPA	X-SAMPA	描述	示例	语音视位
w	w	浊圆唇软顎近音	fuego、fuimos、cuot	u
x	x	清软顎擦音	jamón、general、pe aje、reloj	k
z	z	浊齿龈擦音	rasgo、mismo	s
元音				
a	a	开央不圆唇元音	tanque	a
e	e	半闭前不圆唇元音	peso	e
i	i	闭前不圆唇元音	cinco	i
o	o	半闭后圆唇元音	bosque	o
u	u	闭后圆唇元音	publicar	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 瑞典语 (sv-SE)

下表列出了 Amazon Polly 支持的瑞典语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				

IPA	X-SAMPA	描述	示例	语音视位
b	b	浊双唇塞音	bil	p
d	d	浊齿龈塞音	dal	t
ɖ	d`	浊卷舌塞音	bord	t
f	f	清唇齿擦音	fil	f
g	g	浊软顎塞音	gås	k
h	h	清喉擦音	hal	k
j	j	硬顎近音	jag	i
k	k	清软顎塞音	kal	k
l	l	齿龈边音	lös	t
ɭ	l`	卷舌边音	härlig	t
m	m	双唇鼻音	mil	p
n	n	齿龈鼻音	nålar	t
ɳ	n`	卷舌鼻音	barn	t
ŋ	N	软顎鼻音	ring	k
p	p	清双唇塞音	pil	p
r	r	齿龈颤音	ris	r
s	s	清齿龈摩擦音	sil	s
ʃ	s\	清龈顎擦音	tjock	J
ʂ	s`	清卷舌擦音	fors、schlager	S
t	t	清齿龈塞音	tal	t

IPA	X-SAMPA	描述	示例	语音视位
t	t`	清卷舌塞音	hjort	t
v	v	浊唇齿擦音	vår	f
w	w	浊圆唇软顎近音	aula、airways	u
ɲ	x\	清硬顎软顎擦音	sjuk	k
<b>元音</b>				
ø	2	半闭前圆唇元音	föll、förr	o
ø	2:	长半闭前圆唇元音	föl、nöt、för	o
e	8	半闭央圆唇元音	buss、full	o
ə	@	中央元音	pojken	@
ʉ:	}:	长闭央圆唇元音	hus、ful	u
a	a	开前不圆唇元音	hall、matt	a
æ	{	次开前不圆唇元音	herr	a
ɑ:	A:	长开后不圆唇元音	hal、mat	a
e:	e:	长半闭前不圆唇元音	vet、hel	e
ɛ	E	中前不圆唇开元音	vett、rätt、hetta、hä	E
ɛ:	E:	长中前不圆唇开元音	säl、häl、här	E:
i:	i:	长闭前不圆唇元音	vit、sil	i:
ɪ	我	近闭近前不圆唇元音	vitt、sill	i
o:	o:	长半闭后圆唇元音	hål、mål	o

IPA	X-SAMPA	描述	示例	语音视位
ɔ	O	半开后圆唇元音	háll、moll	O
u:	u:	长闭后圆唇元音	sol、bot	u
ʊ	U	次闭次后圆唇元音	bott	u
y	y	闭前圆唇元音	bytt	u
y:	y:	长闭前圆唇元音	syl、syl	u
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 土耳其语 (tr-TR)

下表列出了 Amazon Polly 支持的土耳其语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
ɾ	4	齿龈闪音	durum	t
ɾ̥	4_0_r	清摩擦齿龈闪音	bir	t
ɾ̣	4_r	摩擦齿龈闪音	raf	t
b	b	浊双唇塞音	raf	p
c	c	清硬腭塞音	kedı	k

IPA	X-SAMPA	描述	示例	语音视位
d	d	浊齿龈塞音	dede	t
ɖ	dZ	浊龈后塞擦音	cam	S
f	f	清唇齿擦音	fare	f
g	g	浊软顎塞音	galibi	k
h	h	清喉擦音	hasta	k
j	j	硬顎近音	yat	i
ʝ	J\	浊硬顎塞音	genç	J
k	k	清软顎塞音	akıl	k
l	l	齿龈边音	lale	t
ɮ	5	软顎化齿龈边音	labirent	t
m	m	双唇鼻音	maaş	p
n	n	齿龈鼻音	anı	t
p	p	清双唇塞音	ip	p
s	s	清齿龈摩擦音	ses	s
ʃ	S	清龈后擦音	aşı	S
t	t	清齿龈塞音	ütü	t
ɟ	tS	清龈后塞擦音	çaba	S
v	v	浊唇齿擦音	ekvator、kahveci、al	f
z	z	浊齿龈擦音	ver	s
ʒ	Z	浊龈后擦音	azık	S

IPA	X-SAMPA	描述	示例	语音视位
元音				
ø	2	半闭前圆唇元音	göl	0
œ	9	半开前圆唇元音	banliyö	O
a	a	开前不圆唇元音	kal	a
a:	a:	长开前不圆唇元音	davacı	a
æ	{	次开前不圆唇元音	özlem、güvenlik、gü	a
e	e	半闭前不圆唇元音	keçi	e
ɛ	E	中前不圆唇开元音	dede	E
i	i	闭前不圆唇元音	bir	i
i:	i:	长闭前不圆唇元音	izah	i
ɪ	我	近闭近前不圆唇元音	keçi	i
ɯ	M	闭后不圆唇元音	kıl	i
o	o	半闭后圆唇元音	kol	o
o:	o:	长半闭后圆唇元音	dolar	o
u	u	闭后圆唇元音	durum	u
u:	u:	长闭后圆唇元音	ruhum	u
ɯ	U	次闭次后圆唇元音	dolu	u
y	y	闭前圆唇元音	güvenlik	u
ɣ	Y	次闭次前圆唇元音	aşı	u
其他符号				

IPA	X-SAMPA	描述	示例	语音视位
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

## 威尔士语 (cy-GB)

下表列出了 Amazon Polly 支持的威尔士语语音的国际音标 (IPA) 音素、拓展音标字母评估法 (X-SAMPA) 符号和对应的语音视位。

### 音素/语音视位表

IPA	X-SAMPA	描述	示例	语音视位
辅音				
b	b	浊双唇塞音	baban	p
d	d	浊齿龈塞音	deg	t
ɖʒ	dZ	浊龈后塞擦音	garej	S
ð	D	浊齿擦音	deuddeg	T
f	f	清唇齿擦音	ffacs	f
g	g	浊软顎塞音	gadael	k
h	h	清喉擦音	haearn	k
j	j	硬顎近音	astudio	i
k	k	清软顎塞音	cant	k
l	l	齿龈边音	lan	t
ɬ	K	清齿龈边擦音	llan	t

IPA	X-SAMPA	描述	示例	语音视位
m	m	双唇鼻音	mae	p
ᵹ	m_0	清双唇鼻音	ymhen	p
n	n	齿龈鼻音	naw	t
ɳ	n_0	清齿龈鼻音	anhawster	t
ŋ	N	软顎鼻音	argyfwng	k
ɲ	N_0	清软顎鼻音	anghenion	k
p	p	清双唇塞音	pump	p
r	r	齿龈颤音	rhoi	r
ɾ	r_0	清齿龈颤音	garw	r
s	s	清齿龈摩擦音	saith	s
ʃ	S	清龈后擦音	siawns	S
t	t	清齿龈塞音	tegan	t
t͡ʃ	tS	清龈后塞擦音	cytsain	S
θ	T	清齿擦音	aberth	T
v	v	浊唇齿擦音	prawf	f
w	w	浊圆唇软顎近音	rhagweld	u
x	X 形	清小舌擦音	chwech	k
z	z	浊齿龈擦音	aids	s
ʒ	Z	浊龈后擦音	rouge	S
元音				

IPA	X-SAMPA	描述	示例	语音视位
ə	@	中央元音	ychwanega	@
a	a	开前不圆唇元音	acen	a
ai	ai	双元音	dau	a
au	au	双元音	awdur	a
ɑ:	A:	长开后不圆唇元音	mab	a
ɑ:i	A:1	双元音	aelod	a
e:	e:	长半闭前不圆唇元音	peth	e
ɛ	E	中前不圆唇开元音	pedwar	E
ɛi	Ei	双元音	beic	E
i:	i:	长闭前不圆唇元音	tri	i
ɪ	我	近闭近前不圆唇元音	miliwn	i
ɪu	1u	双元音	unigryw	i
o:	o:	长半闭后圆唇元音	oddi	o
ɔ	O	半开后圆唇元音	oddieithr	O
ɔi	Oi	双元音	troi	O
ɔu	Ou	双元音	rownd	O
u:	u:	长闭后圆唇元音	cwch	u
ʊ	U	次闭次后圆唇元音	acwstig	u
ʊi	Ui	双元音	wyth	u

IPA	X-SAMPA	描述	示例	语音视位
其他符号				
'	"	主重音	Alabama	
,	%	辅重音	Alabama	
.	.	音节划分	A.la.ba.ma	

# Amazon Polly 语音引擎

Amazon Polly 具有四种语音引擎，可将输入文本转化为逼真的语音。其中包括：生成式Long-form、神经和标准型。要使用 Amazon Polly 语音，请选择引擎和语音合成 API 操作。然后提供输入文本供引擎合成，然后选择音频输出格式。在给定这些输入的情况下，Amazon Polly 将提供的文本合成为高质量语音音频流。

以下各个部分包括有关 Amazon Polly 提供的语音引擎的详细信息。

## 主题

- [生成式语音](#)
- [Long-form 声音](#)
- [神经语音](#)
- [标准语音](#)
- [选择语音引擎](#)

## 生成式语音

Amazon Polly 的生成式文本转语音 ( TTS ) 引擎提供了最像人类、最具情感参与度和自适应能力的对话式语音，可供用户通过 Amazon Polly 控制台使用。

生成式引擎是迄今为止最大的 Amazon Polly TTS 模型。该引擎部署了一个包含十亿参数的转换器，用于将原始文本转换为语音代码，然后部署基于卷积的解码器，该解码器以可流式传输的增量方式将这些语音代码转换为波形。这种方法展示了大型语言模型 ( LLM ) 在使用越来越多的公开和专有数据 ( 包括各种语音、语言和风格 ) 进行训练时，所具备的广泛报道的涌现能力。

生成式引擎可以创建合成语音，这种语音具有情感参与度、言语果断并且高度口语化，其方式与人类语音高度相似。您可以在以下场景中使用这些语音：知识广博的客户助理、虚拟培训师或采用合成语音媲美人类的广告商。

### Note

这些语音背后的最先进技术就属于用于语言和语音建模的生成式人工智能范式。这项技术的一个副作用是，对训练数据和模型的任何更新都可能会导致语音听起来略有不同，即使随着模型的更新，语音的整体质量有所提高，也是如此。这可能会影响由长期合成的不同内容部分组成的应用场景，例如一个季度的播客。

## 可用的生成式语音

Amazon Polly 目前以生成变体形式提供 43 种声音。

	语言	语言代码	Name/ID	性别
1	英语 ( 澳大利亚 )	en-AU	Olivia	女
2	英语 ( 英国 )	en-GB	Amy Brian	女 男
3	英语 ( 印度 )	en-IN	Kajal	女
4	英语 ( 爱尔兰 )	en-IE	Niamh	女
5	英语 ( 新西兰 )	en-NZ	Aria	女
6	英语 ( 新加坡 )	en-SG	Jasmine	女
7	英语 ( 南非 )	en-ZA	Ayanda	女
8	英语 ( 美国 )	en-US	Danielle Joanna Matthew Ruth Salli Stephen Tiffany	女 女 男 女 女 男 女
9	荷兰语 ( 比利时 )	nl-BE	Lisa	女
10	荷兰语 ( 荷兰 )	nl-NL	Laura	女

	语言	语言代码	Name/ID	性别
11	法语 (比利时)	fr-BE	Isabelle	女
12	法语 (加拿大)	fr-CA	Gabrielle	女
			Liam	男
13	法语 (法国)	fr-FR	Ambre	女
			Céline	女
			弗洛里安	男
			Léa	女
			Rémi	男
14	德语 (奥地利)	de-AT	Hannah	女
15	德语 (德国)	de-DE	Daniel	男
			伦纳特	男
			Vicki	女
16	德语 (瑞士)	de-CH	Sabrina	女
17	意大利语 (意大利)	it-IT	比阿特丽斯	女
			Bianca	女
			Lorenzo	男
18	韩语 (韩国)	ko-KR	Seoyeon	女
19	波兰语 (波兰)	pl-PL	Ewa	女
			Ola	女
20	葡萄牙语 (巴西)	pt-BR	Camila	女

	语言	语言代码	Name/ID	性别
21	西班牙语 (墨西哥)	es-MX	Andrés	男
			Mía	女
22	西班牙语 (西班牙)	es-ES	Lucia	女
			Sergio	男
23	西班牙语 (美国)	es-US	Lupe	女
			Pedro	男

### Note

生成式语音费用在 [Amazon Polly 定价信息页面](#) 上列示。

## 特征和区域兼容性

Amazon Polly 生成式语音在以下区域可用：

- 美国东部 (弗吉尼亚北部) : us-east-1
- 欧洲地区 (法兰克福) : eu-central-1
- 美国西部 (俄勒冈) : us-west-2
- 亚太地区 (东京) : ap-northeast-1
- 亚太地区 (首尔) : ap-northeast-2
- 亚太地区 (新加坡) : ap-southeast-1
- 欧洲地区 (伦敦) : eu-west-2
- 加拿大 (中部) : ca-central-1
- 其他区域不可用

生成式语音支持以下功能：

- 现在，生成引擎中提供了双向流媒体 API，允许同时流式传输输入和输出。此 API 可在以下 AWS 区域使用：美国东部（弗吉尼亚北部）、欧洲（法兰克福）、美国西部（俄勒冈）、亚太地区（新加坡）、欧洲（伦敦）和加拿大（中部）。请访问[文档](#)，详细了解如何使用它。
- Real-time 和异步语音合成操作。
- 生成式引擎不支持新闻播音员风格。
- Amazon Polly 支持许多（但不是所有）SSML 标签。有关 NTTS-supported SSML 标签的更多信息，请参阅[支持的 SSML 标签](#)
- 与标准语音一样，您可以从各种采样率中进行选择，以优化应用程序的带宽和音频质量。标准和神经语音的有效采样率为 8 kHz、16 kHz、22 kHz 或 24 kHz。标准语音的默认值为 22 kHz。生成式语音的默认频率为 24 kHz。Amazon Polly 支持 MP3、OGG (Vorbis) 和原始 PCM 音频流格式。

目前不支持生成式语音标记。

#### Note

目前，欧洲（伦敦）和加拿大（中部）地区仅支持以下生成声音：乔安娜（en-US）、露丝（en-US）、Salli（en-US）、Stephen（en-US）、Amy（en-GB）、Brian（en-GB）、Olivia（en-au）、Florian（fr-FR）、Ambre（fr-FR）、Lorenzo（it-it）、Beatrice（it-it）、Jasmine（en-SG）、Aria（en-nz）、Lennart（de-de）、Vicki（de-de）、Sabrina（de-ch）、Hannah（de-at）、Niamh（en-ie）、Camila（pt-BR）、Lisa（nl-be）和 Seoyeon（ko-kr）

#### Note

在可能性非常小的模型幻觉情况下（并且所采用的生成式引擎的模型行为是按令牌渲染语音），存在一种强制性的紧急停止机制。该内置机制会阻止模型进一步渲染语音。此安全功能基于数据分析，其中模型有可能会产生幻觉，通常是在句子的末尾。在某些情况下，模型认为自身会产生幻觉，然后最终可能会在生成步骤中切断一个单词，从而只渲染这个单词的一半。这可能会产生不恰当的结果。

## Long-form 声音

Amazon Polly 的 Long-form 引擎可以产生类似人类、极具表现力和情感熟练的声音。Long-form 声音旨在吸引听众对较长内容的注意力，例如新闻文章、培训材料或营销视频。

Amazon Polly Long-form 语音是使用尖端的深度学习 TTS 技术开发的。该模型学习复制人类语言的音素、韵律、语调以及其他语音和声学方面，从而产生高度自然的语音输出。

引 Long-form 引擎使用文本嵌入来解释文本的含义。使用文本嵌入，Long-form 引擎可以生成自然语音的正确重点、停顿和语气。最后得到的语音能够结合人类交流中存在的各种情感元素。这包括模仿惊讶的对话或者将对话与叙事区分开来。这些结合在一起，便打造出听起来像真人一样的优质语音产品。

### Note

这些语音背后的最先进技术就属于用于语言和语音建模的生成式人工智能范式。这项技术的一个副作用是，对训练数据和模型的任何更新都可能会导致语音听起来略有不同，即使随着模型的更新，语音的整体质量有所提高，也是如此。这可能会影响由长期合成的不同内容部分组成的应用场景，例如一个季度的播客。

## 可用的长篇语音

Amazon Polly 目前提供四种 en-US 和两种 es-ES 长篇语音。这两种语言都有女声和男声可供选择。英语长篇语音 Daniel、Gregory 和 Ruth 还有对话式 NTTS 变体。

	语言	语言代码	Name/ID	性别
1	英语 ( 美国 )	en-US	Danielle	女
			Gregory	男
			Ruth	女
			Patrick	男
2	西班牙语 ( 西班牙 )	es-ES	Alba	女
			Raúl	男

## 特征和区域兼容性

Amazon Polly 长篇语音在以下区域可用：

- 美国东部 ( 弗吉尼亚北部 ) : us-east-1

- 其他区域不可用

Amazon Polly Long-form 引擎支持以下功能：

- Real-time 和异步语音合成操作。
- 所有[语音标记](#)。
- Amazon Polly 支持许多（但不是所有）SSML 标签。有关 NTTS-supported SSML 标签的更多信息，请参阅[支持的 SSML 标签](#)
- 与标准语音一样，您可以从各种采样率中进行选择，以优化应用程序的带宽和音频质量。标准、长篇和神经语音的有效采样率为 8 kHz、16 kHz、22 kHz 或 24 kHz。标准语音的默认值为 22 kHz。长篇和神经语音的默认值为 24 kHz。Amazon Polly 支持 MP3、OGG (Vorbis) 和原始 PCM 音频流格式。

#### Note

Long-form 语音费用在 [Amazon Polly 定价信息](#)页面上指定。

## 神经语音

Amazon Polly 具有一个神经文本转语音 ( NTTS ) 引擎，可以生成比标准语音质量更高的语音。标准 TTS 语音使用拼接合成。此标准引擎将记录的语音音素串联在一起，生成发音非常自然的合成语音。然而，语音中不可避免的语调变化和用于分割波形的技术限制了语音的质量。Amazon Polly NTTS 引擎不使用标准的拼接合成方法来生成语音。它具有两个部分：

- 一个神经网络 – 将音素序列（最基本的语言单元）转换为声谱图序列。（声谱图是不同频段内能量等级的快照。）
- 一种声码器 – 可将声谱图转换为近乎连续的音频信号。

神经 TTS 系统的第一个组成部分是序列到序列模型。该模型不仅仅从相应的输入创建其结果，而且还考虑输入元素的序列如何配合使用。该模型选择它输出的声谱图，使其频带强调人脑在处理语音时使用的声学特征。

然后该模型的输出传递给神经声码器。声码器会将声谱图转换为语音波形。在用于构建通用级拼接合成系统的大型数据集上训练时，这种序列到序列的方法将产生质量更高、发音更自然的语音。

## 可用的神经语音

神经语音有 36 种语言和语言变体。下表列出了这些语音。

	语言和语言变体	语言代码	Name/ID	性别
1	阿拉伯语 ( 海湾 )	ar-AE	Hala	女
			Zayd	男
2	比利时荷兰语 ( 佛兰芒语 )	nl-BE	Lisa	女
3	加泰罗尼亚语	ca-ES	Arlet	女
4	捷克语	cs-CZ	Jitka	女
5	中文 ( 粤语 )	yue-CN	Hiujin	女
6	中文 ( 普通话 )	cmn-CN	知语	女
7	丹麦语	da-DK	Sofie	女
8	荷兰语	nl-NL	Laura	女
9	英语 ( 澳大利亚 )	en-AU	Olivia	女
10	英语 ( 英国 )	en-GB	Amy*	女
			Emma	女
			Brian	男
			Arthur	男
11	英语 ( 印度 )	en-IN	Kajal	女
12	英语 ( 爱尔兰 )	en-IE	Niamh	女
13	英语 ( 新西兰 )	en-NZ	Aria	女

	语言和语言变体	语言代码	Name/ID	性别
14	英语 ( 新加坡 )	en-SG	Jasmine	女
15	英语 ( 南非 )	en-ZA	Ayanda	女
16	英语 ( 美国 )	en-US	Danielle	女
			Gregory	男
			Ivy	女 ( 孩童 )
			Joanna*	女
			Kendra	女
			Kimberly	女
			Salli	女
			Joey	男
			Justin	男 ( 孩童 )
			Kevin	男 ( 孩童 )
			Matthew*	男
			Ruth	女
			Stephen	男
17	芬兰语	fi-FI	Suvi	女
18	法语 ( 比利时 )	fr-BE	Isabelle	女
19	法语 ( 加拿大 )	fr-CA	Gabrielle	女
			Liam	男
20	法语	fr-FR	Léa	女
			Rémi	男

	语言和语言变体	语言代码	Name/ID	性别
21	德语	de-DE	Vicki	女
			Daniel	男
22	德语 ( 奥地利 )	de-AT	Hannah	女
23	德语 ( 瑞士 )	de-CH	Sabrina	女
24	印地语	hi-IN	Kajal	女
25	意大利语	it-IT	Bianca	女
			Adriano	男
26	日语	ja-JP	Takumi	男
			Kazuha	女
			Tomoko	女
27	韩语	ko-KR	Seoyeon	女
			Jihye	女
28	挪威语	nb-NO	Ida	女
29	波兰语	pl-PL	Ola	女
30	葡萄牙语 ( 巴西 )	pt-BR	Camila	女
			Vitoria/Vitoria	女
			Thiago	男
31	葡萄牙语 ( 欧洲 )	pt-PT	在那里 s/Ines	女
32	西班牙语 ( 西班牙 )	es-ES	Lucia	女
			Sergio	男

	语言和语言变体	语言代码	Name/ID	性别
33	西班牙语 (墨西哥)	es-MX	Mia	女
			Andrés	男
34	西班牙语 (美国)	es-US	Lupe*	女
			Pedro	男
35	瑞典语	sv-SE	Elin	女
36	土耳其语	tr-TR	Burcu	女

\*Amy、Joanna、Lupe 和 Matthew 语音可使用播音讲话风格。有关更多信息，请参阅 [应用新闻播音员的语音](#)。

## 特征和区域兼容性

神经语音并非在所有 AWS 地区都可用，也不支持 Amazon Polly 的所有功能。

以下区域支持神经语音：

- 美国东部 (弗吉尼亚北部) : us-east-1
- 美国西部 (俄勒冈) : us-west-2
- 非洲 (开普敦) : af-south-1
- 亚太地区 (东京) : ap-northeast-1
- 亚太地区 (首尔) : ap-northeast-2
- 亚太地区 (大阪) : ap-northeast-3
- 亚太地区 (孟买) : ap-south-1
- 亚太地区 (新加坡) : ap-southeast-1
- 亚太地区 (悉尼) : ap-southeast-2
- 亚太地区 (马来西亚) : ap-southeast-5
- 加拿大 (中部) : ca-central-1
- 欧洲地区 (法兰克福) : eu-central-1
- 欧洲地区 (爱尔兰) : eu-west-1

- 欧洲地区 ( 伦敦 ) : eu-west-2
- 欧洲地区 ( 巴黎 ) : eu-west-3
- 欧洲 ( 西班牙 ) : eu-south-2
- 欧洲 ( 苏黎世 ) : eu-central-2
- AWS GovCloud (US-West): us-gov-west-1

这些区域的终端节点和协议与标准语音所用的相同。有关更多信息，请参阅 [Amazon Polly 终端节点和配额](#)。

神经语音支持以下功能：

- Real-time 和异步语音合成操作。
- 新闻播音员风格。有关讲话风格的更多信息，请参阅 [应用新闻播音员的语音](#)。
- 所有语音标记。
- 大多 ( 但不是所有 ) Amazon Polly 支持的 SSML 标签。有关 NTTS-supported SSML 标签的更多信息，请参阅支持的标签。

与标准语音一样，您可以从各种采样率中进行选择，以优化应用程序的带宽和音频质量。标准和神经语音的有效采样率为 8 kHz、16 kHz、22 kHz 或 24 kHz。标准语音的默认值为 22 kHz。神经语音的默认值为 24kHz。Amazon Polly 支持 MP3、OGG (Vorbis) 和原始 PCM 音频流格式。

## 标准语音

Amazon Polly 具有一个使用串联合成的标准引擎。此标准引擎将记录的语音音素串联在一起，生成发音非常自然的合成语音。

### 可用的标准语音

Amazon Polly 目前提供 29 种语言和语言变体的 40 种女性和 20 种男性标准语音。

	语言	语言代码	Name/ID	性别
1	阿拉伯语	arb	Zeina	女
2	中文 ( 普通话 )	cmn-CN	知语	女

	语言	语言代码	Name/ID	性别
3	丹麦语	da-DK	Naja	女
			Mads	男
4	荷兰语	nl-NL	Lotte	女
			Ruben	男
5	英语 ( 澳大利亚 )	en-AU	Nicole	女
			Russell	男
6	英语 ( 英国 )	en-GB	Amy	女
			Emma	女
			Brian	男
7	英语 ( 印度 )	en-IN	Aditi	女
			Raveena	女
8	英语 ( 美国 )	en-US	Ivy	女
			Joanna	女
			Kendra	女
			Kimberly	女
			Salli	女
			Joey	男
			Kevin	男
9	英语 ( 威尔士 )	en-GB-WLS	Geraint	男

	语言	语言代码	Name/ID	性别
10	法语	fr-FR	Cé line/Celine	女
			Léa	女
			Mathieu	男
11	法语 ( 加拿大 )	fr-CA	Chantal	女
12	德语	de-DE	Marlene	女
			Vicki	女
			Hans	男
13	印地语	hi-IN	Aditi	女
14	冰岛语	is-IS	Dó ra/Dora	女
			Karl	男
15	意大利语	it-IT	Carla	女
			Bianca	女
			Giorgio	男
16	日语	ja-JP	Mizuki	女
			Takumi	男
17	韩语	ko-KR	Seoyeon	女
18	挪威语	nb-NO	Liv	女

	语言	语言代码	Name/ID	性别
19	波兰语	pl-PL	Ewa	女
			Maja	女
			Jacek	男
			Jan	男
20	葡萄牙语 (巴西)	pt-BR	Camila	女
			Vitoria/Vitoria	女
			Ricardo	男
21	葡萄牙语 (欧洲)	pt-PT	Ines/Ines	女
			Cristiano	男
22	罗马尼亚语	ro-RO	Carmen	女
23	俄语	ru-RU	Tatyana	女
			Maxim	男
24	西班牙语 (西班牙)	es-ES	Conchita	女
			Lucia	女
			Enrique	男
25	西班牙语 (墨西哥)	es-MX	Mia	女
26	西班牙语 (美国)	es-US	Lupe	女
			Penelope/Penelope	女
			Miguel	男

	语言	语言代码	Name/ID	性别
27	瑞典语	sv-SE	Astrid	女
28	土耳其语	tr-TR	Filiz	男
29	威尔士语	cy-GB	Gwyneth	女

## 特征和区域兼容性

Amazon Polly 标准语音在以下 Amazon Polly 区域可用：

- 美国东部 ( 弗吉尼亚北部 ) : us-east-1
- 美国东部 ( 俄亥俄 ) : us-east-2
- 美国西部 ( 加利福尼亚北部 ) : us-west-1
- 美国西部 ( 俄勒冈 ) : us-west-2
- 非洲 ( 开普敦 ) : af-south-1
- 亚太地区 ( 香港 ) : ap-east-1
- 亚太地区 ( 东京 ) : ap-northeast-1
- 亚太地区 ( 首尔 ) : ap-northeast-2
- 亚太地区 ( 大阪 ) : ap-northeast-3
- 亚太地区 ( 孟买 ) : ap-south-1
- 亚太地区 ( 新加坡 ) : ap-southeast-1
- 亚太地区 ( 悉尼 ) : ap-southeast-2
- 亚太地区 ( 马来西亚 ) : ap-southeast-5
- 中国 ( 宁夏 ) : cn-northwest-1
- 加拿大 ( 中部 ) : ca-central-1
- 欧洲地区 ( 法兰克福 ) : eu-central-1
- 欧洲地区 ( 爱尔兰 ) : eu-west-1
- 欧洲地区 ( 伦敦 ) : eu-west-2
- 欧洲地区 ( 巴黎 ) : eu-west-3
- 欧洲 ( 西班牙 ) : eu-south-2
- 欧洲地区 ( 斯德哥尔摩 ) : eu-north-1

- 中东 ( 巴林 ) : me-south-1
- 南美洲 ( 圣保罗 ) : sa-east-1
- AWS GovCloud (US-West): us-gov-west-1

这些区域的端点和协议与神经语音所用的相同。有关更多信息，请参阅 [Amazon Polly 终端节点和配额](#)。

Amazon Polly 标准引擎支持以下特征 ( 待定 ) :

- Real-time 和异步语音合成操作。
- 所有 [语音标记](#)。
- Amazon Polly 支持许多 ( 但不是所有 ) SSML 标签。有关 NTTS-supported SSML 标签的更多信息，请参阅 [支持的 SS ML 标签](#)。
- 您可以从各种采样率中进行选择，以优化应用程序的带宽和音频质量。标准语音的默认采样率为 22 kHz。Amazon Polly 支持 MP3、OGG (Vorbis) 和原始 PCM 音频流格式。

#### Note

标准语音费用在 [Amazon Polly 定价信息页面](#) 上列示。

## 选择语音引擎

您可以通过 Amazon Polly 控制台访问 Amazon Polly 的声音，或者。AWS CLI

在控制台上选择语音引擎

1. 打开 Amazon Polly 主机，网址为。 <https://console.aws.amazon.com/polly/>
2. 从 Amazon Polly 控制台中，选择所需的语音引擎。
3. 从语音下拉菜单中选择所需语音。
4. 使用您选择的文本生成 TTS 音频。

要在中选择语音引擎 AWS CLI，请在SyntheszieSpeech或 StartSpeechSynthesisTask API 操作VoiceId中指定Engine和。要查看一些示例，请参阅[快速入门代码示例](#)和 [Python 示例](#)。

# 语音标记

语音标记 是描述合成语音的元数据，例如，句子或单词在音频流中的起始和结束位置。当您为文本请求语音标记时，Amazon Polly 将返回此元数据，而不是合成语音。通过将语音标记与合成语音音频流配合使用，您可以为您的应用程序提供 stronger 的视觉体验。

例如，通过将元数据与来自文本的音频流组合使用，您就能够将语音与面部动画同步（嘴唇同步），或者在说出字幕单词时对其进行突出显示。

使用神经引擎、长格式或标准 text-to-speech 引擎时可以使用语音标记。

## 主题

- [语音标记类型](#)
- [语音视位和 Amazon Polly](#)
- [语音标记输出](#)
- [请求语音标记](#)
- [没有 SSML 的语音标记示例](#)
- [有 SSML 的语音标记示例](#)

## 语音标记类型

您可以使用 [SynthesizeSpeech](#) 或 [StartSpeechSynthesisTask](#) 命令的 [SpeechMarkTypes](#) 选项来请求语音标记。您可以指定要从您输入的文本返回的元数据元素。您可以请求最多 4 种类型的元数据，但每个请求必须至少指定一种类型。未根据请求生成音频输出。

例如 AWS CLI，在：

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

Amazon Polly 生成使用以下元素的语音标记：

- 句子 – 表示输入文本中的句子元素。
- 单词 – 表示输入文本中的单词元素。
- 语音视位 – 描述说每个音素时的面部和口腔运动。有关更多信息，请参阅 [语音视位和 Amazon Polly](#)。
- ssml – 描述 SSML 输入文本中的 <mark> 元素。有关更多信息，请参阅 [由 SSML 文档生成语音](#)。

## 语音视位和 Amazon Polly

语音视位代表说一个词时的面部和口腔位置。它是一个音素的视觉等价物，是形成单词的基本声学单元。语音视位是语音的基本可视构建基块。

每种语言都有一组对应于其特定音素的语音视位。在语言中，每个音素都有相应的代表口腔发音时的形状的语音视位。然而，并非所有语音视位都能映射到特定的音素，因为许多音素虽然听上去不一样，但发音时的形状相同。例如，在英语中，单词“pet”（宠物）和“bet”（打赌）声音上是不同的。但是，在目视观察中（无声音），它们看起来完全相同。

下图列出了美国英语语音的部分国际音标 (IPA) 音素和拓展音标字母评估法 (X-SAMPA) 符号，及其对应的语音视位。

如需获得完整列表和所有可用语音的表格，请参阅 [Amazon Polly 中的语言](#)。

IPA	X-SAMPA	说明	示例	语音视位
辅音				
b	b	浊双唇塞音	bed	p
d	d	浊齿龈塞音	dig	t
ɟʒ	dʒ	浊龈后塞擦音	jump	S
ð	D	浊齿擦音	then	T
f	f	清唇齿擦音	five	f
g	g	浊软顎塞音	game	k
h	h	清喉擦音	house	k
...	...	...	...	...

## 语音标记输出

Amazon Polly 将返回以换行符分隔的 JSON 流中的语音标记对象。语音标记对象包含以下字段：

- time – 相应音频流开头的时戳（以毫秒为单位）

- type – 语音的类型 ( 句子、单词、语音视位或 ssm1 标记 ) 。
- start – 输入文本中对象开头的偏移量 ( 以字节而不是字符为单位 , 不包括语音视位标记 )
- end – 输入文本中对象末尾的偏移量 ( 以字节而不是字符为单位 , 不包括语音视位标记 )
- value – 根据语音标记类型变化
  - SSML : <mark> SSML 标签
  - viseme : 语音视位名称
  - word 或 sentence : 输入文本的子字符串 , 由开始和结束字段分隔

例如 , Amazon Polly 从文本“Mary had a little lamb” ( 玛丽有一只小羊羔 ) 生成以下 word 语音标记对象 :

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

所描述的单词 ( “had” ( 具有 ) ) 开始于音频流开始后 373 毫秒 , 并从输入文本的字节 5 处开始 , 在字节 8 处结束。

#### Note

此元数据用于 Joanna 语音 ID。如果您使用另一个有相同输入文本的声音 , 元数据可能有所不同。

## 请求语音标记


您可以使用控制台或 synthesize-speech 命令从 Amazon Polly 请求语音标记。然后 , 您可以查看元数据或将其保存到文件中。

### Console

在控制台上生成语音标记

1. 登录 AWS 管理控制台 并打开 Amazon Polly 控制台 , 网址为。 <https://console.aws.amazon.com/polly/>
2. 选择文本到语音转换选项卡。
3. 打开 SSML 以使用 SSML。

4. 在输入框中键入或粘贴您的文本。
5. 对于语言，选择您的文本使用的语言。
6. 对于语音，选择您要使用的语音。
7. 要更改文本发音，请展开其他设置，打开自定义发音，然后在应用词典中，选择所需的词典。
8. 要验证语音，请选择收听。
9. 打开语音文件格式设置。

 Note

下载 OGG MP3、PCM、mu-law 或 a-law 格式不会生成语音标记。

10. 对于文件格式，选择语音标记。
11. 对于语音标记类型，请选择要生成的语音标记的类型。选择 SSML 元数据的选项仅在 SSML 打开时可用。有关通过 Amazon Polly 使用 SSML 的更多信息，请参阅 [由 SSML 文档生成语音](#)。
12. 选择下载。

## AWS CLI

除了输入文本之外，以下元素也都是返回此元数据所必需的：

- `output-format`

当返回语音标记时，Amazon Polly 仅支持 JSON 格式。

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

如果您使用的是不受支持的输出格式，Amazon Polly 将引发异常。

- `voice-id`

为了确保元数据与相关联的音频流匹配，请指定相同语音用于生成合成语音音频流。可用语音的语速不相同。如果您使用的语音与用于生成语音的不同，则元数据将与音频流不匹配。

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

- `speech-mark-types`

指定您需要的语音标记类型。您可以请求任何语音标记类型，但必须指定至少一个类型。

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

- **text-type**

纯文本是 Amazon Polly 的默认输入文本，因此，如果要返回 SSML 语音标记，必须使用 `text-type ssml`。

- **outfile**

指定写入元数据的输出文件。

```
MaryLamb.txt
```

以下 AWS CLI 示例是针对 Unix、Linux 和 macOS 进行格式化的。对于 Windows，请将每行末尾的反斜杠 (\) Unix 行继续符替换为脱字号 (^) 并在输入文本周围使用全角引号 (")，内部标签使用单引号 (')。

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

## 没有 SSML 的语音标记示例

以下示例显示了您所请求元数据的简单句子在屏幕上显示的效果：“Mary had a little lamb”(玛丽有一只小羊羔)。为简单起见，我们在此示例中未包括 SSML 语音标记。

以下 AWS CLI 示例是针对 Unix、Linux 和 macOS 进行格式化的。对于 Windows，请将每行末尾的反斜杠 (\) Unix 行继续符替换为脱字号 (^) 并在输入文本周围使用全角引号 (")，内部标签使用单引号 (')。

```
aws polly synthesize-speech \  
  --output-format json \  
  --voice-id Joanna \  
  --text 'Mary had a little lamb.'
```

```
--text 'Mary had a little lamb.' \  
--speech-mark-types='["viseme", "word", "sentence"]' \  
MaryLamb.txt
```

当您发出此请求时，Amazon Polly 会在 .txt 文件中返回以下内容：

```
{ "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" }
```

在这个输出中，文本的每个部分都由语言标记断开：

- 句子“Mary had a little lamb.”（玛丽有一只小羊羔。）
- 文本中的每个单词：“Mary”、“had”、“a”、“little”和“lamb”。
- 相应音频流中每个声音的语音视位：“p”、“E”、“r”、“i”等。有关语音视位的更多信息，请参阅 [语音视位和 Amazon Polly](#)。

## 有 SSML 的语音标记示例

从 SSML 增强文本生成语音标记的过程与 SSML 不存在时的过程相似。使用 `synthesize-speech` 命令，并指定 SSML 增强文本和您所需的语音标记类型，如下例所示。为了使示例更容易阅读，我们未包含语音视位语音标记，但这些也可以包含在内。

以下 AWS CLI 示例是针对 Unix、Linux 和 macOS 进行格式化的。对于 Windows，请将每行末尾的反斜杠 (\) Unix 行继续符替换为脱字号 (^) 并在输入文本周围使用全角引号 ( “ ” )，内部标签使用单引号 ( ‘ ’ )。

```
aws polly synthesize-speech \  
  --output-format json \  
  --voice-id Joanna \  
  --text-type ssm1 \  
  --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
```

当您发出此请求时，Amazon Polly 会在 .txt 文件中返回以下内容：

```
{"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"}
```

## 由 SSML 文档生成语音

您可使用 Amazon Polly 从纯文本或从通过语音合成标记语言 (SSML) 标记的文档生成语音。使用 SSML 增强文本让您可以进一步控制 Amazon Polly 如何根据您提供的文本生成语音。

您可以利用 SSML 标签自定义并控制语音的各个方面，例如发音、音量和语速。在中 AWS 管理控制台，要转换为音频的 SSML 增强文本在页面的 SSML 选项卡上输入。Text-to-Speech 以纯文本形式输入的文本依赖于所选语言和语音的默认设置，而使用 SSML 增强的文本不仅会向 Amazon Polly 告知您所说的内容，还会告知您希望如何说。除了添加的 SSML 标签，Amazon Polly 合成 SSML 增强文本的方式与合成纯文本的方式相同。请参阅[使用 Amazon Polly 合成语音示例](#)了解更多信息。

在使用 SSML 时，您用 `< speak >` 标签包含整个文本，以便让 Amazon Polly 知道您正在使用 SSML。例如：

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

然后，您对 `< speak >` 标签中的文本使用特定的 SSML 标签来定义所需的文本发音方式。您可以添加停顿、改变讲话的节奏、降低或提高声音的音量或添加许多其他自定义项，使文本的发音适合您。有关可使用的 SSML 标签的完整列表，请参阅[支持的 SSML 标签](#)。

例如，您可以在文本中增加一个较长的停顿，或更改语速或音高。其他选项包括：

- 强调特定的单词或短语
- 使用语音发音
- 包括呼吸声
- 轻读
- 使用新闻播音员风格。

有关 Amazon Polly 支持的 SSML 标签及其使用方法的完整详细信息，请参阅[支持的 SSML 标签](#)

使用 SSML 时，有几个预留字符需要特殊处理。这是因为 SSML 将这些字符用作其代码的一部分。为了使用它们，您可以使用特定实体对其进行转义。有关更多信息，请参阅[SSML 中的预留字符](#)。

Amazon Polly 可通过一部分 SSML 标记标签（由[W3C 推荐的语音合成标记语言 \(SSML\) 版本 1.1](#) 定义）提供此类控制。

您可在 Amazon Polly 控制台中使用 SSML，也可通过 AWS CLI 使用 SSML。以下主题说明了如何使用 SSML 生成语音并控制输出，以便精准地满足您的需求。

## 主题

- [SSML 中的预留字符](#)
- [在控制台上使用 SSML](#)
- [通过合成语音命令使用 SSML](#)
- [合成 SSML 增强文档](#)
- [支持的 SSML 标签](#)

## SSML 中的预留字符

有五种预定义字符通常无法在 SSML 语句中使用。这些实体根据语言规范予以预留。这些字符如下：

Character

义  
代  
码

&quot;  
号  
( 双  
引  
号 )

&amp;  
示  
和  
的  
符  
号

&ap;  
号  
或  
单  
引  
号

## Character

义  
代  
码

&lt;  
于  
号

&gt;  
于  
号

由于 SSML 会将这些字符用在其代码中，要在 SSML 中使用这些符号，您必须在使用时转义该字符。您使用转义码而不是实际字符，以便在仍然创建有效的 SSML 文档时正确显示。例如，以下句子

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

在 SSML 中将转义为

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

在这种情况下，撇号和表示和的符号的特殊字符将被转义，因此 SSML 文档仍然有效。

对于 &、< 和 > 符号，使用 SSML 时始终需要转义码。另外，当您使用撇号/单引号 (') 作为撇号时，也必须使用转义码。

但是，当使用双引号 (") 或 apostrophe/single 引号 (') 作为引号时，是否使用转义码取决于上下文。

双引号

- 在由双引号界定的属性值中必须进行转义。例如，在以下 AWS CLI 代码中

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

- 在文本上下文中不需要进行转义。例如，在下面的上下文中

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

- 在由单引号界定的属性值中时，不需要进行转义。例如，在下面的 AWS CLI 代码中

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

## 单引号

- 用作撇号时必须进行转义。例如，在下面的上下文中

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

- 在文本上下文中不需要进行转义。例如，在下面的上下文中

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

- 在由双引号界定的代码属性中，不需要进行转义。例如，在下面的 AWS CLI 代码中

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

## 在控制台上使用 SSML

在以下示例中，您将使用 SSML 标签告知 Amazon Polly 在朗读一个短段落时将“W3C”替换为“World Wide Web Consortium”。您还可以使用标签引入停顿以及轻读某词。将此练习的结果与 [应用词典（合成语音）](#) 进行比较。

有关 SSML 的更多信息以及示例，请参阅 [支持的 SSML 标签](#)。

由 SSML 增强文本合成语音（控制台）

1. 登录 AWS 管理控制台 并打开 Amazon Polly 控制台，网址为 <https://console.aws.amazon.com/polly/>
2. 请选择文本到语音转换选项卡（如果未显示）。
3. 打开 SSML。
4. 在文本框中键入或粘贴以下文本：

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

Amazon Polly 可通过 SSML 标签获知如何处理文本：

- `<break time="1s"/>` 告知 Amazon Polly 在最开始的两句话之间停顿 1 秒。
- `<sub alias="World Wide Web Consortium">W3C</sub>` 告知 Amazon Polly 将缩写 W3C 替换为 World Wide Web Consortium。
- `<amazon:effect name="whispered">Nice job</amazon:effect>` 告知 Amazon Polly 轻读第二个“Nice job”。

#### Note

使用时 AWS CLI，请用引号将输入文本括起来，以将其与周围的代码区分开来。Amazon Polly 控制台不会显示您的代码，所以您在使用时不需要将输入文本放在引号中。

5. 对于语言，选择英语（美国），然后选择一个语音。
6. 要收听语音，请选择收听。
7. 要保存语音文件，请选择下载。如果要另存为其他格式，请展开其他设置，打开语音文件格式设置，选择所需的格式，然后选择下载。

## 通过合成语音命令使用 SSML

此示例展示了如何使用具有 SSML 字符串的 `synthesize-speech` 命令。在使用 `synthesize-speech` 命令时，通常您需要提供以下要素：

- 输入文本（必需）
- 开始和结束标签（必需）
- 输出格式

- 语音

在此示例中，您通过引号指定简单的文本字符串，以及必要的开始和结束 `< speak >< /speak >` 标签。

### Important

在 Amazon Polly 控制台中不需要将输入文本放在引号中，但使用 AWS CLI 必须将输入文本放在引号中。另外一项重要的注意事项是，您需要区分输入文本两端的引号和个别标签需要使用的引号。

例如，您可以在输入文本两端使用标准引号 (")，并将单引号 (') 用于内部标签，也可相反使用。Unix、Linux 和 macOS 适用于这两种选项。但对于 Windows，必须在输入文本两端使用标准引号，并将单引号用于标签。

对于所有操作系统，您都可以在输入文本两端使用标准引号 (")，并将单引号 (') 用于内部标签。例如：

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

对于 Unix、Linux 和 macOS，您也可以在输入文本两端使用单引号 (')，并将标准引号 (") 用于内部标签：

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

以下 AWS CLI 示例是针对 Unix、Linux 和 macOS 进行格式化的。对于 Windows，请将每行末尾的反斜杠 (\) Unix 行继续符替换为脱字号 (^) 并在输入文本周围使用全角引号 (")，内部标签使用单引号 (')。

```
aws polly synthesize-speech \  
--text-type ssm1 \  
--text '<speak>Hello world</speak>' \  
--output-format mp3 \  
--voice-id Joanna \  
speech.mp3
```

要试听合成语音，请使用任何播放器播放生成的 `speech.mp3` 文件。

## 合成 SSML 增强文档

如果输入文本较长，可能将 SSML 内容保存为文件，简单地在 `synthesize-speech` 命令中指定文件名更加方便。例如，您可以把以下内容保存为名为 `example.xml` 的文件：

```
<?xml version="1.0"?>
<spek 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</spek>
```

`xml:lang` 属性指定 `en-US`（美国英语）作为输入文本的语言。有关输入文本的语言和所选语音的语言对 `SynthesizeSpeech` 操作的影响，请参阅 [为特定词语指定另一种语言](#)。

### 要运行 SSML 增强文件

1. 将 SSML 保存为文件（例如 `example.xml`）。
2. 从存储 XML 文件的路径运行以下 `synthesize-speech` 命令，并用 `file:\\example.xml` 替换输入文本，指定 SSML 文件作为输入。由于此命令指向一个文件，不包含实际输入文本，所以您不需要使用引号。

#### Note

以下 AWS CLI 示例是针对 Unix、Linux 和 macOS 进行格式化的。对于 Windows，请将每行末尾的反斜杠 (\) Unix 行继续符替换为脱字号 (^)。

```
aws polly synthesize-speech \
--text-type ssm1 \
--text file://example.xml \
--output-format mp3 \
--voice-id Joanna \
speech.mp3
```

3. 要试听合成语音，请使用任何播放器播放生成的 `speech.mp3` 文件。

## 支持的 SSML 标签

标准语音支持除 `<amazon:domain name="news">` 之外的所有标签。下表提供了其它语音的标签可用性。

Amazon Polly 支持以下 SSML 标签：

处理建议	SSML 标签	神经语音可用性	长篇语音可用性	生成式语音可用性
<a href="#">添加停顿</a>	<code>&lt;break&gt;</code>	完全可用	完全可用	完全可用
<a href="#">强调词语</a>	<code>&lt;emphasis&gt;</code>	不可用	不可用	不可用
<a href="#">为特定词语指定另一种语言</a>	<code>&lt;lang&gt;</code>	完全可用	完全可用	完全可用
<a href="#">在您的文本中插入自定义标签</a>	<code>&lt;mark&gt;</code>	完全可用	完全可用	部分可用
<a href="#">在段落之间添加停顿</a>	<code>&lt;p&gt;</code>	完全可用	完全可用	完全可用
<a href="#">使用语音发音</a>	<code>&lt;phoneme&gt;</code>	完全可用	完全可用	部分可用
<a href="#">控制音量、语速和音高</a>	<code>&lt;prosody&gt;</code>	部分可用	部分可用	部分可用
<a href="#">为合成语音设置最长持续时间</a>	<code>&lt;prosody amazon:max-duration&gt;</code>	不可用	不可用	不可用
<a href="#">在句子之间添加停顿</a>	<code>&lt;s&gt;</code>	完全可用	完全可用	完全可用
<a href="#">控制如何朗读特殊的词语类型</a>	<code>&lt;say-as&gt;</code>	部分可用	完全可用	完全可用
<a href="#">标识 SSML 增强文本</a>	<code>&lt;speak&gt;</code>	完全可用	完全可用	完全可用
<a href="#">首字母缩略词和缩写的发音</a>	<code>&lt;sub&gt;</code>	完全可用	完全可用	完全可用
<a href="#">指定词性以改善发音</a>	<code>&lt;w&gt;</code>	完全可用	完全可用	完全可用

处理建议	SSML 标签	神经语音可用性	长篇语音可用性	生成式语音可用性
<a href="#">添加呼吸音</a>	<amazon:auto-breaths>	不可用	不可用	不可用
<a href="#">新闻播音员风格</a>	<amazon:domain name="news">	仅限选定的神经语音	不可用	不可用
<a href="#">添加动态范围压缩</a>	<amazon:effect name="drc">	完全可用	完全可用	不可用
<a href="#">柔和地朗读</a>	<amazon:effect phonation="soft">	不可用	不可用	不可用
<a href="#">控制音质</a>	<亚马逊 vocal-tract-length :effect >	不可用	不可用	不可用
<a href="#">轻读</a>	<amazon:effect name="whispered">	不可用	不可用	不可用

### Note

如果您在标准、神经或长篇格式中使用不受支持的 SSML 标签，则会出现错误。

## 标识 SSML 增强文本

<speaK>

生成式、长篇、神经和标准 TTS 格式都支持此标签。

<speaK> 标签是所有 Amazon Polly SSML 文本的根元素。所有 SSML 增强文本均位于 <speaK> 标签之内。

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

## 添加停顿

`<break>`

生成式、长篇、神经和标准 TTS 格式都支持此标签。

要向文本添加停顿，请使用 `<break>` 标签。您可以根据强度设置停顿（等同于逗号、句子或段落后的停顿），也可以将停顿设置为特定的时间长度（以秒为单位或以毫秒为单位）。如果您未指定属性来确定停顿时长，则 Amazon Polly 将使用默认值（即 `<break strength="medium"/>`），这将为停顿添加逗号后停顿的时长。

strength 属性值：

- none：无停顿。使用 none 去除正常情况下会出现的停顿，例如句号之后的停顿。
- x-weak：与 none 长度相同，无停顿。
- weak：设置与逗号后的停顿相同的持续时间的停顿。
- medium：与 weak 长度相同。
- strong：设置与句子后的停顿相同的持续时间的停顿。
- x-strong：设置与段落后的停顿相同的持续时间的停顿。

time 属性值：

- `[number]s`：停顿的持续时长，以秒为单位。持续时长上限为 10s。
- `[number]ms`：停顿的持续时长，以毫秒为单位。持续时长上限为 10000ms。

例如：

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

如果未将属性用于 break 标签，则结果将因文本而异：

- 如果 break 标签旁边没有其他标点，则将创建 `<break strength="medium"/>`（逗号时长停顿）。

- 如果标签位于逗号旁边，则将标签升级到 `<break strength="strong"/>` ( 句子时长停顿 )。
- 如果标签位于句号旁边，则将标签升级到 `<break strength="x-strong"/>` ( 段落时长停顿 )。

## 强调词语

`<emphasis>`

只有标准 TTS 格式支持此标签。

要强调词语，请使用 `<emphasis>` 标签。强调词语会更改语速和音量。Amazon Polly 通过更大声、更缓慢地朗读文本来进行强调。不需强调的内容会读得更轻、更快。请使用 `level` 属性指定强调程度。

`level` 属性值：

- `Strong`：提高音量、降低语速，朗读更大声、更缓慢。
- `Moderate`：提高音量、降低语速，但程度弱于 `strong`。`Moderate` 为默认值。
- `Reduced`降低音量，加快语速。朗读更轻柔、更快。

### Note

语音的正常语速和音量在 `moderate` 和 `reduced` 级别之间。

例如：

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

## 为特定词语指定另一种语言

`<lang>`

生成式、长篇、神经和标准 TTS 格式都支持此标签。对于生成语音，`<lang>` 标签只能在完整句子周围使用。

利用 `<lang>` 标签将特定的词语、短语或句子指定为另一种语言。外语词汇和短语如果放在 `<lang>` 标签对之间一般会朗读得更好。请使用 `xml:lang` 属性指定语言。有关可用语言的完整列表，请参阅 [Amazon Polly 中的语言](#)。

如果您不应用 `<lang>` 标签，输入文本中的所有词语将使用 `voice-id` 中指定的语音的语言朗读。如果您应用 `<lang>` 标签，将使用这种语言朗读词语。

例如，如果 `voice-id` 是 Joanna ( 说美国英语 ) ，Amazon Polly 会使用 Joanna 的语音朗读以下内容，没有法语腔调：

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

如果您使用 Joanna 语音时加入 `<lang>` 标签，Amazon Polly 会用 Joanna 的语音以美国口音的法语朗读这个句子。

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

因为 Joanna 的母语不是法语，发音会以她的母语为基础，也就是美国英语。例如，虽然完美的法语发音在 `français` 这个词中有一个小舌颤音 /R/，但 Joanna 的美国英语语音将这个音素发为相应的 /r/。

如果您使用说意大利语的 `voice-id` Giorgio 朗读以下文本，Amazon Polly 会以 Giorgio 的语音通过意大利语发音朗读这个句子：

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

如果您使用同一语音，但加入 `<lang>` 标签，Amazon Polly 会以意大利口音的英语进行 Bruce Springsteen 的发音。

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

```
</speak>
```

合成语音时，此标签也可用作可选 [DefaultLangCode](#) 选项的替代品。但是，这样做需要您使用 SSML 设置文本格式。

## 在您的文本中插入自定义标签

```
<mark>
```

长篇、神经和标准 TTS 格式都支持此标签。此标签对生成式语音没有任何作用，因为语音标记不适用于生成式语音。

要在文本中插入自定义标签，请使用 `<mark>` 标签。Amazon Polly 不会针对此标签采取任何操作，但会在 SSML 元数据中返回此标签的位置。此标签可以是您希望注明的任何内容，但需要具备以下格式：

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

例如，假设标签名称是 "animal"，输入文本是：

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

Amazon Polly 可能会返回以下 SSML 元数据：

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

## 在段落之间添加停顿

```
<p>
```

生成式、长篇、神经和标准 TTS 格式都支持此标签。

要在文本段落之间添加停顿时间，请使用 `<p>` 标签。使用此标签，停顿时长将长于母语者在逗号或句子结束后停顿的时长。在段落两端使用 `<p>` 标签：

```
<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>
```

这与使用 `<break strength="x-strong"/>` 指定停顿的效果相同。

## 使用语音发音

`<phoneme>`

长篇、神经和标准 TTS 格式都支持此标签。

要使 Amazon Polly 针对特定文本使用语音发音，请使用 `<phoneme>` 标签。

`<phoneme>` 标签需要具有两个属性。它们指示 Amazon Polly 使用的音标和更正发音的音标符号：

- alphabet
  - ipa— 指明将使用国际音标 (IPA)。
  - x-sampa— 指明将使用拓展音标字母评估法 (X-SAMPA)。
- ph
  - 指定发音的音标符号。有关更多信息，请参阅 [Amazon Polly 中的语言](#)。

添加 `<phoneme>` 标签后，Amazon Polly 将使用 `ph` 属性指定的发音，而不是所选语音所用的语言默认关联的标准发音。

例如 "pecan" 这个词可以发两个音。在以下示例中，“pecan”在每一行指定了不同的发音。Amazon Polly 根据 `ph` 属性发音 pecan，而不是使用默认发音。

### 国际音标 (IPA)

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

### 拓展音标字母评估法 (X-SAMPA)

```
<speak>
```

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

中文普通话使用拼音进行语音发音。

## 拼音

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

日语使用读音假名和发音假名。

## 读音假名

```
<speak>
  ###<phoneme alphabet="x-amazon-yomigana" ph="####">##</phoneme>###
  ###<phoneme alphabet="x-amazon-yomigana" ph="####">##</phoneme>###
  ###<phoneme alphabet="x-amazon-yomigana" ph="Hirokazu">##</phoneme>###
</speak>
```

## 发音假名

```
<speak>
  ###<phoneme alphabet="x-amazon-pron-kana" ph="##'##">##</phoneme>###
</speak>
```

## 控制音量、语速和音高

### <prosody>

标准的 TTS 语音完全支持 prosody 标签属性。生成式、神经和长篇语音支持 volume 和 rate 属性，但不支持 pitch 属性。对于生成式语音，prosody 标签只能在完整句子周围使用。

要控制所选语音的音量、语速或音高，请使用 prosody 标签。

音量、语速和音高取决于所选的具体语音。不同语言的语音各有不同，说同一语言的不同人之间也各不相同。因此，虽然所有语言的属性都是类似的，但各种语言之间的差异很明显，没有适用于所有语言的值。

prosody 标签有三个属性，每个属性均有若干可用的设置值。每个属性使用相同的语法：

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

- volume

- default：将当前语音的音量重置为默认级别。
- silent、x-soft、soft、medium、loud、x-loud：将当前语音的音量设置为预定义值。
- +ndB、-ndB：相对于当前音量水平调整音量。值 +0dB 表示没有变化，+6dB 表示大约为当前音量的两倍，-6dB 表示大约为当前音量的一半。

例如，可通过以下方式设置段落的音量：

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

或者，您也可以通过以下方式设置它：

```
<speack>  
    And sometimes a lower volume <prosody volume="-6dB">is a more effective way of  
    interacting with your audience.</prosody>  
</speack>
```

- rate

- x-slow、slow、medium、fast、x-fast：将所选语音设为预先定义的音高值。
- n%：语速的非负值百分比变更。例如，值为 100% 意味着语速不变，值为 200% 意味着语速是默认值的两倍，值为 50% 意味着语速是默认值的一半。此值的范围在 20-200% 之间。

例如，可通过以下方式设置段落的语速：

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

或者，您也可以通过以下方式设置它：

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

## • pitch

- default : 将当前语音的音高重置为默认级别。
- x-low、low、medium、high、x-high : 将当前语音的音高设置为预定义值。
- +n% 或 -n% : 按相对百分比调节音高。例如，值 +0% 表示没有基准音高更改，+5% 提供了更高一点的基准音高，-5% 会产生更低一点的基准音高。

例如，可通过以下方式设置段落的音高：

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

或者，您也可以通过以下方式设置它：

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

<prosody> 标签必须至少包含一个属性，但同一标签中可包含更多属性。

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

还可使用嵌套标签进行组合，例如：

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

```
</speak>
```

### Note

<prosody>目前部分可用于生成语音。

## 为合成语音设置最长持续时间

```
<prosody amazon:max-duration>
```

目前只有标准 TTS 格式支持此标签。

要控制您希望语音在合成后花费多长时间，请使用具有 <prosody> 属性的 `amazon:max-duration` 标签。

合成的语音的持续时间会因您选择的语音而略有不同。这样很难使合成的语音与需要精确计时的视觉效果或其他活动匹配。对于翻译应用程序来说，此问题被扩大，因为使用不同语言说特定短语所需的时间可能大不相同。

<prosody amazon:max-duration> 标签使合成的语音与所需的时间量（持续时间）匹配。

此标签使用以下语法：

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

使用 <prosody amazon:max-duration> 标签，您可以指定持续时间，以秒或毫秒为单位：

- `ns`：最长持续时间，以秒为单位
- `nms`：最长持续时间，以毫秒为单位

例如，以下说出的文本的最长为 2 秒：

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

放置在标签中的文本没有超过指定的持续时间。如果所选语音或语言所需时间通常长于该持续时间，Amazon Polly 可加快语音以使其符合指定的持续时间。

如果指定的持续时间长于以正常速率读取文本所需的时间，Amazon Polly 将正常读取语音。它不会减慢语音或添加静默，因此生成的音频比请求的短。

### Note

Amazon Polly 提高的速度不会超过正常速率的 5 倍。如果文本的说出速度超过此值，则通常不明智。如果语音在加速到最大值后仍无法符合指定的持续时间，则音频将加速，但持续时间将长于指定的持续时间。

您可以在 `<prosody amazon:max-duration>` 标签中包括单个句子或多个句子，并且可以在文本中使用多个 `<prosody amazon:max-duration>` 标签。

例如：

```
<speak>
  <prosody amazon:max-duration="2400ms">
    Human speech is a powerful way to communicate.
  </prosody>
  <break strength="strong"/>
  <prosody amazon:max-duration="5100ms">
    Even a simple 'Hello' can convey a lot of information depending on the pitch,
    intonation, and tempo.
  </prosody>
  <break strength="strong"/>
  <prosody amazon:max-duration="8900ms">
    We naturally understand this information, which is why speech is ideal for
    creating applications where
    a screen isn't practical or possible, or simply isn't convenient.
  </prosody>
</speak>
```

使用 `<prosody amazon:max-duration>` 标签可增加 Amazon Polly 返回合成的语音时的延迟。延迟程度取决于段落及其长度。我们建议使用由较短文本段落组成的文本。

## 限制

在如何使用 `<prosody amazon:max-duration>` 标签及其如何与其他 SSML 标签结合使用方面存在限制：

- `<prosody amazon:max-duration>` 标签中的文本不能超过 1500 个字符。
- 不能嵌套 `<prosody amazon:max-duration>` 标签。如果您将一个 `<prosody amazon:max-duration>` 标签放置在另一个此标签中，则 Amazon Polly 会忽略内部标签。

例如，在下面，将忽略 `<prosody amazon:max-duration="5s">` 标签：

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

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

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

- 不能在 `<prosody>` 标签中使用具有 `rate` 属性的 `<prosody amazon:max-duration>` 标签。这是因为二者都会影响文本的朗读速度。

在以下示例中，Amazon Polly 将忽略 `<prosody rate="2">` 标签：

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

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

## 停顿和 `max-duration`

当使用 `max-duration` 标签时，您仍可以在文本中插入停顿。但是，Amazon Polly 在计算语音的最长持续时间时会包括停顿的长度。此外，Amazon Polly 会保留放置在段落中的逗号和句号处出现的短停顿并包括在最长持续时间中。

例如，在下面的数据块中，在 8 秒语音中出现由逗号和句号引起的 600 毫秒中断：

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

## 在句子之间添加停顿

```
<s>
```

生成式、长篇、神经和标准 TTS 格式都支持此标签。

要在行与行或句子与句子之间添加停顿，请使用 `<s>` 标签。使用此标签的效果与以下情况效果相同：

- 用句点 (.) 结束一个句子
- 利用 `<break strength="strong"/>` 指定停顿

与 `<break>` 标签不同，`<s>` 标签要放在句子两端。对于按行朗读的文本这种语音合成方法很有用，例如诗歌。

在以下示例中，`<s>` 标签在第一句和第二句之后加入一个短的停顿。最后一个句子没有 `<s>` 标签，但它后面也有一个短停顿，因为它以句点结尾。

```
<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>
```

## 控制如何朗读特殊的词语类型

### <say-as>

生成式、长篇、神经和标准 TTS 引擎支持 <say-as> 标签。但请注意，如果 Amazon Polly 使用神经语音并在运行时遇到带有 characters 选项的 <say-as> 标签，则将使用相关的标准语音合成受影响的句子。但是，受影响的句子仍然会按照使用神经语音的情况收费。

使用具有 interpret-as 属性的 <say-as> 标签，告知 Amazon Polly 如何朗读某些字符、词语和数字。这可让您提供更多上下文，以消除 Amazon Polly 呈现文本方法中的任何歧义。

<say-as> 标签使用一个属性 interpret-as，该属性使用若干可能可用的值。每个属性使用相同的语法：

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

interpret-as 具有以下可用值：

- characters 或 spell-out：拼出文本中的每个字母，如所示。a-b-c

#### Note

目前，神经语音不支持此选项。如果您使用神经语音，并且 Amazon Polly 在运行时遇到此 SSML 代码，则将使用相关的标准语音合成受影响的句子。但请注意，这句话仍然会按照使用神经语音的情况收费。

- cardinal 或 number：将数字文本解释为基数，如 1,234。
- ordinal：将数字文本解释为序数，如第 1,234。
- digits：单独拼读每个数字，如 1-2-3-4。
- fraction：将数字文本解释为分数。这可用于真分数（如  $\frac{3}{20}$ ）和带分数（如  $2\frac{1}{2}$ ）。有关更多信息，请参阅下文。
- unit：将数字文本解释为度量单位。该值应是后跟一个单位的数字或分数，且中间没有空格（如 1/2inch），或者只后跟一个单位（如 1meter）。
- date：将文本解释为日期。必须使用格式属性指定日期格式。有关更多信息，请参阅下文。
- time：将数字文本解释为由分和秒组成的持续时间，例如 1'21"。
- address：将文本解释为街道地址的一部分。

- `expletive` : 用哔哔声盖过标签中包含的内容。
- `telephone` : 将数字文本解释为 7 位数或 10 位数的电话号码，例如 2025551212。您还可以使用此值处理电话分机，例如 2025551212x345。有关更多信息，请参阅下文。

#### Note

目前 `telephone` 选项仅适用于部分语言。但是，它适用于说英语变体 ( `en-au`、`en-GB`、`en-in`、`en-US` 和 )、西班牙语变体 ( `es-es`、`es-mx` 和 `es-US en-GB-WLS` )、法语变体 ( `fr-fr` 和 `fr-ca` ) 和葡萄牙语变体 ( `pt-br` 和 `pt-pt` )，以及德语 ( `de-de` )、意大利语 ( `it-it` )、日语 ( `ja-jp` ) 和俄语 ( `ru-jp` ) 的声音 ru)。另请注意，在某些情况下，阿拉伯语 ( `arb` ) 等语言会自动处理设置为电话号码的数字，因此实际上并没有实施 `telephone` SSML 标签。

## 分数

Amazon Polly 会将具有 `interpret-as="fraction"` 属性的 `say-as` 标签中的值视为一般分数。分数采用以下语法形式：

- 分数

语法：*cardinal number*/*cardinal number*，例如 2/9。

例如：`<say-as interpret-as="fraction">2/9</say-as>` 的发音为 "two ninths"。

- 非负带分数

语法：*cardinal number*+*cardinal number*/*cardinal number*，例如 3+1/2。

例如，`<say-as interpret-as="fraction">3+1/2</say-as>` 的发音为 "three and a half"。

#### Note

“3”和“1/2”之间必须有一个 +。Amazon Polly 不支持不带 + 的混合数字，例如“3 1/2”。

## 日期

如果将 `interpret-as` 设为 `date`，您还需要指明日期格式。

它使用以下语法：

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

例如：

```
<say-as interpret-as="date" format="mdy">12-31-1900</say-as>.  
</say-as>
```

下列格式可以与 date 属性一起使用。

- mdy: Month-day-year.
- dmy: Day-month-year.
- ymd: Year-month-day.
- md：月-日。
- dm：日-月。
- ym：年-月。
- my：月-年。
- d: 天。
- m: 月。
- y: 年份。
- yyyyymmdd: Year-month-day。如果您使用这种格式，则可以使用问号让 Amazon Polly 跳过部分日期。

例如，Amazon Polly 将以下内容呈现为“9 月 22 日”：

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

不需要 Format。

电话

即使没有 <say-as> 标签，Amazon Polly 也会尝试根据文本的格式正确地解释您提供的文本。例如，如果文本中包含“202-555-1212”，Amazon Polly 会将它解释为 10 位数的电话号码，并分别朗读每个数字，并在每个连字符处短暂停顿。在这种情况下，您不需要使用 <say-as interpret-

as="telephone">。但是，如果您提供的文本是“2025551212”，希望 Amazon Polly 将它朗读为电话号码，则需要指定 `<say-as interpret-as="telephone">`。

解释每种元素的逻辑是语言特定的。例如，在美国英语和英国英语中，电话号码的发音方法不同（在英国英语中，同一数字会组成一组，例如“两个五”或“三个四”）。要了解差别，请使用美国语音和英国语音测试以下示例：

```
<say-as interpret-as="telephone">2122241555</say-as>
```

## 首字母缩略词和缩写的发音

`<sub>`

生成式、长篇、神经和标准 TTS 格式都支持此标签。

使用具有 `<sub>` 属性的 `alias` 标签，用另一个词（或发音）替换选定文本，例如首字母缩略词或缩写。

使用以下语法：

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

在以下示例中，将用“Mercury”替换元素的化学符号，以使音频内容更加清晰。

```
<sub alias="Mercury">Hg</sub>, because it looks so shiny.
```

## 指定词性以改善发音

`<w>`

生成式、长篇、神经和标准 TTS 格式都支持此标签。

可使用 `<w>` 标签来自定义词语的发音，方式是指定词语的词性或替代含义。可使用 `role` 属性执行此操作。

此标签使用以下语法：

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

下列值可以用于 role 属性：

要指定词性，请执行以下操作：

- amazon:VB：将单词解释为动词（一般现在时）。
- amazon:VBD：将单词解释为过去时动词。
- amazon:DT：将单词解释为限定词。
- amazon:IN：将单词解释为介词。
- amazon:JJ：将单词解释为形容词。
- amazon:NN：将单词解释为名词。

例如，美国英语对单词“read”的发音会根据标签的不同有所变化，具体取决于词性：

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

要指定特定含义：

- amazon:DEFAULT：使用单词的默认时态。
- amazon:SENSE\_1：在现在时中使用单词的非默认时态。例如，名词“bass”的发音不同，具体取决于其含义。默认含义是音域的最低部分。替代含义是一种淡水鱼，也称作“bass”，但发音不同。使用 <w role="amazon:SENSE\_1">bass</w> 为音频文本呈现非默认发音（淡水鱼）。

如果您合成以下内容，则会听出这种发音和含义上的差异：

```
<speack>
  Depending on your meaning, the word <say-as interpret-as="characters">bass</say-
  as>
  may be interpreted as either a musical element: bass, or as its alternative
  meaning,
```

```
a freshwater fish <w role="amazon:SENSE_1">bass</w>.  
</speak>
```

### Note

一些语言可能有支持的词性的其他选择。

## 添加呼吸音

`<amazon:breath>` 和 `<amazon:auto-breaths>`

只有标准 TTS 格式支持此标签。

自然声音语音包括正确说出的字词和呼吸音。通过向合成语音添加呼吸音，可以使合成语音听起来更自然。`<amazon:breath>` 和 `<amazon:auto-breaths>` 标签可提供呼吸。您有以下选项：

- 手动模式：您可以在文本中设置呼吸音的位置、长度和音量
- 自动模式：Amazon Polly 自动将呼吸音插入语音输出
- 混合模式：由您和 Amazon Polly 共同添加呼吸音

### 手动模式

在手动模式下，可将 `<amazon:breath/>` 标签放在要插入呼吸的输入文本中。您可以分别使用 `duration` 和 `volume` 属性自定义呼吸的长度和音量：

- `duration`：控制呼吸的长度。有效值为：`default`, `x-short`, `short`, `medium`, `long`, `x-long`。默认值为 `medium`。
- `volume`：控制呼吸音有多大声。有效值为：`default`, `x-soft`, `soft`, `medium`, `loud`, `x-loud`。默认值为 `medium`。

### Note

每个属性值的确切长度和音量取决于使用的具体 Amazon Polly 语音。

要使用默认值设置呼吸音，请不带属性使用 `<amazon:breath/>`。

例如，要使用属性将呼吸的持续时间和音量设置为中等，可以按如下方式设置属性：

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

要使用默认值，只需使用标签：

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

您可以在一个段落内添加个别呼吸音，如下所示：

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

## 自动模式

在自动模式下，您使用 `<amazon:auto-breaths>` 标签来告知 Amazon Polly 以适当的时间间隔自动产生呼吸音。您可以设置间隔的频率、音量和持续时间。将 `</amazon:auto-breaths>` 标签放在要应用自动呼吸的文本的开头，然后将结束标签放在文本末尾。

### Note

与手动模式标签 `<amazon:breath/>` 不同，`<amazon:auto-breaths>` 标签必须具有结束标签 (`</amazon:auto-breaths>`)。

您可以将以下可选属性与 `<amazon:auto-breaths>` 标签结合使用：

- `volume` : 控制呼吸音有多大声。有效值为: `default`, `x-soft`, `soft`, `medium`, `loud`, `x-loud`。默认值为 `medium`。
- `frequency` : 控制文本中出现呼吸音的频率。有效值为: `default`, `x-low`, `low`, `medium`, `high`, `x-high`。默认值为 `medium`。
- `duration` : 控制呼吸的长度。有效值为: `default`, `x-short`, `short`, `medium`, `long`, `x-long`。默认值为 `medium`。

默认情况下，呼吸音的频率取决于输入文本。但是，呼吸音通常发生在逗号和句点之后。

以下示例显示如何使用 `<amazon:auto-breaths>` 标签。要确定对您的内容使用哪些选项，请将适用的示例复制到 Amazon Polly 控制台并听其中的差异。

- 不带可选参数使用自动化模式。

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

- 将自动化模式与音量控制结合使用。未指定的参数 (`duration` 和 `frequency`) 设置为默认值 (`medium`)。

```
<speack>
  <amazon:auto-breaths volume="x-soft">Amazon Polly is a service that turns text
  into lifelike
  speech, allowing you to create applications that talk and build entirely new
  categories of
  speech-enabled products. Amazon Polly is a text-to-speech service, that uses
  advanced deep
  learning technologies to synthesize speech that sounds like a human voice. With
  dozens of
```

```
lifelike voices across a variety of languages, you can select the ideal voice
and build speech-
enabled applications that work in many different countries.</amazon:auto-
breaths>
</speak>
```

- 将自动化模式与频率控制结合使用。未指定的参数 (duration 和 volume) 设置为默认值 (medium)。

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

- 将自动化模式与多个参数结合使用。对于未指定的 Duration 参数，Amazon Polly 将使用默认值 (medium)。

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

## 新闻播音员风格

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

新闻播音员风格仅适用于 Matthew 或 Joanna 语音，这些语音仅提供美式英语 ( en-US ) 版；适用于 Lupe 语音，该语音提供美国西班牙语 ( es-US ) 版；还适用于 Amy 语音，该语音提供英式英语 ( en-GB ) 版。仅在使用 Neural 格式时支持。

要使用新闻播音员风格，请使用 SSML 标签和以下语法：

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

例如，您可以通过 Amy 语音使用新闻播音员风格，如下所示：

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

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

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

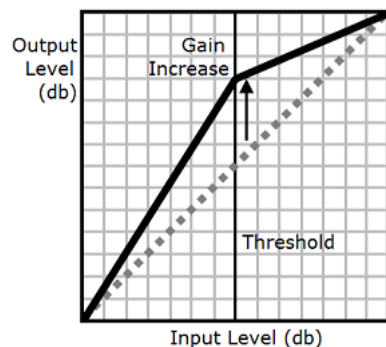
## 添加动态范围压缩

`<amazon:effect name="drc">`

长篇、神经和标准 TTS 格式都支持此标签。

根据音频文件中使用的文本、语言和语音，声音范围从轻柔到响亮。环境声音，如一辆正在行驶的车辆的声音，通常可以掩盖较轻柔的声音，从而使得音轨难以听清。要增强音频文件中某些声音的音量，请使用动态范围压缩 (drc) 标签。

drc 标签可为您的音频设置中等“响度”阈值，并且围绕该阈值增大声音的音量 ( 增益 )。它将应用最接近于该阈值的最大增益增加，而且增益增加远远小于该阈值。



这使得中等声音在嘈杂环境中更易于听见，从而使整个音频文件更清晰。

drc 标签是一个布尔值参数（既可能存在，也可能不存在）。它使用语法 `<amazon:effect name="drc">` 且以 `</amazon:effect>` 结束。

您可以结合使用 drc 标签和 Amazon Polly 支持的任何语音或语言。可以将该标签应用于整个录音部分，或仅应用于几个词。例如：

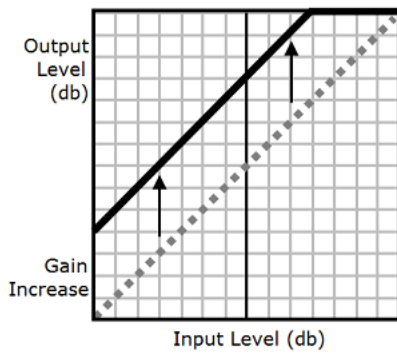
```
<speach>
  Some audio is difficult to hear in a moving vehicle, but <amazon:effect
  name="drc"> this audio
  is less difficult to hear in a moving vehicle.</amazon:effect>
</speach>
```

### Note

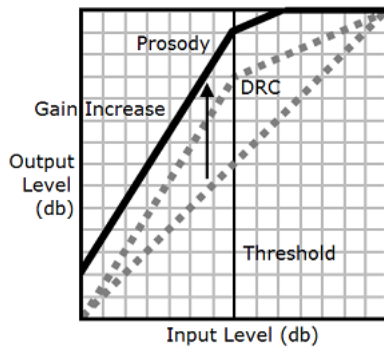
当您在 `amazon:effect` 语法中使用“drc”时，它是区分大小写的。

## 结合使用 **drc** 和 **prosody volume** 标签

如下图所示，prosody volume 标签均匀增大整个音频文件的音量，从原始音量级别（虚线）增大到调节后的音量级别（实线）。要进一步增大该文件中特定部分的音量，请结合使用 drc 标签和 prosody volume 标签。结合使用标签不会影响 prosody volume 标签的设置。



当结合使用 `drc` 和 `prosody volume` 标签时，Amazon Polly 会首先应用 `drc` 标签，以增大中等声音（这些声音接近阈值）。然后，再应用 `prosody volume` 标签，进一步均匀增大整个音轨的音量。



要结合使用这两个标签，可将一个标签嵌入另一个标签中。例如：

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

在本文中，`prosody volume` 标签将整段的音量增大为“响亮”。`drc` 标签增强第二句内中等值的音量。

### **Note**

结合使用 `drc` 和 `prosody volume` 标签时，请使用标准 XML 实践来嵌套标签。

## 柔和地朗读

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

目前只有标准 TTS 格式支持此标签。

要指定应使用 softer-than-normal 语音朗读输入文本，请使用 `<amazon:effect phonation="soft">` 标签。

使用以下语法：

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

例如，您可以将此标签用于 Matthew 语音，如下所示：

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

## 控制音质

```
<亚马逊 vocal-tract-length:effect >
```

目前只有标准 TTS 格式支持此标签。

音色是声音的音质，可帮助您区分不同的声音，即使当这些声音具有相同的音高和响度时。影响到语音音色的最重要生理特征之一是声道的长度。声道是空气从声带的顶部直到嘴唇边缘跨越的一个腔体。

要在 Amazon Polly 中控制输出语音的音色，请使用 `vocal-tract-length` 标签。此标签可更改发言者声道的长度，这听起来类似于更改发言者的身高。增加 `vocal-tract-length` 时，发言者听上去像是个子更高。减小时，发言者听上去像是个子更矮。您可以将此标签与 Amazon Polly Text-to-Speech 作品集中的任何声音配合使用。

要更改音色，请使用以下值：

- `+n%` 或 `-n%`：按当前语音的相对百分比进行更改来调整声道长度。例如，`+4%` 或 `-2%`。有效值范围为 `+100%` 至 `-50%`。此范围之外的值将被剪辑。例如，`+111%` 听起来像 `+100%`，`-60%` 听起来像 `-50%`。

- n% : 按当前语音的声道长度的绝对百分比来更改声道长度。例如, 110% 或 75%。110% 的绝对值等同于 +10% 的相对值。100% 的绝对值等同于当前语音的默认值。

以下示例演示了如何更改声道长度来更改音色 :

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

### 结合使用多个标签

您可以结合使用 `vocal-tract-length` 标签和 Amazon Polly 支持的任何其他 SSML 标签。由于音色 (声道长度) 与音高具有紧密联系, 通过结合使用 `vocal-tract-length` 和 `<prosody pitch>` 标签, 您可能会得到最佳效果。为了生成最真实的语音, 建议您对这两个标签使用不同的更改百分比。试用各种组合, 以得到希望的最佳效果。

以下示例演示如何结合使用标签。

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

## 轻读

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

目前只有标准 TTS 格式支持此标签。

此标签表示输入文本应使用低声而不是正常语音说出。这可以与 Amazon Polly Text-to-Speech 产品组合中的任何声音一起使用。

它使用以下语法：

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

例如：

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

在这种情况下，由所选的 Amazon Polly 语音说出的合成语音会以低声方式呈现，但短语“她说”是正常语音。

您可以通过将韵律结构速度降低最多 10% 来增强“低声”效果，具体视您期望的效果而定。

例如：

```
<speack>  
  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>  
</speack>
```

在为低声语音生成语音标记时，音频流还必须包括低声语音，以确保语音标记与音频流匹配。

## 管理词典

借助发音词典，您能够自定义单词的发音。Amazon Polly 提供可用于在 AWS 区域中存储词典的 API 操作。然后，这些词典特定于该特定区域。您可以使用该 SynthesizeSpeech 操作，在合成文本时使用该区域的一个或多个词典。此操作可在合成开始之前将指定的词典应用于输入文本。有关更多信息，请参阅 [SynthesizeSpeech](#)。

### Note

这些词典必须符合发音词典规范 (PLS) W3C 建议。有关更多信息，请参阅 W3C 网站上的 [发音词典规范 \(PLS\) 1.0 版](#)。

以下是将语音合成引擎与词典配合使用方法的例子：

- 普通单词中，有时字母会被数字代替，如“g3t sm4rt”(get smart)。人类能够正确读出这些单词。然而，文本到语音 (TTS) 引擎是按字面意思读取文本，完全按照名字的拼写发音。这种情况下，您可以使用 Amazon Polly，利用词典来自定义合成语音。在此示例中，您可以在词典中为单词“g3t sm4rt”指定别名 (get smart)。
- 您的文本可能包含一个首字母缩略词，例如 W3C。您可以使用词典来定义单词 W3C 的别名，以便以完整的展开形式 (万维网联盟) 读取。

词典能够让您更好地控制 Amazon Polly 如何发出所选择语言中不常见的发音。例如，您可以使用音标指定发音。有关更多信息，请参阅 W3C 网站上的 [发音词典规范 \(PLS\) 1.0 版](#)。

### 主题

- [使用多个词典](#)
- [上传词典](#)
- [应用词典 \(合成语音\)](#)
- [在控制台上筛选词典列表](#)
- [在控制台上下载词典](#)
- [删除词典](#)

## 使用多个词典

您最多可以为您的文本应用五个词典。如果同一个字素出现在您应用于文本的多个词典中，则它们的应用顺序可能会对生成语音产生影响。例如，给定以下文本，“你好，我叫 Bob。”且两个不同的词典中的两个词素都使用了字素 Bob。

### LexA

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

### LexB

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

如果词典以 LexA 然后 LexB 的顺序列出，合成的语音将是“你好，我叫罗伯特”。如果它们按 LexB 然后 LexA 的顺序列出，合成语音则是“你好，我叫 Bobby”。

Example– 先应用 LexA，然后应用 LexB

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

语音输出：“你好，我叫 Robert。”

Example– 先应用 LexB，然后应用 LexA

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

bobBA.mp3

语音输出：“你好，我叫 Bobby。”

有关使用 Amazon Polly 控制台应用词典的信息，请参阅 [应用词典 \(合成语音\)](#)。

## 上传词典

您使用的词典必须符合发音词典规范 ( PLS ) W3C 建议。有关更多信息，请参阅 W3C 网站上的 [发音词典规范 \(PLS\) 1.0 版](#)。

### Console - Lexicons tab

要使用发音词典，您必须先将其上传。您可以从控制台中的两个位置上传词典：Text-to-Speech 选项卡和 Lexicons 选项卡。

以下过程描述如何添加词典，您可以使用这些词典来自定义所选语言的不常见单词和短语如何发音。

#### 从词典选项卡添加词典

1. 登录到 AWS 管理控制台 并打开 Amazon Polly 控制台，网址：<https://console.aws.amazon.com/polly/>。
2. 选择 Lexicons 选项卡。
3. 选择上传词典。
4. 为词典提供名称，然后使用选择词典文件来查找要上传的词典。您只能上传扩展名为 .pls 或 .xml 的 PLS 文件。
5. 选择上传词典。如果相同名称的词典（无论是 .pls 或 .xml 文件）存在，上传词典会覆盖现有词典。

### Console - TTS tab

#### 从“文本转语音”选项卡添加词典

1. 登录到 AWS 管理控制台 并打开 Amazon Polly 控制台，网址：<https://console.aws.amazon.com/polly/>。
2. 选择文本到语音转换选项卡。
3. 展开其他设置，打开自定义发音，然后选择上传词典。

4. 为词典提供名称，然后使用选择词典文件来查找要上传的词典。您只能使用扩展名为 .pls 或 .xml 的 PLS 文件。
5. 选择上传词典。如果相同名称的词典（无论是 .pls 或 .xml 文件）存在，上传词典会覆盖现有词典。

## AWS CLI - one lexeme

借助 Amazon Polly，可以在特定 AWS 区域内使用 [PutLexicon](#) 为您的账户存储发音词典。然后，在服务开始合成文本之前，您可以在 [SynthesizeSpeech](#) 请求中指定一个或多个已存储的词典。有关更多信息，请参阅 [管理词典](#)。

请考虑以下兼容 W3C PLS 的词典。

```
<?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>
```

请注意以下几点：

- 在 <lexicon> 元素中指定的两个属性：
  - 该 xml:lang 属性指定应用该词典的语言代码 en-US。如果您在 SynthesizeSpeech 调用中指定的语音具有相同的语言代码 (en-US)，则 Amazon Polly 可以使用此示例词典。

### Note

您可以使用 DescribeVoices 操作以查找与语音关联的语言代码。

- `alphabet` 属性指定了 IPA，这意味着使用国际语音字母表 (IPA) 字母表发音。IPA 是用于书写发音的字母之一。Amazon Polly 还支持拓展音标字母评估法 (X-SAMPA)。
- `<lexeme>` 元素用于描述 `<grapheme>` (即单词的文本表示) 和 `<alias>` 之间的映射。

要测试此词典，请执行以下操作：

1. 将该词典保存为 `example.pls`。
2. 在 `us-east-2` 地区，运行 `put-lexicon` AWS CLI 命令来存储词典 (使用名称 `w3c`)。

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

3. 运行 `synthesize-speech` 命令以将示例文本合成为音频流 (`speech.mp3`)，并指定可选 `lexicon-name` 参数。

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

4. 播放生成的 `speech.mp3`，注意单词 `W3C` 在文本已替换为万维网联盟。

上述示例词典使用别名。未使用的词典中提到的 IPA 字母。以下词典使用带 IPA 字母 `<phoneme>` 元素指定语音发音。

```
<?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>

```

按照相同步骤测试此词典。请确保您指定的输入文本中有单词“pecan”（山核桃）（例如，“Pecan pie is delicious”（山核桃馅饼很好吃））。

有关 PutLexicon API 操作的附加代码示例，请参阅以下资源：

- Java 示例：[PutLexicon](#)
- Python (Boto3) 示例：[PutLexicon](#)

## AWS CLI - multiple lexemes

借助 Amazon Polly，可以在特定 AWS 区域内使用 [PutLexicon](#) 为您的账户存储发音词典。然后，在服务开始合成文本之前，您可以在 [SynthesizeSpeech](#) 请求中指定一个或多个已存储的词典。有关更多信息，请参阅 [管理词典](#)。

在此示例中，您在词典中指定的词素仅用于输入文本，以进行合成。考虑以下词典：

```

<?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>

```

词典指定了三个词素，其中两个词素为字素 W3C 定义了一个别名，如下所示：

- 第一个 <lexeme> 元素定义了一个别名（万维网联盟）。
- 第二个 <lexeme> 定义了一个替代别名 (WWW Consortium)。

Amazon Polly 使用第一个别名替换词典中的任意给定字素。

第三个 <lexeme> 为单词 Consortium 定义了一个替换词 (Community)。

首先，让我们来测试此词典。假设您想要将以下示例文本合成到音频文件 (speech.mp3)，并且在 SynthesizeSpeech 的调用中指定词典。

```
The W3C is a Consortium
```

SynthesizeSpeech 首先按如下所示应用词典：

- 根据第一个词素，单词 W3C 被修改为 World Wide Web Consortium。修改后的文本如下所示：

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

- 第三个词素中定义的别名仅适用于作为原始文本一部分的单词 Consortium，产生以下文本：

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

可以使用 AWS CLI 对此进行测试，如下所示：

1. 将该词典保存为 example.pls。
2. 在 us-east-2 地区，运行 put-lexicon 命令来存储词典 (使用名称 w3c)。

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

3. 运行 list-lexicons 命令以验证 w3c 词典在返回的词典列表中。

```
aws polly list-lexicons
```

4. 运行 synthesize-speech 命令以将示例文本合成为音频文件 (speech.mp3)，并指定可选 lexicon-name 参数。

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

5. 播放生成的 `speech.mp3` 文件，以验证合成语音是否反映了文本的更改。

有关 PutLexicon API 操作的附加代码示例，请参阅以下资源：

- Java 示例：[PutLexicon](#)
- Python (Boto3) 示例：[PutLexicon](#)

## 应用词典（合成语音）

您使用的词典必须符合发音词典规范 ( PLS ) W3C 建议。有关更多信息，请参阅 W3C 网站上的[发音词典规范 \(PLS\) 1.0 版](#)。

### Console

以下过程将通过应用 `W3c.pls` 词典将“万维网联盟”替换为“W3C”演示如何将词典应用到您的输入文本。如果您对文本应用多个词典，则它们将以自上而下的顺序应用，第一个匹配优先于以后的匹配。仅当词典中指定的语言与所选择的语言相同时，词典才会应用于文本。

您可以将一个词典应用到纯文本或 SSML 输入。

### Example– 应用 W3C.pls 词典

要创建您在本练习中将需要用到的词典，请参阅[上传词典](#)。使用纯文本编辑器来创建主题顶部显示的 `W3C.pls` 词典。请记住您保存此文件的位置。

将 `W3C.pls` 词典应用到您的输入

在此示例中，我们介绍了使用词典用“万维网联盟”替代“W3C”的方法。对此练习的美国英语和另一语言结果与[在控制台上使用 SSML](#) 的同语言结果进行比较。

1. 登录到 AWS 管理控制台 并打开 Amazon Polly 控制台，网址：<https://console.aws.amazon.com/polly/>。

## 2. 请执行以下操作之一：

- 关闭 SSML，然后将此文本键入或粘贴到文本输入框中。

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

- 打开 SSML，然后将此文本键入或粘贴到文本输入框中。

```
<speack>He wasn't paying attention.<break time="1s"/>  
In the middle of the 10/3/2014 W3C meeting  
he shouted, "Score!" quite loudly.</speack>
```

3. 从语言列表中，选择英语（美国），然后选择您要用于此文本的语音。
4. 展开其他设置，然后打开自定义发音。
5. 从词典列表中选择 W3C（English, US）。

如果未列出 W3C（English, US）词典，请选择 Upload lexicon 并将其上传，然后从列表中选择该词典。要创建此词典，请参阅[上传词典](#)。

6. 要立即收听语音，请选择收听。
7. 将语音保存到文件中
  - a. 选择下载。
  - b. 要更改到不同的文件格式，请打开语音文件格式设置，选择所需文件格式，然后选择下载。

重复上述步骤，但选择不同的语言并注意输出中的差异。

## AWS CLI

在对 SynthesizeSpeech 的调用中，您可以指定多个词典。在这种情况下，指定的第一个词典（按照从左到右的顺序）覆盖前面的任意词典。

请考虑以下两个词典。请注意，每个词典都为同一个字素 W3C 描述了不同的别名。

- 词典 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>
```

- 词典 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>
```

假设您分别将这些词典存储为 w3c 和 w3cAlternate。如果在 SynthesizeSpeech 调用中指定词汇符号（先是 w3c 然后是 w3cAlternate），则第一个词典中定义的 W3C 的别名优先于第二个。要测试词典，请执行以下操作：

1. 在本地将词典保存在名为 w3c.pls 和 w3cAlternate.pls 的文件中。
2. 使用 put-lexicon AWS CLI 命令上传这些词典。
  - 上传 w3c.pls 词典并将其存储为 w3c。

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

- 在服务中将 `w3cAlternate.pls` 词典上传为 `w3cAlternate`。

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

3. 运行 `synthesize-speech` 命令以将示例文本合成为音频流 (`speech.mp3`)，并使用 `lexicon-name` 参数指定两个词典。

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

4. 测试生成的 `speech.mp3`。具体如下：

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

## 在控制台上筛选词典列表

以下过程介绍如何筛选词典列表，以便仅显示所选语言的词典。

### Console

按语言筛选列出的词典

1. 登录到 AWS 管理控制台 并打开 Amazon Polly 控制台，网址：<https://console.aws.amazon.com/polly/>。
2. 选择 Lexicons 选项卡。
3. 选择任何语言。
4. 从语言列表中，选择要筛选的语言。

该列表仅显示所选语言的词典。

## AWS CLI

Amazon Polly 提供了 [ListLexicons](#) API 操作，您可以使用该操作获取您账户特定 AWS 区域内的发音词典的列表。以下 AWS CLI 调用列出了您账户中 us-east-2 区域中的词典。

```
aws polly list-lexicons
```

以下是一个示例响应，其中显示两个名为 w3c 和 tomato 的词典。对于每个词典，响应都会返回元数据，例如应用词典的语言代码、词典中定义的语素数量、词典大小（以字节为单位）等。语言代码描述了一个词典中定义的词素适用的语言和区域设置。

```
{
  "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"
    }
  ]
}
```

以下资源包含有关 ListLexicons 操作的附加信息：

- Java 示例：[ListLexicons](#)

- Python (Boto3) 示例：[ListLexicon](#)

## 在控制台上下载词典

以下过程介绍如何下载一个或多个词典。您可以在文件中添加、删除或修改词典条目，然后重新上传词典中的条目，以使您的词典保持最新。

### Console

#### 下载一个或多个词典

1. 登录到 AWS 管理控制台 并打开 Amazon Polly 控制台，网址：<https://console.aws.amazon.com/polly/>。
2. 选择 Lexicons 选项卡。
3. 选择您要下载的词典。
  - a. 要下载单个词典，请从列表中选择其名称。
  - b. 要将多个词典下载为单个压缩归档文件，请在列表选中您要下载的每个条目旁边的复选框。
4. 选择下载。
5. 打开您要下载词典的文件夹。
6. 选择保存。

### AWS CLI

Amazon Polly 提供了 [GetLexicon](#) API 操作，以检索您存储在账户特定区域中的发音词典的内容。

以下 `get-lexicon` AWS CLI 命令检索 `example` 词典的内容。

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

如果您的账户中还没有存储词典，您可以使用 `PutLexicon` 操作来存储。有关更多信息，请参阅 [上传词典](#)。

以下为示例响应。除了词典内容之外，响应还会返回元数据，例如应用词典的语言代码、词典中定义的词素数量、资源的亚马逊资源名称 (ARN) 以及词典的大小 (字节)。该 `LastModified` 值是 Unix 时间戳。

```
{
  "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
  }
}
```

以下资源包含 GetLexicon 操作的附加代码示例：

- Java 示例：[GetLexicon](#)
- Python (Boto3) 示例：[GetLexicon](#)

## 删除词典

以下过程介绍如何删除词典。删除词典后，您必须将其重新添加，然后才能再次使用。您可以通过选择单个词典旁边的复选框同时删除一个或多个词典。

### Console

#### 删除词典

1. 登录到 AWS 管理控制台 并打开 Amazon Polly 控制台，网址：<https://console.aws.amazon.com/polly/>。
2. 选择 Lexicons 选项卡。
3. 选择要从列表中删除的一个或多个词典。
4. 选择删除。
5. 输入确认文本，然后选择删除以从“区域”中删除词典，或选择取消保留词典。

## AWS CLI

Amazon Polly 提供了 [DeleteLexicon](#) API 操作，可将发音词典从您账户中的特定 AWS 区域删除。以下 AWS CLI 可删除指定的词典。

以下 AWS CLI 示例针对 Linux、Unix 和 macOS 编排了格式。对于 Windows，请将每行末尾的反斜杠 (\) Unix 行继续符替换为脱字号 (^) 并在输入文本周围使用全角引号 (")，内部标签使用单引号 (')。

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

以下资源包含有关 DeleteLexicon 操作的附加信息：

- Java 示例：[DeleteLexicon](#)
- Python (Boto3) 示例：[DeleteLexicon](#)

## 长音频文件

要为大段文本创建 TTS 文件，请使用 Amazon Polly 的异步合成功能。这使用了三个 `SpeechSynthesisTask` APIs：

- `StartSpeechSynthesisTask`：启动新合成任务。
- `GetSpeechSynthesisTask`：返回有关以前提交的合成任务的详细信息。
- `ListSpeechSynthesisTasks`：列出所有已提交的合成任务。

`SynthesizeSpeech` 操作以近乎实时的方式生成音频，大多数情况下延迟较短。为此，此操作只能合成 3000 个字符。

Amazon Polly 的异步合成功能通过更改文档的合成和返回方式克服了处理较大文本文档的挑战。当通过使用 `StartSpeechSynthesisTask` 提交输入文本来发出合成请求时，Amazon Polly 会将请求排队，然后在系统资源可用时立即在后台异步处理这些请求。然后，Amazon Polly 将生成的语音或语音标记流直接上传到（必需的）Amazon Simple Storage Service (Amazon S3) 存储桶，并通过（可选的）SNS 主题通知您已完成文件的可用性。

通过这种方式，除近乎实时的处理之外的所有其他功能都可用于长度长达 100000 个计费字符（或总计 200000 个字符）的文本。

要使用此方法合成文档，您必须具有音频文件可保存到其中的可写 Amazon S3 存储桶。您可以通过提供可选的 SNS 主题标识符在合成音频准备就绪时收到通知。当合成任务完成后，Amazon Polly 将在该主题上发布消息。在合成任务未成功的情况下，此消息可能还包含有用的错误信息。为此，请确保创建合成任务的用户也可以发布到 SNS 主题。有关如何创建和订阅 SNS 主题的更多信息，请参阅 [Amazon SNS 文档](#)。

### 加密

您可以加密形式将输出文件存储在 S3 存储桶中（如果需要）。为此，请启用 [Amazon S3 存储桶加密](#)，它使用可用的最强数据块密码之一（256 位高级加密标准 (AES-256)）。

### 主题

- [为异步合成设置 IAM 策略](#)
- [创建长音频文件](#)

## 为异步合成设置 IAM 策略

要使用异步合成功能，您将需要允许以下操作的 IAM 策略：

- 使用新的 Amazon Polly 操作
- 写入到输出 S3 存储桶
- 发布到状态 SNS 主题 [可选]

以下策略仅授予异步合成所需的必要权限并可附加到 IAM 用户。

## 创建长音频文件

您可以通过 Amazon Polly 控制台来使用异步合成创建长语音，您可以通过 AWS CLI 来使用相同功能。这是通过使用与任何其他合成非常类似的文本到语音转换选项卡实现的。

### Console

还通过控制台提供了其他异步合成功能。S3 合成任务选项卡反映了 ListSpeechSynthesisTasks 功能，显示了保存到 S3 存储桶中的所有任务，并使您能够在需要进行筛选。单击特定的单个任务将显示其详细信息，反映 GetSpeechSynthesisTask 功能。

使用 Amazon Polly 控制台合成大文本


1. 登录 AWS 管理控制台 并打开 Amazon Polly 控制台，网址为。 <https://console.aws.amazon.com/polly/>
2. 选择文本到语音转换选项卡。如果合适，请选择长篇作为引擎。
3. 打开或关闭 SSML 后，在输入框中键入或粘贴您的文本。
4. 为您的文本选择语言、区域和语音。
5. 选择保存至 S3。

#### Note

如果文本长度超过实时 SynthesizeSpeech 操作的 3000 个字符限制，则下载和收听选项将灰显。


6. 控制台会打开一个表单，以便您可以选择存储输出文件的位置。

- a. 填写目标 Amazon S3 存储桶的名称。
- b. ( 可选 ) 填写输出的前缀键。

 Note

输出 S3 存储桶必须可写。

- c. 如果您希望在合成任务完成后收到通知，请提供可选的 SNS 主题标识符。

 Note

SNS 必须可由当前控制台用户发布才能使用此选项。有关更多信息，请参阅 [Amazon Simple Notification Service \(SNS\)](#)

- d. 选择保存至 S3。

检索有关您的语音合成任务的信息

1. 在控制台中，选择 S3 合成任务选项卡。
2. 将按日期顺序显示任务。要按状态筛选任务，请选择所有状态，然后选择要使用的状态。
3. 要查看特定任务的详细信息，请选择链接的任务 ID。

## AWS CLI

Amazon Polly 异步合成功能使用三个功能 `SpeechSynthesisTask` APIs 来处理大量文本：

- `StartSpeechSynthesisTask`：启动新合成任务。
- `GetSpeechSynthesisTask`：返回有关以前提交的合成任务的详细信息。
- `ListSpeechSynthesisTasks`：列出所有已提交的合成任务。

### 合成大量文本 (`StartSpeechSynthesisTask`)

当您要创建的音频文件大于使用实时 `SynthesizeSpeech` 可创建的文件时，请使用 `StartSpeechSynthesisTask` 操作。除了 `SynthesizeSpeech` 操作所需的参数外，`StartSpeechSynthesisTask` 还需要 Amazon S3 存储桶的名称。也可使用另外两个可选参数：输出文件的键前缀和 SNS 主题的 ARN ( 如果您希望收到有关任务的状态通知 )。

- `OutputS3BucketName` : 合成应上传到的 Amazon S3 存储桶的名称。此存储桶应与 Amazon Polly 服务位于同一区域中。此外，用于进行调用的 IAM 用户应具有对此存储桶的访问权限。[必需]
- `OutputS3KeyPrefix` : 输出文件的键前缀。如果您希望将输出语音文件保存在存储桶中自定义的类似于目录的键中，请使用此参数。[可选]
- `SnsTopicArn` : 要使用的 SNS 主题 ARN (如果您希望收到有关任务状态的通知)。此 SNS 主题应与 Amazon Polly 服务位于同一区域中。此外，用于进行调用的 IAM 用户应具有对此主题的访问权限。[可选]

例如，以下示例可用于在美国东部 ( 俄亥俄州 ) 区域运行该 `start-speech-synthesis-task` AWS CLI 命令：

以下 AWS CLI 示例是针对 Unix、Linux 和 macOS 进行格式化的。对于 Windows，请将每行末尾的反斜杠 (\) Unix 行继续符替换为脱字号 (^) 并在输入文本周围使用全角引号 ( “ ” )，内部标签使用单引号 ( ‘ ’ )。

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

这将生成与以下内容类似的响应：

```
"SynthesisTask":  
{  
  "OutputFormat": "mp3",  
  "OutputUri": "https://s3.us-east-2.amazonaws.com/your-bucket-name/optional/  
prefix/path/file.<task_id>.mp3",  
  "TextType": "text",  
  "CreationTime": [...],  
  "RequestCharacters": [...],  
  "TaskStatus": "scheduled",  
  "TaskId": [task_id],  
  "VoiceId": "Joanna"  
}
```

`start-speech-synthesis-task` 操作会返回几个新字段：

- `OutputUri`：输出语音文件的位置。
- `TaskId`：Amazon Polly 生成的语音合成任务的唯一标识符。
- `CreationTime`：初次提交任务时的时间戳。
- `RequestCharacters`：任务中的计费字符数。
- `TaskStatus`：提供有关提交的任务的状态的信息。

提交任务后，初始状态将显示 `scheduled`。当 Amazon Polly 开始处理任务后，状态将更改为 `inProgress`，稍后更改为 `completed` 或 `failed`。如果任务失败，则在调用 `GetSpeechSynthesisTask` 或 `ListSpeechSynthesisTasks` 操作时将返回错误消息。

任务完成后，会在 `OutputUri` 中指定的位置提供语音文件。

检索有关您的语音合成任务的信息

您可以使用 `GetSpeechSynthesisTask` 操作获取有关任务的信息，如错误、状态等。为此，您将需要 `task-id` 所返回的 `StartSpeechSynthesisTask`。

例如，以下示例可用于运行 `get-speech-synthesis-task` AWS CLI 命令：

```
aws polly get-speech-synthesis-task \  
--region us-east-2 \  
--endpoint-url "https:// polly.us-east-2.amazonaws.com/" \  
--task-id task identifier
```

您还可以使用 `ListSpeechSynthesisTasks` 操作列出在当前区域中运行的所有语音合成任务。

例如，以下示例可用于运行 `list-speech-synthesis-tasks` AWS CLI 命令：

```
aws polly list-speech-synthesis-tasks \  
--region us-east-2 \  
--endpoint-url "https:// polly.us-east-2.amazonaws.com/"
```

# Amazon Polly 中的配额

Amazon Polly 通过拒绝过多请求，对客户流量实行配额制。对于单个 AWS 账户，在单个区域内，使用标准语音的 SynthesizeSpeech 请求的默认配额为每秒 80 个事务 (tps)。如果限制没有增加，并且您使用标准语音每秒生成 100 个 SynthesizeSpeech 请求，则每秒 80 个请求将成功，而每秒 20 个请求将受到 Amazon Polly 的限制。此类请求将返回一个 HTTP 状态为 400 的响应和一个表示 ThrottlingException 的响应标头。Amazon Polly 还会根据请求速率对所有操作进行节流。

## 语音合成限制示例

- 合成英语字母表的前 24 个字母，一次只能合成一个字母。如果每个字母的合成时间少于 50 毫秒，操作限制为 8tps，则合成 24 个字母至少需要三秒钟。在此期间，您每秒最多可以合成八个字母。任何进一步的请求都将受到限制。由于请求持续的时间很短，因此它们将按顺序合成，不会重叠。
- 合成 16 个段落。如果每个段落在一秒或更短的时间内在客户端合成并完全接收且操作限制为八个并发请求，则合成所有 16 篇文章至少需要四秒钟。在第一秒内，您最多可以启动八个请求。在并发请求期间，由于并发限制，任何启动新合成的尝试都将受到限制。在第一批请求完成后，您可以在前两秒钟后合成剩余的八个段落。

在使用 Amazon Polly 时，请记住以下限制。

## 主题

- [支持的区域](#)
- [配额和节流率](#)
- [发音词典](#)
- [SynthesizeSpeech API 操作](#)
- [SpeechSynthesisTask API 操作](#)
- [语音合成标记语言 \(SSML\)](#)

## 支持的区域

有关提供 Amazon Polly 的 AWS 区域列表，请参阅中的 [Amazon Polly 终端节点和配额](#)。Amazon Web Services 一般参考

- 有关支持生成式语音的区域，请参阅[生成式语音](#)。
- 有关支持长篇语音的区域，请参阅[Long-form 音](#)。

- 有关支持神经语音的区域，请参阅神经 TTS 的 [the section called “特征和区域兼容性”](#)。

## 配额和节流率

下表定义了每个 Amazon Polly 操作的节流率。必要时，您可以使用 AWS 管理控制台 来请求增加可调整配额的配额。

操作	限制
词典	
DeleteLexicon	这些操作组合起来为每秒任意 2 个事务 (tps)。  允许的最大突增为 4 tps。
PutLexicon	
GetLexicon	
ListLexicons	
语音	
DescribeVoices	80 tps，突增限制为 100 tps
SynthesizeSpeech	生成式语音：8 tps  Long-form 语音：8 tps，爆发限制为 10 tps  神经语音：8 tps，突增限制为 10 tps  标准语音：80 tps，突增限制为 100 tps
StartSpeechSynthesisTask	生成式语音：1 tps  Long-form 声音：1 tps  神经语音：10 tps  标准语音：10 tps，突增限制为 12 tps
StartSpeechSynthesisStream	生成式语音：8 tps

操作	限制
GetSynthesizeSpeechTask 和 ListSynthesizeSpeechTask	总共最多允许 10 tps

## 并发请求

对于生成式语音，Amazon Polly 最多支持 26 个并发请求。对于长篇语音，Amazon Polly 最多支持 26 个并发请求。对于神经语音，Amazon Polly 支持 8tps，突发限制为 10tps，最多可处理 18 个并发请求。Amazon Polly 还支持对并发请求进行限制。对于标准语音，Amazon Polly 支持 80tps，最多可处理 80 个并发请求。

对于 StartSpeechSynthesisStream，Amazon Polly 最多支持 8 个并发请求。

## 减少节流的最佳实践

- 使用退避和抖动重试节流，以便您在短时间内分散负载，并不影响可用性的情况下处理意外的使用高峰。AWS 代码示例目录 已配置为在许多编程语言中默认执行此操作。请访问[功能重试行为](#)以查看详细信息。
- 使用 [Amazon Polly 指标](#)。Amazon Polly 会自动发布 CloudWatch 到以分析您的当前使用情况并预测使用量增长。

### Note

在申请增加配额（如适用）之前，请按照本页上的指南计算您的 tps 需求。为了保持较低的成本，Amazon Polly 仅根据客户需求确保所需的计算资源。

## 发音词典

- 您可以在每个账户中存储最多 100 个词典。
- 词典名称可以是不超过 20 个字符的字母数字字符串。
- 每个词典的大小最多为四万个字符。（请注意，词典的大小会影响 SynthesizeSpeech 操作的延迟。）
- 您可以在词典中为每个 <phoneme> 或 <alias> 替换指定最多 100 个字符。

有关使用词典的信息，请参阅 [管理词典](#)。

## SynthesizeSpeech API 操作

在估算 SynthesizeSpeech 的使用量时，请记住，Amazon Polly 生成的音频，尤其是用于交互式应用程序的音频，通常至少需要几秒钟才能播放。这会将请求速率降低至 SynthesizeSpeech，即便有大量并发使用者，也是如此。此外，Amazon Polly 会根据其合成的并发请求数量来限制 SynthesizeSpeech 请求。并发请求没有单独设置。并发请求限制值始终与允许的 tps 数量相同，并随着 tps 数量的增加而增加。

短篇故事示例应用程序。您可以使用 Amazon Polly 来构建一款可以播放一系列短篇故事的应用程序。使用这种应用程序，将会从第一个故事开始播放，然后播放下一个故事，依此类推，直到用户退出应用程序。每个故事的合成时间大约为 0.5 秒，播放时间为 10 秒。在这种情况下，使用者每使用应用程序 10 秒钟，就会调用 SynthesizeSpeech 一次。这相当于每 10 名同时使用应用程序的使用者每秒有一次调用。如果您有 1000 名使用者同时使用该应用程序，则 SynthesizeSpeech 的平均调用速率仅为每秒 100 个事务。

请注意以下与使用 SynthesizeSpeech API 操作相关的限制：

- 输入文本最大为 3000 个计费字符（总计 6000 个字符）。SSML 标签不会算作计费字符。
- 您可以指定最多 5 个要应用于输入文本的词典。
- 输出音频流（合成）限制为 10 分钟。达到此限制之后，将截断任何剩余语音。

有关更多信息，请参阅 [SynthesizeSpeech](#)。

### Note

使用 SynthesizeSpeech API 操作可绕过 StartSynthesizeSpeechTask API 操作的一些限制。有关更多信息，请参阅 [长音频文件](#)。

## SpeechSynthesisTask API 操作

请注意以下与使用 StartSpeechSynthesisTask、GetSpeechSynthesisTask 和 ListSpeechSynthesisTasks API 操作相关的限制：

- 输入文本最大为 100,000 个计费字符（总计 200,000 个字符）。SSML 标签不会算作计费字符。

- 您可以指定最多 5 个要应用于输入文本的词典。

## 语音合成标记语言 (SSML)

请注意以下与使用 SSML 有关的限制：

- `<audio>`、`<lexicon>`、`<lookup>` 和 `<voice>` 标签不受支持。
- `<break>` 元素可以指定的最大持续时间为 10 秒。
- `<prosody>` 标签不支持低于 -80% 的费率属性值。

有关更多信息，请参阅 [由 SSML 文档生成语音](#)。

# Amazon Polly 的示例代码和应用程序

本节提供了可供您探索 Amazon Polly 的代码示例和示例应用程序。

示例代码主题包含按编程语言整理的代码片段，这些代码片段被分成针对不同 Amazon Polly 功能的示例。示例应用程序主题包含按编程语言整理的应用程序，这些应用程序可独立地用于探索 Amazon Polly。

在开始使用这些示例前，我们建议您先阅读 [Amazon Polly 工作原理](#) 并遵守 [Amazon Polly 入门](#) 中所述的步骤。

## 主题

- [Java 示例](#)
- [Python 示例](#)
- [Java 示例](#)
- [Python 示例 \(HTML5 客户端和 Python 服务器\)](#)
- [iOS 示例](#)
- [Android 示例](#)

## Java 示例

以下代码示例显示如何使用基于 Java 的应用程序完成各种 Amazon Polly 任务。这些示例不是完整示例，但可以包含在使用 [适用于 Java 的 AWS SDK](#) 的更大的 Java 应用程序中。

## 代码段

- [DeleteLexicon](#)
- [DescribeVoices](#)
- [GetLexicon](#)
- [ListLexicons](#)
- [PutLexicon](#)
- [StartSpeechSynthesisTask](#)
- [语音标记](#)
- [SynthesizeSpeech](#)

## DeleteLexicon

以下 Java 代码示例显示如何使用基于 Java 的应用程序删除存储在 AWS 区域中的特定词典。已删除的词典不可用于语音合成，也不能使用 `GetLexicon` 或 `ListLexicon` API 检索。

有关此操作的更多信息，请参阅 [DeleteLexicon API 参考](#)。

### SDK v2

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
 */

package com.example.polly;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DeleteLexiconRequest;
import software.amazon.awssdk.services.polly.model.DeleteLexiconResponse;
import software.amazon.awssdk.services.polly.model.PollyException ;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteLexiconSample {

    public static void main(String args[]) {

        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        deleteLexicon(polly) ;
        polly.close();
    }
}
```

```
private static String LEXICON_NAME = "SampleLexicon";
public static void deleteLexicon(PollyClient client) {

    try {
        DeleteLexiconRequest deleteLexiconRequest = DeleteLexiconRequest.builder()
            .name(LEXICON_NAME).build();

        DeleteLexiconResponse deleteLexiconResult =
client.deleteLexicon(deleteLexiconRequest);

    } catch (PollyException e) {
        System.err.println("Exception caught: " + e);
        System.exit(1);
    }
}
}
```

## SDK v1

```
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

以下 Java 代码示例显示如何使用基于 Java 的应用程序生成可在请求语音合成时使用的语音列表。您可以选择指定语言代码以筛选可用语音。例如，如果您指定 en-US，则该操作将返回所有可用美国英语语音的列表。

有关此操作的更多信息，请参阅 [DescribeVoices API 参考](#)。

```
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

以下 Java 代码示例显示如何使用基于 Java 的应用程序生成存储在 AWS 区域中的特定发音词典的内容。

有关此操作的更多信息，请参阅 [GetLexicon API 参考](#)。

### SDK v2

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
 */

package com.example.polly;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.GetLexiconRequest;
import software.amazon.awssdk.services.polly.model.GetLexiconResponse;
import software.amazon.awssdk.services.polly.model.PollyException ;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetLexiconSample {

    public static void main(String args[]) {

        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create())
```

```
        .build();

        getLexicon(polly) ;
        polly.close();
    }

    private static String LEXICON_NAME = "SampleLexicon";
    public static void getLexicon(PollyClient client) {

        try {
            GetLexiconRequest getLexiconRequest = GetLexiconRequest.builder()
                .name(LEXICON_NAME).build();

            GetLexiconResponse getLexiconResult = client.getLexicon(getLexiconRequest);
            System.out.println("The name of the Lexicon is " +
getLexiconResult.lexicon().name());

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

## SDK v1

```
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

以下 Java 代码示例显示如何使用基于 Java 的应用程序生成存储在 AWS 区域中的发音词典的列表。

有关此操作的更多信息，请参阅 [ListLexicons](#) API 参考。

```
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())
                }
            } while (nextToken != null);
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

```
        + ", 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

以下 Java 代码示例显示如何使用基于 Java 的应用程序将发音词典存储在 AWS 区域中。

有关此操作的更多信息，请参阅 [PutLexicon API 参考](#)。

### SDK v2

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.polly;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.PutLexiconRequest;
import software.amazon.awssdk.services.polly.model.PutLexiconResponse;
import software.amazon.awssdk.services.polly.model.PollyException ;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class PutLexiconSample {

    public static void main(String args[]) {

        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        putLexicon(polly) ;
        polly.close();
    }

    private static 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 static String LEXICON_NAME = "SampleLexicon";
    public static void putLexicon(PollyClient client) {

        try {
            PutLexiconRequest putLexiconRequest = PutLexiconRequest.builder()
                .name(LEXICON_NAME).content(LEXICON_CONTENT).build();

            PutLexiconResponse putLexiconResult = client.putLexicon(putLexiconRequest);

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

## SDK v1

```
package com.amazonaws.polly.samples;
```

```
import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.AmazonPollyClientBuilder;
import com.amazonaws.services.polly.model.PutLexiconRequest;

public class PutLexiconSample {
    AmazonPolly client = AmazonPollyClientBuilder.defaultClient();

    private String LEXICON_CONTENT = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
        "<lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/pronunciation-lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " +
        "xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\" " +
        "alphabet=\"ipa\" xml:lang=\"en-US\">" +
        "<lexeme><grapheme>test1</grapheme><alias>test2</alias></lexeme>" +
        "</lexicon>";
    private String LEXICON_NAME = "SampleLexicon";

    public void putLexicon() {
        PutLexiconRequest putLexiconRequest = new PutLexiconRequest()
            .withContent(LEXICON_CONTENT)
            .withName(LEXICON_NAME);

        try {
            client.putLexicon(putLexiconRequest);
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

## StartSpeechSynthesisTask

以下 Java 代码示例演示如何使用基于 Java 的应用程序来合成长语音（多达十万个计费字符）并将其直接存储在 Amazon S3 存储桶中。

有关更多信息，请参阅 [StartSpeechSynthesisTask](#) API 参考。

### SDK v2

```
/*
    Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
    SPDX-License-Identifier: Apache-2.0
*/
```

```
package com.example.polly;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.*;

import java.time.Duration;
import org.awaitility.Durations;
import org.awaitility.Awaitility;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartSpeechSynthesisTaskSample {

    public static void main(String args[]) {

        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        startSpeechSynthesisTask(polly) ;
        polly.close();
    }

    private static final String PLAIN_TEXT = "This is a sample text to be
synthesized.";
    private static final String OUTPUT_FORMAT_MP3 = OutputFormat.MP3.toString();
    private static final String OUTPUT_BUCKET = "synth-books-buckets";
    private static final String SNS_TOPIC_ARN = "arn:aws:sns:eu-
west-2:123456789012:synthesize-finish-topic";
    private static final Duration SYNTHESIS_TASK_POLL_INTERVAL =
Durations.FIVE_SECONDS;
    private static final Duration SYNTHESIS_TASK_POLL_DELAY = Durations.TEN_SECONDS;
    private static final Duration SYNTHESIS_TASK_TIMEOUT = Durations.FIVE_MINUTES;
    public static void startSpeechSynthesisTask(PollyClient client) {
```

```

    try {
        StartSpeechSynthesisTaskRequest startSpeechSynthesisTaskRequest =
StartSpeechSynthesisTaskRequest.builder()

.outputFormat(OUTPUT_FORMAT_MP3).text(PLAIN_TEXT).textType(TextType.TEXT)

.voiceId(VoiceId.AMY).outputS3BucketName(OUTPUT_BUCKET).snsTopicArn(SNS_TOPIC_ARN)
        .engine("neural").build();

        StartSpeechSynthesisTaskResponse startSpeechSynthesisTaskResponse =
            client.startSpeechSynthesisTask(startSpeechSynthesisTaskRequest);
        String taskId = startSpeechSynthesisTaskResponse.synthesisTask().taskId();

        Awaitility.await().with()
            .pollInterval(SYNTHESIS_TASK_POLL_INTERVAL)
            .pollDelay(SYNTHESIS_TASK_POLL_DELAY)
            .atMost(SYNTHESIS_TASK_TIMEOUT)
            .until(
                () -> getSynthesisTaskStatus(client,
taskId).equals(TaskStatus.COMPLETED.toString())
            );

    } catch (PollyException e) {
        System.err.println("Exception caught: " + e);
        System.exit(1);
    }
}

private static String getSynthesisTaskStatus(PollyClient client, String taskId) {
    GetSpeechSynthesisTaskRequest getSpeechSynthesisTaskRequest =
GetSpeechSynthesisTaskRequest.builder()
        .taskId(taskId).build();
    GetSpeechSynthesisTaskResponse result =
client.getSpeechSynthesisTask(getSpeechSynthesisTaskRequest);
    return result.synthesisTask().taskStatusAsString();
}
}

```

## SDK v1

```
package com.amazonaws.parrot.service.tests.speech.task;
```

```
import com.amazonaws.parrot.service.tests.AbstractParrotServiceTest;
import com.amazonaws.services.polly.AmazonPolly;
import com.amazonaws.services.polly.model.*;
import org.awaitility.Duration;

import java.util.concurrent.TimeUnit;

import static org.awaitility.Awaitility.await;

public class StartSpeechSynthesisTaskSample {

    private static final int SYNTHESIS_TASK_TIMEOUT_SECONDS = 300;
    private static final AmazonPolly AMAZON_POLLY_CLIENT =
AmazonPollyClientBuilder.defaultClient();
    private static final String PLAIN_TEXT = "This is a sample text to be
synthesized.";
    private static final String OUTPUT_FORMAT_MP3 = OutputFormat.Mp3.toString();
    private static final String OUTPUT_BUCKET = "synth-books-buckets";
    private static final String SNS_TOPIC_ARN = "arn:aws:sns:eu-
west-2:123456789012:synthesize-finish-topic";
    private static final Duration SYNTHESIS_TASK_POLL_INTERVAL = Duration.FIVE_SECONDS;
    private static final Duration SYNTHESIS_TASK_POLL_DELAY = Duration.TEN_SECONDS;

    public static void main(String... args) {
        StartSpeechSynthesisTaskRequest request = new StartSpeechSynthesisTaskRequest()
            .withOutputFormat(OUTPUT_FORMAT_MP3)
            .withText(PLAIN_TEXT)
            .withTextType(TextType.Text)
            .withVoiceId(VoiceId.Amy)
            .withOutputS3BucketName(OUTPUT_BUCKET)
            .withSnsTopicArn(SNS_TOPIC_ARN)
            .withEngine("neural");

        StartSpeechSynthesisTaskResult result =
AMAZON_POLLY_CLIENT.startSpeechSynthesisTask(request);
        String taskId = result.getSynthesisTask().getTaskId();

        await().with()
            .pollInterval(SYNTHESIS_TASK_POLL_INTERVAL)
            .pollDelay(SYNTHESIS_TASK_POLL_DELAY)
            .atMost(SYNTHESIS_TASK_TIMEOUT_SECONDS, TimeUnit.SECONDS)
            .until(
```

```
        () ->
getSynthesisTaskStatus(taskId).equals(TaskStatus.Completed.toString())
        );
    }

    private static SynthesisTask getSynthesisTask(String taskId) {
        GetSpeechSynthesisTaskRequest getSpeechSynthesisTaskRequest = new
GetSpeechSynthesisTaskRequest()
            .withTaskId(taskId);
        GetSpeechSynthesisTaskResult result
=AMAZON_POLLY_CLIENT.getSpeechSynthesisTask(getSpeechSynthesisTaskRequest);
        return result.getSynthesisTask();
    }

    private static String getSynthesisTaskStatus(String taskId) {
        GetSpeechSynthesisTaskRequest getSpeechSynthesisTaskRequest = new
GetSpeechSynthesisTaskRequest()
            .withTaskId(taskId);
        GetSpeechSynthesisTaskResult result
=AMAZON_POLLY_CLIENT.getSpeechSynthesisTask(getSpeechSynthesisTaskRequest);
        return result.getSynthesisTask().getTaskStatus();
    }
}
}
```

## 语音标记

以下代码示例显示如何使用基于 Java 的应用程序为输入的文本合成语音标记。此功能使用 SynthesizeSpeech API。

有关此功能的更多信息，请参见[语音标记](#)。

有关 API 的更多信息，请参阅 [SynthesizeSpeech](#) API 参考。

### SDK v2

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
 */

package com.example.polly;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.*;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SpeechMarksSample {

    public static void main(String args[]) {

        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        speechMarksSample(polly) ;
        polly.close();
    }

    private static final String OUTPUT_FILE = "./speechMarks.json";
    public static void speechMarksSample(PollyClient client) {

        try {
            SynthesizeSpeechRequest speechMarksSampleRequest =
            SynthesizeSpeechRequest.builder()
                .outputFormat(OutputFormat.JSON)
                .speechMarkTypes(SpeechMarkType.VISEME, SpeechMarkType.WORD)
                .voiceId(VoiceId.JOANNA)
                .text("This is a sample text to be synthesized")
                .build();

            try (FileOutputStream outputStream = new FileOutputStream(new
            File(OUTPUT_FILE))) {
```

```

        ResponseInputStream<SynthesizeSpeechResponse> synthesizeSpeechResponse
= client
            .synthesizeSpeech(speechMarksSampleRequest);
        byte[] buffer = new byte[2 * 1024];
        int readBytes;

        try (InputStream in = synthesizeSpeechResponse){
            while ((readBytes = in.read(buffer)) > 0) {
                outputStream.write(buffer, 0, readBytes);
            }
        }
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
}

```

## SDK v1

```

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

以下 Java 代码示例演示如何使用基于 Java 的应用程序来合成包含较短文本的语音以进行近乎实时的处理。

有关更多信息，请参阅 [SynthesizeSpeech API 参考](#)。

```
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.")
            .withEngine("neural");

        try (FileOutputStream outputStream = new FileOutputStream(new
File(outputFileName))) {
            SynthesizeSpeechResult synthesizeSpeechResult =
client.synthesizeSpeech(synthesizeSpeechRequest);
            byte[] buffer = new byte[2 * 1024];
            int readBytes;

            try (InputStream in = synthesizeSpeechResult.getAudioStream()){
                while ((readBytes = in.read(buffer)) > 0) {
                    outputStream.write(buffer, 0, readBytes);
                }
            }
        } catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}
```

## Python 示例

以下代码示例显示如何使用基于 Python (boto3) 的应用程序完成各种使用 Amazon Polly 的任务。这些示例不是完整示例，但可以包含在使用[AWS SDK for Python \(Boto\)](#)的更大的 Python 应用程序中。

代码段

- [DeleteLexicon](#)
- [GetLexicon](#)

- [ListLexicon](#)
- [PutLexicon](#)
- [StartSpeechSynthesisTask](#)
- [SynthesizeSpeech](#)

## DeleteLexicon

以下 Python 代码示例使用 AWS SDK for Python (Boto) 删除在本地 AWS 配置中指定的区域中的词典。该示例只删除指定的词典。在实际删除词典前，它会要求您确认要继续操作。

以下代码示例使用存储在 AWS 软件开发工具包配置文件中的默认凭证。有关创建配置文件的信息，请参阅 [设置 AWS CLI](#)。

有关此操作的更多信息，请参阅 [DeleteLexicon API 参考](#)。

```
from argparse import ArgumentParser
from sys import version_info

from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError

# Define and parse the command line arguments
cli = ArgumentParser(description="DeleteLexicon example")
cli.add_argument("name", type=str, metavar="LEXICON_NAME")
arguments = cli.parse_args()

# Create a client using the credentials and region defined in the adminuser
# section of the AWS credentials and configuration files
session = Session(profile_name="adminuser")
polly = session.client("polly")

# Request confirmation
prompt = input if version_info >= (3, 0) else raw_input
proceed = prompt((u"This will delete the \"{0}\" lexicon,"
                 " do you want to proceed? [y,n]: ").format(arguments.name))

if proceed in ("y", "Y"):
    print(u"Deleting {0}...".format(arguments.name))

    try:
```

```
# Request deletion of a lexicon by name
response = polly.delete_lexicon(Name=arguments.name)
except (BotoCoreError, ClientError) as error:
    # The service returned an error, exit gracefully
    cli.error(error)

print("Done.")
else:
    print("Cancelled.")
```

## GetLexicon

以下 Python 代码使用 AWS SDK for Python (Boto) 来检索一个 AWS 区域中存储的所有词典。该示例接受词典名称作为命令行参数，并仅获取该词典，打印出词典在本地保存的 tmp 路径。

以下代码示例使用存储在 AWS 软件开发工具包配置文件中的默认凭证。有关创建配置文件的更多信息，请参阅 [设置 AWS CLI](#)。

有关此操作的更多信息，请参阅 [GetLexicon API 参考](#)。

```
from argparse import ArgumentParser
from os import path
from tempfile import gettempdir

from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError

# Define and parse the command line arguments
cli = ArgumentParser(description="GetLexicon example")
cli.add_argument("name", type=str, metavar="LEXICON_NAME")
arguments = cli.parse_args()

# Create a client using the credentials and region defined in the adminuser
# section of the AWS credentials and configuration files
session = Session(profile_name="adminuser")
polly = session.client("polly")

print(u"Fetching {0}...".format(arguments.name))

try:
    # Fetch lexicon by name
    response = polly.get_lexicon(Name=arguments.name)
except (BotoCoreError, ClientError) as error:
```

```
# The service returned an error, exit gracefully
cli.error(error)

# Get the lexicon data from the response
lexicon = response.get("Lexicon", {})

# Access the lexicon's content
if "Content" in lexicon:
    output = path.join(gettempdir(), u"%s.pls" % arguments.name)
    print(u"Saving to %s..." % output)

    try:
        # Save the lexicon contents to a local file
        with open(output, "w") as pls_file:
            pls_file.write(lexicon["Content"])
    except IOError as error:
        # Could not write to file, exit gracefully
        cli.error(error)
else:
    # The response didn't contain lexicon data, exit gracefully
    cli.error("Could not fetch lexicons contents")

print("Done.")
```

## ListLexicon

以下 Python 代码示例使用 AWS SDK for Python (Boto) 在本地 AWS 配置中指定的区域中列出您的账户中的词典。有关创建配置文件的信息，请参阅 [设置 AWS CLI](#)。

有关此操作的更多信息，请参阅 [ListLexicons](#) API 参考。

```
import sys

from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError

# Create a client using the credentials and region defined in the adminuser
# section of the AWS credentials and configuration files
session = Session(profile_name="adminuser")
polly = session.client("polly")

try:
    # Request the list of available lexicons
```

```
response = polly.list_lexicons()
except (BotoCoreError, ClientError) as error:
    # The service returned an error, exit gracefully
    print(error)
    sys.exit(-1)

# Get the list of lexicons in the response
lexicons = response.get("Lexicons", [])
print("{0} lexicon(s) found".format(len(lexicons)))

# Output a formatted list of lexicons with some of the attributes
for lexicon in lexicons:
    print((u" - {Name} ({Attributes[LanguageCode]}), "
          "{Attributes[LexemesCount]} lexeme(s)").format(**lexicon))
```

## PutLexicon

以下代码示例显示如何使用基于 Python (boto3) 的应用程序将发音词典存储在 AWS 区域中。

有关此操作的更多信息，请参阅 [PutLexicon API 参考](#)。

请注意以下几点：

- 您需要通过提供本地词典文件名和存储词典名更新该代码。
- 该示例假定您拥有在名为 pls 的子目录中创建的词典文件。您需要视情况新路径。

以下代码示例使用存储在 AWS 软件开发工具包配置文件中的默认凭证。有关创建配置文件的的信息，请参阅 [设置 AWS CLI](#)。

有关此操作的更多信息，请参阅 [PutLexicon API 参考](#)。

```
from argparse import ArgumentParser

from boto3 import Session
from botocore.exceptions import BotoCoreError, ClientError

# Define and parse the command line arguments
cli = ArgumentParser(description="PutLexicon example")
cli.add_argument("path", type=str, metavar="FILE_PATH")
cli.add_argument("-n", "--name", type=str, required=True,
                 metavar="LEXICON_NAME", dest="name")
```

```
arguments = cli.parse_args()

# Create a client using the credentials and region defined in the adminuser
# section of the AWS credentials and configuration files
session = Session(profile_name="adminuser")
polly = session.client("polly")

# Open the PLS lexicon file for reading
try:
    with open(arguments.path, "r") as lexicon_file:
        # Read the pls file contents
        lexicon_data = lexicon_file.read()

        # Store the PLS lexicon on the service.
        # If a lexicon with that name already exists,
        # its contents will be updated
        response = polly.put_lexicon(Name=arguments.name,
                                     Content=lexicon_data)
except (IOError, BotoCoreError, ClientError) as error:
    # Could not open/read the file or the service returned an error,
    # exit gracefully
    cli.error(error)

print(u"The \"{0}\" lexicon is now available for use.".format(arguments.name))
```

## StartSpeechSynthesisTask

以下 Python 代码示例使用 AWS SDK for Python (Boto) 在本地 AWS 配置中指定的区域中列出您的账户中的词典。有关创建配置文件的信息，请参阅 [设置 AWS CLI](#)。

有关更多信息，请参阅 [StartSpeechSynthesisTask API 参考](#)。

```
import boto3
import time

polly_client = boto3.Session(
    aws_access_key_id='',
    aws_secret_access_key='',
    region_name='eu-west-2').client('polly')

response = polly_client.start_speech_synthesis_task(VoiceId='Joanna',
                                                    OutputS3BucketName='synth-books-buckets',
```

```
        OutputS3KeyPrefix='key',
        OutputFormat='mp3',
        Text='This is a sample text to be synthesized.',
        Engine='neural')

taskId = response['SynthesisTask']['TaskId']

print( "Task id is {} ".format(taskId))

task_status = polly_client.get_speech_synthesis_task(TaskId = taskId)

print(task_status)
```

## SynthesizeSpeech

以下 Python 代码示例使用 AWS SDK for Python (Boto) 将语音与较短的文本合成以进行近实时处理。有关更多信息，请参阅参考中的 [SynthesizeSpeech](#) 操作。

此示例使用一小串纯文本。您可以使用 SSML 文本来更好地控制输出。有关更多信息，请参阅 [由 SSML 文档生成语音](#)。

```
import boto3

polly_client = boto3.Session(
    aws_access_key_id=,
    aws_secret_access_key=,
    region_name='us-west-2').client('polly')

response = polly_client.synthesize_speech(VoiceId='Joanna',
    OutputFormat='mp3',
    Text = 'This is a sample text to be synthesized.',
    Engine = 'neural')

file = open('speech.mp3', 'wb')
file.write(response['AudioStream'].read())
file.close()
```

## Java 示例

本示例显示了如何使用 Amazon Polly 从基于 Java 的应用程序流式传输语音。该示例使用 [适用于 Java 的 AWS SDK](#)，借助从列表中选择的声音来读取指定文本。

显示的代码涵盖主要任务，但只能进行微小错误检查。如果 Amazon Polly 遇到错误，应用程序将终止。

要运行此示例应用程序，您需要以下条件：

- Java 8 Java 开发工具包 (JDK)
- [适用于 Java 的 AWS SDK](#)
- [– Apache Maven](#)

## 测试应用程序

1. 确保针对 JDK 设置了 JAVA\_HOME 环境变量。

例如，如果您在 Windows 上安装了 JDK 1.8.0\_121 ( 路径为 C:\Program Files\Java\jdk1.8.0\_121 ) ，则您应在命令提示符下键入以下内容：

```
set JAVA_HOME=""C:\Program Files\Java\jdk1.8.0_121""
```

如果您在 Linux 中安装了 JDK 1.8.0\_121 ( 路径为 /usr/lib/jvm/java8-openjdk-amd64 ) ，您可以在命令提示符下键入以下内容：

```
export JAVA_HOME=/usr/lib/jvm/java8-openjdk-amd64
```

2. 设置 Maven 环境变量以便从命令行中运行 Maven。

例如，如果您在 Windows 上安装了 Maven 3.3.9 ( 路径为 C:\Program Files\apache-maven-3.3.9 ) ，则您可以键入以下内容：

```
set M2_HOME=""C:\Program Files\apache-maven-3.3.9""
set M2=%M2_HOME%\bin
set PATH=%M2%;%PATH%
```

如果您在 Linux 上安装了 Maven 3.3.9 ( 路径为 /home/ec2-user/opt/apache-maven-3.3.9 ) ，则您可以键入以下内容：

```
export M2_HOME=/home/ec2-user/opt/apache-maven-3.3.9
export M2=$M2_HOME/bin
export PATH=$M2:$PATH
```

3. 创建名为 `polly-java-demo` 的新目录。
4. 在 `polly-java-demo` 目录中，创建一个名为 `pom.xml` 的新文件，并粘贴以下代码：

```
<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>
```

```
</plugin>
</plugins>
</build>
</project>
```

5. 在 polly 创建名为 src/main/java/com/amazonaws/demos 的新目录。
6. 在 polly 目录中，创建一个名为 PollyDemo.java 的新 Java 源文件，并粘贴以下代码：

```
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).withEngine("neural");
    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();  
  
}  
}
```

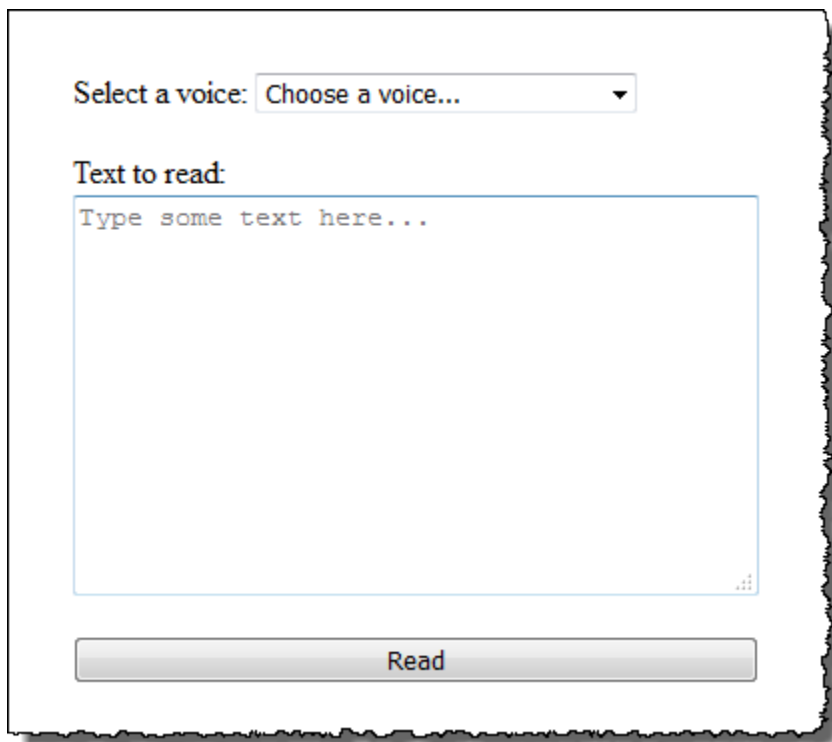
7. 返回到 `polly-java-demo` 目录，以清除、编译和执行演示：

```
mvn clean compile exec:java
```

## Python 示例 ( HTML5 客户端和 Python 服务器 )

此示例应用程序由以下内容组成：

- 使用 HTTP 分块传输编码的 HTTP 1.1 服务器 ( 请参阅 [分块传输编码](#) )
- 与 HTTP 1.1 服务器交互的简单 HTML5 用户界面 ( 如下所示 )：



此示例的目的是展示如何使用 Amazon Polly 从基于浏览器的 HTML5 应用程序流式传输语音。推荐将随着文本的合成而使用 Amazon Polly 制作的音频流的方法用于注重响应性的用例（例如对话系统、屏幕阅读器等）。

要运行此示例应用程序，您需要以下条件：

- 符合 HTML5 和 EcmaScript 5 标准的 Web 浏览器（例如 Chrome 23.0 或更高版本、Firefox 21.0 或更高版本、Internet Explorer 9.0 或更高版本）
- Python 版本在 3.0 以上

## 测试应用程序

1. 将服务器代码保存为 `server.py`。有关代码，请参阅 [Python 示例：Python 服务器代码 \(server.py\)](#)。
2. 将 HTML5 客户端代码另存为 `index.html`。有关代码，请参阅 [Python 示例：HTML5 用户界面 \(index.html\)](#)。
3. 从您保存 `server.py` 的路径运行以下命令以启动应用程序（在某些系统中，在运行命令时，您需要使用 `python3` 而不是 `python`）。

```
$ python server.py
```

在应用程序启动后，终端会显示 URL。

4. 在 Web 浏览器中打开在终端中显示的 URL。

您可以通过向应用程序服务器传输地址和端口以用作 `server.py` 的参数。有关更多信息，请运行 `python server.py -h`。

5. 要收听语音，请从列表中选择一个语音，键入一些文本，然后选择 Read。在 Amazon Polly 将第一个可用音频数据区块转化完之后，可立即开始播放语音。
6. 要在测试完应用程序后停止 Python 服务器，请在运行服务器的终端上按 `Ctrl+C`。

### Note

服务器使用 AWS SDK for Python (Boto) 创建了 Boto3 客户端。客户端使用存储在您计算机 AWS 配置文件中的凭证对向 Amazon Polly 发出的请求进行签名和身份验证。有关如何创建

AWS 配置文件和存储凭据的更多信息，请参阅AWS Command Line Interface 用户指南 [AWS Command Line Interface 中的配置](#)。

## Python 示例：HTML5 用户界面 (index.html)

本节提供中所述的 HTML5 客户端代码[Python 示例 \(HTML5 客户端和 Python 服务器\)](#)。

```
<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',
      'mulaw': 'audio/mulaw',
      'alaw': 'audio/alaw'
    };
```

```
/**
 * 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 示例：Python 服务器代码 ( server.py )

此部分提供了 [Python 示例 \( HTML5 客户端和 Python 服务器 \)](#) 中所述的 Python 服务器的代码。

```
"""
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",
                 "mulaw": "audio/mulaw",
                 "alaw": "audio/alaw"}

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,
                                               Engine="neural")

        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()
```

## iOS 示例

以下示例使用适用于 Amazon Polly 的 iOS 软件开发工具包，借助从语音列表中选择语音来读取指定文本。

此处显示的代码涵盖主要任务，但不处理错误。有关完整代码，请参阅[AWS Mobile SDK for iOS Amazon Polly 演示](#)。

### 初始化

```
// 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()
```

### 获取可用语音的列表

```
// 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
})
```

## 合成语音

```
// 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 示例

以下示例使用适用于 Amazon Polly 的 Android 软件开发工具包，借助从列表中选择的声音读取指定文本。

此处显示的代码涵盖主要任务，但不处理错误。有关完整代码，请参阅[适用于 Android 的 AWS Mobile SDK Amazon Polly 演示](#)。

### 初始化

```
// 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);
```

### 获取可用语音的列表

```
// 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();
```

## 获取音频流的 URL

```
// 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);
```

## 播放合成的语音

```
// 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();
        }
    });
});
```

# 使用 Amazon Polly 的代码示例 AWS 软件开发工具包

以下代码示例展示了如何将 Amazon Polly 与 AWS 软件开发套件 (SDK) 配合使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## 代码示例

- [使用 Amazon Polly 的基本示例 AWS 软件开发工具包](#)
  - [使用 Amazon Polly 执行的操作 AWS 软件开发工具包](#)
    - [DeleteLexicon搭配使用 AWS SDK 或 CLI](#)
    - [DescribeVoices搭配使用 AWS SDK](#)
    - [GetLexicon搭配使用 AWS SDK 或 CLI](#)
    - [GetSpeechSynthesisTask搭配使用 AWS SDK 或 CLI](#)
    - [ListLexicons搭配使用 AWS SDK 或 CLI](#)
    - [ListSpeechSynthesisTasks搭配使用 AWS SDK 或 CLI](#)
    - [PutLexicon搭配使用 AWS SDK 或 CLI](#)
    - [StartSpeechSynthesisTask搭配使用 AWS SDK 或 CLI](#)
    - [SynthesizeSpeech搭配使用 AWS SDK](#)
  - [Amazon Polly 使用的场景 AWS 软件开发工具包](#)
    - [使用将文本转换为语音并转换回文本 AWS SDK](#)
    - [使用 Amazon Polly 创建口型同步应用程序 AWS SDK](#)
    - [创建用于分析客户反馈和合成音频的应用程序](#)
    - [Amazon Polly 入门](#)

## 使用 Amazon Polly 的基本示例 AWS 软件开发工具包

以下代码示例演示了如何将 Amazon Polly 的基本功能与 AWS SDK 结合使用。

## 示例

- [使用 Amazon Polly 执行的操作 AWS 软件开发工具包](#)
  - [DeleteLexicon 搭配使用 AWS SDK 或 CLI](#)
  - [DescribeVoices 搭配使用 AWS SDK](#)
  - [GetLexicon 搭配使用 AWS SDK 或 CLI](#)
  - [GetSpeechSynthesisTask 搭配使用 AWS SDK 或 CLI](#)
  - [ListLexicons 搭配使用 AWS SDK 或 CLI](#)
  - [ListSpeechSynthesisTasks 搭配使用 AWS SDK 或 CLI](#)
  - [PutLexicon 搭配使用 AWS SDK 或 CLI](#)
  - [StartSpeechSynthesisTask 搭配使用 AWS SDK 或 CLI](#)
  - [SynthesizeSpeech 搭配使用 AWS SDK](#)

## 使用 Amazon Polly 执行的操作 AWS 软件开发工具包

以下代码示例演示了如何使用软件开发工具包执行单个 Amazon Polly 操作。AWS 每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Amazon Polly API，是必须在上下文中运行的较大型程序的代码节选。您可以在 [Amazon Polly 使用的场景 AWS 软件开发工具包](#) 中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 [Amazon Polly API 参考](#)。

## 示例

- [DeleteLexicon 搭配使用 AWS SDK 或 CLI](#)
- [DescribeVoices 搭配使用 AWS SDK](#)
- [GetLexicon 搭配使用 AWS SDK 或 CLI](#)
- [GetSpeechSynthesisTask 搭配使用 AWS SDK 或 CLI](#)
- [ListLexicons 搭配使用 AWS SDK 或 CLI](#)
- [ListSpeechSynthesisTasks 搭配使用 AWS SDK 或 CLI](#)
- [PutLexicon 搭配使用 AWS SDK 或 CLI](#)
- [StartSpeechSynthesisTask 搭配使用 AWS SDK 或 CLI](#)
- [SynthesizeSpeech 搭配使用 AWS SDK](#)

## DeleteLexicon 搭配使用 AWS SDK 或 CLI

以下代码示例演示如何使用 DeleteLexicon。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [Amazon Polly 入门](#)

### .NET

适用于 .NET 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Deletes an existing Amazon Polly lexicon using the AWS SDK for .NET.
/// </summary>
public class DeleteLexicon
{
    public static async Task Main()
    {
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();

        var success = await DeletePollyLexiconAsync(client, lexiconName);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {lexiconName}.");
        }
    }
}
```

```
        else
        {
            Console.WriteLine($"Could not delete {lexiconName}.");
        }
    }

    /// <summary>
    /// Deletes the named Amazon Polly lexicon.
    /// </summary>
    /// <param name="client">The initialized Amazon Polly client object.</
param>
    /// <param name="lexiconName">The name of the Amazon Polly lexicon to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public static async Task<bool> DeletePollyLexiconAsync(
        AmazonPollyClient client,
        string lexiconName)
    {
        var deleteLexiconRequest = new DeleteLexiconRequest()
        {
            Name = lexiconName,
        };

        var response = await client.DeleteLexiconAsync(deleteLexiconRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考 [DeleteLexicon](#) 中的。

## CLI

### AWS CLI

#### 删除词典

以下 delete-lexicon 示例删除指定的词典。

```
aws polly delete-lexicon \
```

```
--name w3c
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Polly 开发者指南中的[使用 DeleteLexicon 操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[DeleteLexicon](#)中的。

## SAP ABAP

### 适用于 SAP ABAP 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
TRY.  
    lo_ply->delelexicon( iv_name ).  
    MESSAGE 'Lexicon deleted successfully.' TYPE 'I'.  
CATCH /aws1/cx_plylexiconnotfoundex.  
    MESSAGE 'Lexicon not found.' TYPE 'E'.  
CATCH /aws1/cx_plyservicefailureex.  
    MESSAGE 'Service failure occurred.' TYPE 'E'.  
ENDTRY.
```

- 有关 API 的详细信息，请参阅适用[DeleteLexicon](#)于 SAP 的 AWS SDK ABAP API 参考。

有关 SAP AWS SDK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## DescribeVoices 搭配使用 AWS SDK

以下代码示例演示如何使用 DescribeVoices。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [Amazon Polly 入门](#)

## .NET

### 适用于 .NET 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class DescribeVoices
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();

        var allVoicesRequest = new DescribeVoicesRequest();
        var enUsVoicesRequest = new DescribeVoicesRequest()
        {
            LanguageCode = "en-US",
        };

        try
        {
            string nextToken;
            do
            {
                var allVoicesResponse = await
client.DescribeVoicesAsync(allVoicesRequest);
                nextToken = allVoicesResponse.NextToken;
                allVoicesRequest.NextToken = nextToken;

                Console.WriteLine("\nAll voices: ");
                allVoicesResponse.Voices.ForEach(voice =>
                {
                    DisplayVoiceInfo(voice);
                });
            }
        }
    }
}
```

```
    }
    while (nextToken is not null);

    do
    {
        var enUsVoicesResponse = await
client.DescribeVoicesAsync(enUsVoicesRequest);
        nextToken = enUsVoicesResponse.NextToken;
        enUsVoicesRequest.NextToken = nextToken;

        Console.WriteLine("\nen-US voices: ");
        enUsVoicesResponse.Voices.ForEach(voice =>
        {
            DisplayVoiceInfo(voice);
        });
    }
    while (nextToken is not null);
}
catch (Exception ex)
{
    Console.WriteLine("Exception caught: " + ex.Message);
}
}

public static void DisplayVoiceInfo(Voice voice)
{
    Console.WriteLine($" Name: {voice.Name}\tGender:
{voice.Gender}\tLanguageName: {voice.LanguageName}");
}
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考 [DescribeVoices](#) 中的。

## Java

### 适用于 Java 的 SDK 2.x

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        describeVoice(polly);
        polly.close();
    }

    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                .languageCode("en-US")
                .build();
```

```
        DescribeVoicesResponse enUsVoicesResult =
polly.describeVoices(voicesRequest);
        List<Voice> voices = enUsVoicesResult.voices();
        for (Voice myVoice : voices) {
            System.out.println("The ID of the voice is " + myVoice.id());
            System.out.println("The gender of the voice is " +
myVoice.gender());
        }

    } catch (PollyException e) {
        System.err.println("Exception caught: " + e);
        System.exit(1);
    }
}
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeVoices](#)中的。

## Python

### 适用于 Python 的 SDK ( Boto3 )

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None
```

```
def describe_voices(self):
    """
    Gets metadata about available voices.

    :return: The list of voice metadata.
    """
    try:
        response = self.polly_client.describe_voices()
        self.voice_metadata = response["Voices"]
        logger.info("Got metadata about %s voices.",
len(self.voice_metadata))
    except ClientError:
        logger.exception("Couldn't get voice metadata.")
        raise
    else:
        return self.voice_metadata
```

- 有关 API 的详细信息，请参阅适用[DescribeVoices](#)于 Python 的 AWS SDK (Boto3) API 参考。

## Ruby

### 适用于 Ruby 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
```

```
# and the configuration (region) from the shared configuration file ~/.aws/
config
polly = Aws::Polly::Client.new

# Get US English voices
resp = polly.describe_voices(language_code: 'en-US')

resp.voices.each do |v|
  puts v.name
  puts "  #{v.gender}"
  puts
end
rescue StandardError => e
  puts 'Could not get voices'
  puts 'Error message:'
  puts e.message
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [DescribeVoices](#) 中的。

## Rust

### 适用于 Rust 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
async fn list_voices(client: &Client) -> Result<(), Error> {
  let resp = client.describe_voices().send().await?;

  println!("Voices:");

  let voices = resp.voices();
  for voice in voices {
    println!("  Name:      {}", voice.name().unwrap_or("No name!"));
    println!(
      "    Language: {}",
      voice.language_name().unwrap_or("No language!")
    );
  }
}
```

```
    );  
  
    println!();  
}  
  
println!("Found {} voices", voices.len());  
  
Ok(())  
}
```

- 有关 API 的详细信息，请参阅适用[DescribeVoices](#)于 Rust 的 AWS SDK API 参考。

## SAP ABAP

### 适用于 SAP ABAP 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
TRY.  
    " Only pass optional parameters if they have values  
    IF iv_engine IS NOT INITIAL AND iv_language IS NOT INITIAL.  
        oo_result = lo_ply->describevoices(  
            iv_engine = iv_engine  
            iv_languagecode = iv_language ).  
    ELSEIF iv_engine IS NOT INITIAL.  
        oo_result = lo_ply->describevoices(  
            iv_engine = iv_engine ).  
    ELSEIF iv_language IS NOT INITIAL.  
        oo_result = lo_ply->describevoices(  
            iv_languagecode = iv_language ).  
    ELSE.  
        oo_result = lo_ply->describevoices( ).  
    ENDIF.  
    MESSAGE 'Retrieved voice metadata.' TYPE 'I'.  
CATCH /aws1/cx_plyinvalidnexttokenex.  
    MESSAGE 'The NextToken is invalid.' TYPE 'E'.  
CATCH /aws1/cx_plyservicefailureex.
```

```
MESSAGE 'Service failure occurred.' TYPE 'E'.  
ENDTRY.
```

- 有关 API 的详细信息，请参阅适用[DescribeVoices](#)于 S AP 的 AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## GetLexicon 搭配使用 AWS SDK 或 CLI

以下代码示例演示如何使用 GetLexicon。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [Amazon Polly 入门](#)

### .NET

适用于 .NET 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
using System;  
using System.Threading.Tasks;  
using Amazon.Polly;  
using Amazon.Polly.Model;  
  
/// <summary>  
/// Retrieves information about a specific Amazon Polly lexicon.  
/// </summary>  
public class GetLexicon  
{  
    public static async Task Main(string[] args)  
    {
```

```
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();

        await GetPollyLexiconAsync(client, lexiconName);
    }

    public static async Task GetPollyLexiconAsync(AmazonPollyClient client,
string lexiconName)
    {
        var getLexiconRequest = new GetLexiconRequest()
        {
            Name = lexiconName,
        };

        try
        {
            var response = await client.GetLexiconAsync(getLexiconRequest);
            Console.WriteLine($"Lexicon:\n Name: {response.Lexicon.Name}");
            Console.WriteLine($"Content: {response.Lexicon.Content}");
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error: " + ex.Message);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考[GetLexicon](#)中的。

## CLI

### AWS CLI

#### 检索词典的内容

以下 `get-lexicon` 示例检索指定的发音词典内容。

```
aws polly get-lexicon \  
  --name w3c
```

输出：

```
{
  "Lexicon": {
    "Content": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<lexicon version=
\"1.0\" \n      xmlns=      \"http://www.w3.org/2005/01/pronunciation-lexicon
\" \n      xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" \n
xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon \n
http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\" \n
alphabet=\"ipa\" \n      xml:lang=\"en-US\">\n  <lexeme>\n    <grapheme>W3C</
grapheme>\n      <alias>World Wide Web Consortium</alias>\n  </lexeme>\n</
lexicon>\n",
    "Name": "w3c"
  },
  "LexiconAttributes": {
    "Alphabet": "ipa",
    "LanguageCode": "en-US",
    "LastModified": 1603908910.99,
    "LexiconArn": "arn:aws:polly:us-west-2:880185128111:lexicon/w3c",
    "LexemesCount": 1,
    "Size": 492
  }
}
```

有关更多信息，请参阅 Amazon Polly 开发者指南中的[使用 GetLexicon 操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[GetLexicon](#)中的。

## Python

适用于 Python 的 SDK ( Boto3 )

### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
```

```
    """
    :param polly_client: A Boto3 Amazon Polly client.
    :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
resource.
    """
    self.polly_client = polly_client
    self.s3_resource = s3_resource
    self.voice_metadata = None

def get_lexicon(self, name):
    """
    Gets metadata and contents of an existing lexicon.

    :param name: The name of the lexicon to retrieve.
    :return: The retrieved lexicon.
    """
    try:
        response = self.polly_client.get_lexicon(Name=name)
        logger.info("Got lexicon %s.", name)
    except ClientError:
        logger.exception("Couldn't get lexicon %s.", name)
        raise
    else:
        return response
```

- 有关 API 的详细信息，请参阅适用[GetLexicon](#)于 Python 的AWS SDK (Boto3) API 参考。

## SAP ABAP

### 适用于 SAP ABAP 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

TRY.

```
oo_result = lo_ply->getlexicon( iv_name ).
```

```

DATA(lo_lexicon) = oo_result->get_lexicon( ).
IF lo_lexicon IS BOUND.
  DATA(lv_lex_name) = lo_lexicon->get_name( ).
  MESSAGE |Retrieved lexicon: { lv_lex_name }| TYPE 'I'.
ENDIF.
CATCH /aws1/cx_plylexiconnotfoundex.
  MESSAGE 'Lexicon not found.' TYPE 'E'.
CATCH /aws1/cx_plyservicefailureex.
  MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.

```

- 有关 API 的详细信息，请参阅适用[GetLexicon](#)于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## GetSpeechSynthesisTask 搭配使用 AWS SDK 或 CLI

以下代码示例演示如何使用 GetSpeechSynthesisTask。

### CLI

#### AWS CLI

获取有关语音合成任务的信息

以下 `get-speech-synthesis-task` 示例检索有关指定语音合成任务的信息。

```

aws polly get-speech-synthesis-task \
  --task-id 70b61c0f-57ce-4715-a247-cae8729dcce9

```

输出：

```

{
  "SynthesisTask": {
    "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",
    "TaskStatus": "completed",
    "OutputUri": "https://s3.us-west-2.amazonaws.com/amzn-s3-demo-
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",
    "CreationTime": 1603911042.689,
    "RequestCharacters": 1311,

```

```
        "OutputFormat": "mp3",
        "TextType": "text",
        "VoiceId": "Joanna"
    }
}
```

有关更多信息，请参阅《Amazon Polly 开发人员指南》中的[创建长音频文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetSpeechSynthesisTask](#)中的。

## Python

适用于 Python 的 SDK ( Boto3 )

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
        resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def get_speech_synthesis_task(self, task_id):
        """
        Gets metadata about an asynchronous speech synthesis task, such as its
        status.

        :param task_id: The ID of the task to retrieve.
        :return: Metadata about the task.
        """
```

```
try:
    response =
self.polly_client.get_speech_synthesis_task(TaskId=task_id)
    task = response["SynthesisTask"]
    logger.info("Got synthesis task. Status is %s.", task["TaskStatus"])
except ClientError:
    logger.exception("Couldn't get synthesis task %s.", task_id)
    raise
else:
    return task
```

- 有关 API 的详细信息，请参阅适用[GetSpeechSynthesisTask](#)于 Python 的AWS SDK (Boto3) API 参考。

## SAP ABAP

### 适用于 SAP ABAP 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_ply->getspeechsynthesistask( iv_task_id ).
    DATA(lo_task) = oo_result->get_synthesistask( ).
    IF lo_task IS BOUND.
        DATA(lv_status) = lo_task->get_taskstatus( ).
        MESSAGE |Task status: { lv_status }| TYPE 'I'.
    ENDIF.
CATCH /aws1/cx_plyinvalidtaskidex.
    MESSAGE 'Invalid task ID.' TYPE 'E'.
CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
CATCH /aws1/cx_plysynthesistsknotf00.
    MESSAGE 'Synthesis task not found.' TYPE 'E'.
ENDTRY.
```

- 有关 API 的详细信息，请参阅适用[GetSpeechSynthesisTask](#)于 S AP 的 AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## ListLexicons 搭配使用 AWS SDK 或 CLI

以下代码示例演示如何使用 ListLexicons。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [Amazon Polly 入门](#)

### .NET

适用于 .NET 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Lists the Amazon Polly lexicons that have been defined. By default,
/// lists the lexicons that are defined in the same AWS Region as the default
/// user. To view Amazon Polly lexicons that are defined in a different AWS
/// Region, supply it as a parameter to the Amazon Polly constructor.
/// </summary>
public class ListLexicons
{
    public static async Task Main()
    {
```

```
var client = new AmazonPollyClient();
var request = new ListLexiconsRequest();

try
{
    Console.WriteLine("All voices: ");

    do
    {
        var response = await client.ListLexiconsAsync(request);
        request.NextToken = response.NextToken;

        response.Lexicons.ForEach(lexicon =>
        {
            var attributes = lexicon.Attributes;
            Console.WriteLine($"Name: {lexicon.Name}");
            Console.WriteLine($"\\tAlphabet: {attributes.Alphabet}");
            Console.WriteLine($"\\tLanguageCode:
{attributes.LanguageCode}");
            Console.WriteLine($"\\tLastModified:
{attributes.LastModified}");
            Console.WriteLine($"\\tLexemesCount:
{attributes.LexemesCount}");
            Console.WriteLine($"\\tLexiconArn:
{attributes.LexiconArn}");
            Console.WriteLine($"\\tSize: {attributes.Size}");
        });
    }
    while (request.NextToken is not null);
}
catch (Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考 [ListLexicons](#) 中的。

## CLI

### AWS CLI

列出您的词典

以下 `list-lexicons` 示例列出了您的发音词典。

```
aws polly list-lexicons
```

输出：

```
{
  "Lexicons": [
    {
      "Name": "w3c",
      "Attributes": {
        "Alphabet": "ipa",
        "LanguageCode": "en-US",
        "LastModified": 1603908910.99,
        "LexiconArn": "arn:aws:polly:us-east-2:123456789012:lexicon/w3c",
        "LexemesCount": 1,
        "Size": 492
      }
    }
  ]
}
```

有关更多信息，请参阅 Amazon Polly 开发者指南中的[使用 ListLexicons 操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[ListLexicons](#)中的。

## Java

适用于 Java 的 SDK 2.x

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
            .region(Region.US_WEST_2)
            .build();

        listLexicons(polly);
        polly.close();
    }

    public static void listLexicons(PollyClient client) {
        try {
            ListLexiconsRequest listLexiconsRequest =
                ListLexiconsRequest.builder()
                    .build();

            ListLexiconsResponse listLexiconsResult =
                client.listLexicons(listLexiconsRequest);
            List<LexiconDescription> lexiconDescription =
                listLexiconsResult.lexicons();
            for (LexiconDescription lexDescription : lexiconDescription) {
                System.out.println("The name of the Lexicon is " +
                    lexDescription.name());
            }
        } catch (PollyException e) {
```

```
        System.err.println("Exception caught: " + e);
        System.exit(1);
    }
}
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考 [ListLexicons](#) 中的。

## Python

### 适用于 Python 的 SDK ( Boto3 )

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
        resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def list_lexicons(self):
        """
        Lists lexicons in the current account.

        :return: The list of lexicons.
        """
        try:
            response = self.polly_client.list_lexicons()
            lexicons = response["Lexicons"]
```

```
        logger.info("Got %s lexicons.", len.lexicons))
    except ClientError:
        logger.exception(
            "Couldn't get %s.",
        )
        raise
    else:
        return lexicons
```

- 有关 API 的详细信息，请参阅适用[ListLexicons](#)于 Python 的 AWS SDK (Boto3) API 参考。

## Ruby

### 适用于 Ruby 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/
  config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:#{l.attributes.alphabet}"
    puts "  Language:#{l.attributes.language}"
    puts
  end
rescue StandardError => e
```

```
puts 'Could not get lexicons'  
puts 'Error message:'  
puts e.message  
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [ListLexicons](#) 中的。

## Rust

### 适用于 Rust 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
async fn show_lexicons(client: &Client) -> Result<(), Error> {  
    let resp = client.list_lexicons().send().await?;  
  
    println!("Lexicons:");  
  
    let lexicons = resp.lexicons();  
  
    for lexicon in lexicons {  
        println!("  Name:      {}", lexicon.name().unwrap_or_default());  
        println!(  
            "  Language: {:?}\n",  
            lexicon  
                .attributes()  
                .as_ref()  
                .map(|attrib| attrib  
                    .language_code  
                    .as_ref()  
                    .expect("languages must have language codes"))  
                .expect("languages must have attributes")  
        );  
    }  
  
    println();  
    println!("Found {} lexicons.", lexicons.len());  
}
```

```
println!();  
  
Ok(())  
}
```

- 有关 API 的详细信息，请参阅适用[ListLexicons](#)于 Rust 的 AWS SDK API 参考。

## SAP ABAP

### 适用于 SAP ABAP 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
TRY.  
    oo_result = lo_ply->listlexicons( ).  
    DATA(lt_lexicons) = oo_result->get_lexicons( ).  
    DATA(lv_count) = lines( lt_lexicons ).  
    MESSAGE |Found { lv_count } lexicons| TYPE 'I'.  
CATCH /aws1/cx_plyinvalidnexttokenex.  
    MESSAGE 'Invalid NextToken.' TYPE 'E'.  
CATCH /aws1/cx_plyservicefailureex.  
    MESSAGE 'Service failure occurred.' TYPE 'E'.  
ENDTRY.
```

- 有关 API 的详细信息，请参阅适用[ListLexicons](#)于 SAP 的 AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## ListSpeechSynthesisTasks 搭配使用 AWS SDK 或 CLI

以下代码示例演示如何使用 ListSpeechSynthesisTasks。

## CLI

## AWS CLI

列出您的语音合成任务

以下 `list-speech-synthesis-tasks` 示例列出了您的语音合成任务。

```
aws polly list-speech-synthesis-tasks
```

输出：

```
{
  "SynthesisTasks": [
    {
      "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",
      "TaskStatus": "completed",
      "OutputUri": "https://s3.us-west-2.amazonaws.com/amzn-s3-demo-
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",
      "CreationTime": 1603911042.689,
      "RequestCharacters": 1311,
      "OutputFormat": "mp3",
      "TextType": "text",
      "VoiceId": "Joanna"
    }
  ]
}
```

有关更多信息，请参阅《Amazon Polly 开发人员指南》中的[创建长音频文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ListSpeechSynthesisTasks](#)中的。

## SAP ABAP

适用于 SAP ABAP 的 SDK

**Note**

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

TRY.
  " Only pass optional parameters if they have values
  IF iv_max_results IS NOT INITIAL AND iv_status IS NOT INITIAL.
    oo_result = lo_ply->listspeechsynthesistasks(
      iv_maxresults = iv_max_results
      iv_status = iv_status ).
  ELSEIF iv_max_results IS NOT INITIAL.
    oo_result = lo_ply->listspeechsynthesistasks(
      iv_maxresults = iv_max_results ).
  ELSEIF iv_status IS NOT INITIAL.
    oo_result = lo_ply->listspeechsynthesistasks(
      iv_status = iv_status ).
  ELSE.
    oo_result = lo_ply->listspeechsynthesistasks( ).
  ENDIF.
  DATA(lt_tasks) = oo_result->get_synthesistasks( ).
  DATA(lv_count) = lines( lt_tasks ).
  MESSAGE |Found { lv_count } synthesis tasks| TYPE 'I'.
CATCH /aws1/cx_plyinvalidnexttokenex.
  MESSAGE 'Invalid NextToken.' TYPE 'E'.
CATCH /aws1/cx_plyservicefailureex.
  MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.

```

- 有关 API 的详细信息，请参阅适用[ListSpeechSynthesisTasks](#)于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## PutLexicon搭配使用 AWS SDK 或 CLI

以下代码示例演示如何使用 PutLexicon。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [Amazon Polly 入门](#)

## .NET

### 适用于 .NET 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Creates a new Amazon Polly lexicon using the AWS SDK for .NET.
/// </summary>
public class PutLexicon
{
    public static async Task Main()
    {
        string lexiconContent = "<?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>";
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();
        var putLexiconRequest = new PutLexiconRequest()
        {
            Name = lexiconName,
            Content = lexiconContent,
        };
    }
}
```

```
        try
        {
            var response = await client.PutLexiconAsync(putLexiconRequest);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully created Lexicon:
{lexiconName}.");
            }
            else
            {
                Console.WriteLine($"Could not create Lexicon:
{lexiconName}.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Exception caught: " + ex.Message);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考 [PutLexicon](#) 中的。

## CLI

### AWS CLI

#### 存储词典

以下 `put-lexicon` 示例存储指定的发音词典。该 `example.pls` 文件指定了 W3C PLS-compliant 词典。

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

#### example.pls 的内容

```
{  
  <?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>
}
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon Polly 开发者指南中的[使用 PutLexicon 操作](#)。

- 有关 API 的详细信息，请参阅 AWS CLI 命令参考[PutLexicon](#)中的。

## Python

适用于 Python 的 SDK ( Boto3 )

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
        resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None
```

```

def create_lexicon(self, name, content):
    """
    Creates a lexicon with the specified content. A lexicon contains custom
    pronunciations.

    :param name: The name of the lexicon.
    :param content: The content of the lexicon.
    """
    try:
        self.polly_client.put_lexicon(Name=name, Content=content)
        logger.info("Created lexicon %s.", name)
    except ClientError:
        logger.exception("Couldn't create lexicon %s.")
        raise

```

- 有关 API 的详细信息，请参阅适用[PutLexicon](#)于 Python 的 AWS SDK (Boto3) API 参考。

## Rust

### 适用于 Rust 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

async fn make_lexicon(client: &Client, name: &str, from: &str, to: &str) ->
    Result<(), Error> {
    let content = format!("<?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>{}</grapheme><alias>{}</alias></lexeme>
    </lexicon>", from, to);

```

```

client
    .put_lexicon()
    .name(name)
    .content(content)
    .send()
    .await?;

println!("Added lexicon");

Ok(())
}

```

- 有关 API 的详细信息，请参阅适用[PutLexicon](#)于 Rust 的AWS SDK API 参考。

## SAP ABAP

### 适用于 SAP ABAP 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```

TRY.
  lo_ply->putlexicon(
    iv_name = iv_name
    iv_content = iv_content ).
  MESSAGE 'Lexicon created successfully.' TYPE 'I'.
CATCH /aws1/cx_plyinvalidlexiconex.
  MESSAGE 'Invalid lexicon.' TYPE 'E'.
CATCH /aws1/cx_plylexiconsizeexcdex.
  MESSAGE 'Lexicon size exceeded.' TYPE 'E'.
CATCH /aws1/cx_plymaxlexemelengthe00.
  MESSAGE 'Maximum lexeme length exceeded.' TYPE 'E'.
CATCH /aws1/cx_plymaxlexiconsnoexc00.
  MESSAGE 'Maximum number of lexicons exceeded.' TYPE 'E'.
CATCH /aws1/cx_plyservicefailureex.
  MESSAGE 'Service failure occurred.' TYPE 'E'.
CATCH /aws1/cx_plyunsuppedplsalpha00.
  MESSAGE 'Unsupported PLS alphabet.' TYPE 'E'.

```

```
CATCH /aws1/cx_plyunsuppedpls1angu00.  
MESSAGE 'Unsupported PLS language.' TYPE 'E'.  
ENDTRY.
```

- 有关 API 的详细信息，请参阅适用[PutLexicon](#)于 S AP 的 AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## StartSpeechSynthesisTask 搭配使用 AWS SDK 或 CLI

以下代码示例演示如何使用 StartSpeechSynthesisTask。

### CLI

#### AWS CLI

#### 合成文本

以下 start-speech-synthesis-task 示例合成了 text\_file.txt 中的文本，并将生成的 MP3 文件存储在指定的存储桶中。

```
aws polly start-speech-synthesis-task \  
  --output-format mp3 \  
  --output-s3-bucket-name amzn-s3-demo-bucket \  
  --text file://text_file.txt \  
  --voice-id Joanna
```

输出：

```
{  
  "SynthesisTask": {  
    "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",  
    "TaskStatus": "scheduled",  
    "OutputUri": "https://s3.us-east-2.amazonaws.com/amzn-s3-demo-  
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",  
    "CreationTime": 1603911042.689,  
    "RequestCharacters": 1311,  
    "OutputFormat": "mp3",  
    "TextType": "text",  
    "VoiceId": "Joanna"  }  
}
```

```
}  
}
```

有关更多信息，请参阅《Amazon Polly 开发人员指南》中的[创建长音频文件](#)。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StartSpeechSynthesisTask](#)中的。

## Python

适用于 Python 的 SDK ( Boto3 )

### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class PollyWrapper:  
    """Encapsulates Amazon Polly functions."""  
  
    def __init__(self, polly_client, s3_resource):  
        """  
        :param polly_client: A Boto3 Amazon Polly client.  
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)  
resource.  
        """  
        self.polly_client = polly_client  
        self.s3_resource = s3_resource  
        self.voice_metadata = None  
  
    def do_synthesis_task(  
        self,  
        text,  
        engine,  
        voice,  
        audio_format,  
        s3_bucket,  
        lang_code=None,  
        include_visemes=False,  
        wait_callback=None,  
    ):
```

```

"""
Start an asynchronous task to synthesize speech or speech marks, wait for
the task to complete, retrieve the output from Amazon S3, and return the
data.

An asynchronous task is required when the text is too long for near-real
time
synthesis.

:param text: The text to synthesize.
:param engine: The kind of engine used. Can be standard or neural.
:param voice: The ID of the voice to use.
:param audio_format: The audio format to return for synthesized speech.
When
                    speech marks are synthesized, the output format is
JSON.
:param s3_bucket: The name of an existing Amazon S3 bucket that you have
                    write access to. Synthesis output is written to this
bucket.
:param lang_code: The language code of the voice to use. This has an
effect
                    only when a bilingual voice is selected.
:param include_visemes: When True, a second request is made to Amazon
Polly
                    to synthesize a list of visemes, using the
specified
                    text and voice. A viseme represents the visual
position
                    of the face and mouth when saying part of a word.
:param wait_callback: A callback function that is called periodically
during
                    task processing, to give the caller an opportunity
to
                    take action, such as to display status.
:return: The audio stream that contains the synthesized speech and a list
of visemes that are associated with the speech audio.
"""
try:
    kwargs = {
        "Engine": engine,
        "OutputFormat": audio_format,
        "OutputS3BucketName": s3_bucket,
        "Text": text,
        "VoiceId": voice,

```

```
    }
    if lang_code is not None:
        kwargs["LanguageCode"] = lang_code
    response = self.polly_client.start_speech_synthesis_task(**kwargs)
    speech_task = response["SynthesisTask"]
    logger.info("Started speech synthesis task %s.",
speech_task["TaskId"])

    viseme_task = None
    if include_visemes:
        kwargs["OutputFormat"] = "json"
        kwargs["SpeechMarkTypes"] = ["viseme"]
        response =
self.polly_client.start_speech_synthesis_task(**kwargs)
        viseme_task = response["SynthesisTask"]
        logger.info("Started viseme synthesis task %s.",
viseme_task["TaskId"])
    except ClientError:
        logger.exception("Couldn't start synthesis task.")
        raise
    else:
        bucket = self.s3_resource.Bucket(s3_bucket)
        audio_stream = self._wait_for_task(
            10, speech_task["TaskId"], "speech", wait_callback, bucket
        )


        visemes = None
        if include_visemes:
            viseme_data = self._wait_for_task(
                10, viseme_task["TaskId"], "viseme", wait_callback, bucket
            )
            visemes = [
                json.loads(v) for v in viseme_data.read().decode().split() if
v
            ]

        return audio_stream, visemes
```

- 有关 API 的详细信息，请参阅适用[StartSpeechSynthesisTask](#)于 Python 的AWS SDK (Boto3) API 参考。

## SAP ABAP

## 适用于 SAP ABAP 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
TRY.  
    " Only pass optional parameters if they have values  
    IF iv_lang_code IS NOT INITIAL AND iv_s3_key_prefix IS NOT INITIAL.  
        oo_result = lo_ply->startspeechsynthesistask(  
            iv_engine = iv_engine  
            iv_outputformat = iv_audio_format  
            iv_outputs3bucketname = iv_s3_bucket  
            iv_outputs3keyprefix = iv_s3_key_prefix  
            iv_text = iv_text  
            iv_voiceid = iv_voice_id  
            iv_languagecode = iv_lang_code ).  
    ELSEIF iv_lang_code IS NOT INITIAL.  
        oo_result = lo_ply->startspeechsynthesistask(  
            iv_engine = iv_engine  
            iv_outputformat = iv_audio_format  
            iv_outputs3bucketname = iv_s3_bucket  
            iv_text = iv_text  
            iv_voiceid = iv_voice_id  
            iv_languagecode = iv_lang_code ).  
    ELSEIF iv_s3_key_prefix IS NOT INITIAL.  
        oo_result = lo_ply->startspeechsynthesistask(  
            iv_engine = iv_engine  
            iv_outputformat = iv_audio_format  
            iv_outputs3bucketname = iv_s3_bucket  
            iv_outputs3keyprefix = iv_s3_key_prefix  
            iv_text = iv_text  
            iv_voiceid = iv_voice_id ).  
    ELSE.  
        oo_result = lo_ply->startspeechsynthesistask(  
            iv_engine = iv_engine  
            iv_outputformat = iv_audio_format  
            iv_outputs3bucketname = iv_s3_bucket
```

```
        iv_text = iv_text
        iv_voiceid = iv_voice_id ).
    ENDIF.
    MESSAGE 'Speech synthesis task started.' TYPE 'I'.
CATCH /aws1/cx_plyinvalids3bucketex.
    MESSAGE 'Invalid S3 bucket.' TYPE 'E'.
CATCH /aws1/cx_plyinvalidssmlex.
    MESSAGE 'Invalid SSML.' TYPE 'E'.
CATCH /aws1/cx_plylexiconnotfoundex.
    MESSAGE 'Lexicon not found.' TYPE 'E'.
CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
CATCH /aws1/cx_plytextlengthexcdex.
    MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
ENDTRY.
```

- 有关 API 的详细信息，请参阅适用[StartSpeechSynthesisTask](#)于 S AP 的 AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## SynthesizeSpeech 搭配使用 AWS SDK

以下代码示例演示如何使用 SynthesizeSpeech。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [Amazon Polly 入门](#)

### .NET

适用于 .NET 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class SynthesizeSpeech
{
    public static async Task Main()
    {
        string outputFileName = "speech.mp3";
        string text = "Twas brillig, and the slithy toves did gyre and gimbol
in the wabe";

        var client = new AmazonPollyClient();
        var response = await PollySynthesizeSpeech(client, text);

        WriteSpeechToStream(response.AudioStream, outputFileName);
    }

    /// <summary>
    /// Calls the Amazon Polly SynthesizeSpeechAsync method to convert text
    /// to speech.
    /// </summary>
    /// <param name="client">The Amazon Polly client object used to connect
    /// to the Amazon Polly service.</param>
    /// <param name="text">The text to convert to speech.</param>
    /// <returns>A SynthesizeSpeechResponse object that includes an
AudioStream
    /// object with the converted text.</returns>
    private static async Task<SynthesizeSpeechResponse>
PollySynthesizeSpeech(IAmazonPolly client, string text)
    {
        var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
        {
            OutputFormat = OutputFormat.Mp3,
            VoiceId = VoiceId.Joanna,
            Text = text,
        };

        var synthesizeSpeechResponse =
            await client.SynthesizeSpeechAsync(synthesizeSpeechRequest);
    }
}
```

```
        return synthesizeSpeechResponse;
    }

    /// <summary>
    /// Writes the AudioStream returned from the call to
    /// SynthesizeSpeechAsync to a file in MP3 format.
    /// </summary>
    /// <param name="audioStream">The AudioStream returned from the
    /// call to the SynthesizeSpeechAsync method.</param>
    /// <param name="outputFileName">The full path to the file in which to
    /// save the audio stream.</param>
    private static void WriteSpeechToStream(Stream audioStream, string
outputFileName)
    {
        var outputStream = new FileStream(
            outputFileName,
            FileMode.Create,
            FileAccess.Write);
        byte[] buffer = new byte[2 * 1024];
        int readBytes;

        while ((readBytes = audioStream.Read(buffer, 0, 2 * 1024)) > 0)
        {
            outputStream.Write(buffer, 0, readBytes);
        }

        // Flushes the buffer to avoid losing the last second or so of
        // the synthesized text.
        outputStream.Flush();
        Console.WriteLine($"Saved {outputFileName} to disk.");
    }
}
```

使用软件开发工具包在 Amazon Polly 中使用语音标记合成文本中的语音。 AWS

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;
```

```
public class SynthesizeSpeechMarks
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();
        string outputFileName = "speechMarks.json";

        var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
        {
            OutputFormat = OutputFormat.Json,
            SpeechMarkTypes = new List<string>
            {
                SpeechMarkType.Viseme,
                SpeechMarkType.Word,
            },
            VoiceId = VoiceId.Joanna,
            Text = "This is a sample text to be synthesized.",
        };

        try
        {
            using (var outputStream = new FileStream(outputFileName,
                FileMode.Create, FileAccess.Write))
            {
                var synthesizeSpeechResponse = await
client.SynthesizeSpeechAsync(synthesizeSpeechRequest);
                var buffer = new byte[2 * 1024];
                int readBytes;

                var inputStream = synthesizeSpeechResponse.AudioStream;
                while ((readBytes = inputStream.Read(buffer, 0, 2 * 1024)) >
0)
                {
                    outputStream.Write(buffer, 0, readBytes);
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考 [SynthesizeSpeech](#) 中的。

## Java

### 适用于 Java 的 SDK 2.x

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class PollyDemo {
```

```
private static final String SAMPLE = "Congratulations. You have successfully
built this working demo " +
    " of Amazon Polly in Java Version 2. 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 static void main(String args[]) {
    PollyClient polly = PollyClient.builder()
        .region(Region.US_WEST_2)
        .build();

    talkPolly(polly);
    polly.close();
}

public static void talkPolly(PollyClient polly) {
    try {
        DescribeVoicesRequest describeVoiceRequest =
DescribeVoicesRequest.builder()
            .engine("standard")
            .build();

        DescribeVoicesResponse describeVoicesResult =
polly.describeVoices(describeVoiceRequest);
        Voice voice = describeVoicesResult.voices().stream()
            .filter(v -> v.name().equals("Joanna"))
            .findFirst()
            .orElseThrow(() -> new RuntimeException("Voice not found"));
        InputStream stream = synthesize(polly, SAMPLE, voice,
OutputFormat.MP3);
        AdvancedPlayer player = new AdvancedPlayer(stream,
javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
        player.setPlaybackListener(new PlaybackListener() {
            public void playbackStarted(PlaybackEvent evt) {
                System.out.println("Playback started");
                System.out.println(SAMPLE);
            }

            public void playbackFinished(PlaybackEvent evt) {
                System.out.println("Playback finished");
            }
        });
    }
}
```

```
    });

    // play it!
    player.play();

    } catch (PollyException | JavaLayerException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static InputStream synthesize(PollyClient polly, String text, Voice
voice, OutputFormat format)
    throws IOException {
    SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
        .text(text)
        .voiceId(voice.id())
        .outputFormat(format)
        .build();

    ResponseInputStream<SynthesizeSpeechResponse> synthRes =
polly.synthesizeSpeech(synthReq);
    return synthRes;
}
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[SynthesizeSpeech](#)中的。

## Python

### 适用于 Python 的 SDK ( Boto3 )

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
```

```

def __init__(self, polly_client, s3_resource):
    """
    :param polly_client: A Boto3 Amazon Polly client.
    :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
resource.
    """
    self.polly_client = polly_client
    self.s3_resource = s3_resource
    self.voice_metadata = None

def synthesize(
    self, text, engine, voice, audio_format, lang_code=None,
include_visemes=False
):
    """
    Synthesizes speech or speech marks from text, using the specified voice.

    :param text: The text to synthesize.
    :param engine: The kind of engine used. Can be standard or neural.
    :param voice: The ID of the voice to use.
    :param audio_format: The audio format to return for synthesized speech.
When
        speech marks are synthesized, the output format is
JSON.
    :param lang_code: The language code of the voice to use. This has an
effect
        only when a bilingual voice is selected.
    :param include_visemes: When True, a second request is made to Amazon
Polly
        to synthesize a list of visemes, using the
specified
        text and voice. A viseme represents the visual
position
        of the face and mouth when saying part of a word.
    :return: The audio stream that contains the synthesized speech and a list
of visemes that are associated with the speech audio.
    """
    try:
        kwargs = {
            "Engine": engine,
            "OutputFormat": audio_format,
            "Text": text,
            "VoiceId": voice,

```

```
    }
    if lang_code is not None:
        kwargs["LanguageCode"] = lang_code
    response = self.polly_client.synthesize_speech(**kwargs)
    audio_stream = response["AudioStream"]
    logger.info("Got audio stream spoken by %s.", voice)
    visemes = None
    if include_visemes:
        kwargs["OutputFormat"] = "json"
        kwargs["SpeechMarkTypes"] = ["viseme"]
        response = self.polly_client.synthesize_speech(**kwargs)
        visemes = [
            json.loads(v)
            for v in response["AudioStream"].read().decode().split()
            if v
        ]
        logger.info("Got %s visemes.", len(visemes))
except ClientError:
    logger.exception("Couldn't get audio stream.")
    raise
else:
    return audio_stream, visemes
```

- 有关 API 的详细信息，请参阅适用[SynthesizeSpeech](#)于 Python 的 AWS SDK (Boto3) API 参考。

## Ruby

### 适用于 Ruby 的 SDK

#### Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'
```

```
begin
  # Get the filename from the command line
  if ARGV.empty?
    puts 'You must supply a filename'
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: #{filename}"
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/
  config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
                                output_format: 'mp3',
                                text: contents,
                                voice_id: 'Joanna'
                              })

  # Save output
  # Get just the file name
  # abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = "#{first_part}.mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: #{mp3_file}"
rescue StandardError => e
  puts 'Got error:'
```

```
puts 'Error message:'
puts e.message
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考 [SynthesizeSpeech](#) 中的。

## Rust

### 适用于 Rust 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
async fn synthesize(client: &Client, filename: &str) -> Result<(), Error> {
    let content = fs::read_to_string(filename);

    let resp = client
        .synthesize_speech()
        .output_format(OutputFormat::Mp3)
        .text(content.unwrap())
        .voice_id(VoiceId::Joanna)
        .send()
        .await?;

    // Get MP3 data from response and save it
    let mut blob = resp
        .audio_stream
        .collect()
        .await
        .expect("failed to read data");

    let parts: Vec<&str> = filename.split('.').collect();
    let out_file = format!("{}", String::from(parts[0]), ".mp3");

    let mut file = tokio::fs::File::create(out_file)
        .await
        .expect("failed to create file");
```

```
file.write_all_buf(&mut blob)
    .await
    .expect("failed to write to file");

Ok(())
}
```

- 有关 API 的详细信息，请参阅适用[SynthesizeSpeech](#)于 Rust 的AWS SDK API 参考。

## SAP ABAP

### 适用于 SAP ABAP 的 SDK

#### Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

```
TRY.
  " Only pass optional language code if it has a value
  IF iv_lang_code IS NOT INITIAL.
    oo_result = lo_ply->synthesizespeech(
      iv_engine = iv_engine
      iv_outputformat = iv_output_fmt
      iv_text = iv_text
      iv_voiceid = iv_voice_id
      iv_languagecode = iv_lang_code ).
  ELSE.
    oo_result = lo_ply->synthesizespeech(
      iv_engine = iv_engine
      iv_outputformat = iv_output_fmt
      iv_text = iv_text
      iv_voiceid = iv_voice_id ).
  ENDIF.
  MESSAGE 'Speech synthesized successfully.' TYPE 'I'.
CATCH /aws1/cx_plyinvalidssmlex.
  MESSAGE 'Invalid SSML.' TYPE 'E'.
CATCH /aws1/cx_plylexiconnotfoundex.
  MESSAGE 'Lexicon not found.' TYPE 'E'.
CATCH /aws1/cx_plyservicefailureex.
```

```
MESSAGE 'Service failure occurred.' TYPE 'E'.
CATCH /aws1/cx_plytextlengthexclex.
MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
ENDTRY.
```

- 有关 API 的详细信息，请参阅适用[SynthesizeSpeech](#)于 S AP 的 AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## Amazon Polly 使用的场景 AWS 软件开发工具包

以下代码示例向您展示了如何使用软件开发工具包在 Amazon Polly 中实现常见场景。AWS 这些场景演示了如何通过调用 Amazon Polly 中的多个函数或与其他 AWS 服务结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

### 示例

- [使用将文本转换为语音并转换回文本 AWS SDK](#)
- [使用 Amazon Polly 创建口型同步应用程序 AWS SDK](#)
- [创建用于分析客户反馈和合成音频的应用程序](#)
- [Amazon Polly 入门](#)

## 使用将文本转换为语音并转换回文本 AWS SDK

以下代码示例展示了如何：

- 使用 Amazon Polly 将纯文本 (UTF-8) 输入文件合成音频文件。
- 将音频文件上传到 Amazon S3 存储桶。
- 使用 Amazon Transcribe 将音频文件转换为文本。
- 显示文本。

## Rust

### 适用于 Rust 的 SDK

使用 Amazon Polly 将纯文本 (UTF-8) 输入文件合成音频文件，将音频文件上传到 Amazon S3 存储桶，使用 Amazon Transcribe 将该音频文件转换为文本，然后显示文本。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- Amazon Polly
- Amazon S3
- Amazon Transcribe

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## 使用 Amazon Polly 创建口型同步应用程序 AWS SDK

以下代码示例演示了如何通过 Amazon Polly 创建口型同步应用程序。

## Python

### 适用于 Python 的 SDK ( Boto3 )

演示如何使用 Amazon Polly 和 Tkinter 创建口型同步应用程序，该应用程序显示动画面孔说话以及由 Amazon Polly 合成的语音。Lip-sync 是通过向 Amazon Polly 索取与合成语音相匹配的发声模组列表来完成的。

- 从 Amazon Polly 获取语音元数据并将其显示在 Tkinter 应用程序中。
- 从 Amazon Polly 获取合成语音音频和匹配的视觉语音标记。
- 在播放音频时，同步播放动画表情中的嘴部动作。
- 提交异步合成任务以获取长文本，并从 Amazon Simple Storage Service (Amazon S3) 存储桶检索输出。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- Amazon Polly

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## 创建用于分析客户反馈和合成音频的应用程序

以下代码示例显示如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

### .NET

#### 适用于 .NET 的 SDK

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。
- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目[GitHub](#)。

#### 本示例中使用的服务

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

### Java

#### 适用于 Java 的 SDK 2.x

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。
- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。

本示例中使用的服务

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

## JavaScript

适用于 JavaScript (v3) 的软件开发工具包

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。
- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。以下摘录显示了在 Lambda 函数中 适用于 JavaScript 的 AWS SDK 是如何使用的。

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
```

```
* Determine the language and sentiment of the extracted text.
*
* @param {{ source_text: string}} extractTextOutput
*/
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
  eventBridgeS3Event
 */
```

```
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/textract/latest/dg/API\_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string }}
 * sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
  });
```

```
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string }}
  textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });
```

```
const { TranslatedText } = await translateClient.send(translateCommand);

return { translated_text: TranslatedText };
};
```

### 本示例中使用的服务

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

## Ruby

### 适用于 Ruby 的 SDK

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。
- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。

### 本示例中使用的服务

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

## Amazon Polly 入门

以下代码示例展示了如何：

- 清理资源

### Bash

#### AWS CLI 使用 Bash 脚本

#### Note

还有更多相关信息 GitHub。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon Polly Getting Started Script
# This script demonstrates how to use Amazon Polly with the AWS CLI

set -euo pipefail

# Set up logging
LOG_FILE="polly-tutorial.log"
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
WORK_DIR=$(mktemp -d)
trap 'cleanup_temp' EXIT

cleanup_temp() {
    rm -rf "$WORK_DIR"
}

echo "Starting Amazon Polly tutorial at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
```

```
# Use bash array to safely handle arguments
bash -c "$1" 2>&1 | tee -a "$LOG_FILE" || return $?
}

# Function to check for errors
check_error() {
    if echo "$1" | grep -iq "error"; then
        echo "ERROR detected in output. Exiting script." | tee -a "$LOG_FILE"
        echo "$1" | tee -a "$LOG_FILE"
        return 1
    fi
    return 0
}

# Function to handle errors and cleanup
handle_error() {
    local line_number=$1
    echo "Error occurred at line $line_number. Attempting cleanup..." | tee -a
"$LOG_FILE"
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "" | tee -a "$LOG_FILE"
    echo "===== " | tee -a
"$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "===== " | tee -a
"$LOG_FILE"

    # Delete lexicon if it exists
    if [[ -n "${LEXICON_NAME:-}" ]]; then
        echo "Deleting lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
        if aws polly delete-lexicon --name "$LEXICON_NAME" 2>&1 | tee -a
"$LOG_FILE"; then
            echo "Lexicon deleted successfully." | tee -a "$LOG_FILE"
        else
            echo "Warning: Failed to delete lexicon." | tee -a "$LOG_FILE"
        fi
    fi

    # Remove audio files
```

```
for file in output.mp3 ssm1-output.mp3 lexicon-output.mp3 example.pls; do
    if [[ -f "$file" ]]; then
        rm -f "$file"
        echo "Removed $file" | tee -a "$LOG_FILE"
    fi
done

echo "Cleanup complete." | tee -a "$LOG_FILE"
}

# Trap errors with line number
trap 'handle_error ${LINENO}' ERR

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    echo "AWS CLI is not installed. Please install it first." | tee -a
"$LOG_FILE"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "AWS credentials are not configured. Please configure them first." | tee
-a "$LOG_FILE"
    exit 1
fi

# Step 1: Verify Amazon Polly is available
echo "Step 1: Verifying Amazon Polly availability" | tee -a "$LOG_FILE"
if aws polly describe-voices --query 'Voices[0].Name' --output text &> /dev/null;
then
    echo "Amazon Polly is available. Proceeding with tutorial." | tee -a
"$LOG_FILE"
else
    echo "Amazon Polly is not available in your AWS CLI installation or region."
| tee -a "$LOG_FILE"
    echo "Please update your AWS CLI to the latest version or check your region."
| tee -a "$LOG_FILE"
    exit 1
fi

# Step 2: List available voices
echo "" | tee -a "$LOG_FILE"
echo "Step 2: Listing available voices" | tee -a "$LOG_FILE"
```

```
log_cmd "aws polly describe-voices --language-code en-US --output text --query
  'Voices[0:3].[Id, LanguageCode, Gender]'" || true

# Step 3: Basic text-to-speech conversion
echo "" | tee -a "$LOG_FILE"
echo "Step 3: Converting text to speech" | tee -a "$LOG_FILE"
OUTPUT_FILE="${WORK_DIR}/output.mp3"
POLLY_TEXT="Hello, welcome to Amazon Polly. This is a sample text to speech
  conversion."
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --text
  '$POLLY_TEXT' '$OUTPUT_FILE'" || true

if [[ -f "$OUTPUT_FILE" ]]; then
  echo "Successfully created output.mp3 file." | tee -a "$LOG_FILE"
  echo "You can play this file with your preferred audio player." | tee -a
  "$LOG_FILE"
  cp "$OUTPUT_FILE" output.mp3
else
  echo "Failed to create output.mp3 file." | tee -a "$LOG_FILE"
  exit 1
fi

# Step 4: Using SSML for enhanced speech
echo "" | tee -a "$LOG_FILE"
echo "Step 4: Using SSML for enhanced speech" | tee -a "$LOG_FILE"
SSML_OUTPUT="${WORK_DIR}/ssml-output.mp3"
SSML_TEXT='<speaK>Hello! <break time="1s"/> This is a sample of <emphasis>SSML
  enhanced speech</emphasis>.</speaK>'
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Matthew --
  text-type ssmL --text '$SSML_TEXT' '$SSML_OUTPUT'" || true

if [[ -f "$SSML_OUTPUT" ]]; then
  echo "Successfully created ssml-output.mp3 file." | tee -a "$LOG_FILE"
  echo "You can play this file with your preferred audio player." | tee -a
  "$LOG_FILE"
  cp "$SSML_OUTPUT" ssml-output.mp3
else
  echo "Failed to create ssml-output.mp3 file." | tee -a "$LOG_FILE"
  exit 1
fi

# Step 5: Working with lexicons
echo "" | tee -a "$LOG_FILE"
echo "Step 5: Working with lexicons" | tee -a "$LOG_FILE"
```

```
# Generate a random identifier for the lexicon (max 20 chars, alphanumeric only)
LEXICON_NAME="example$(openssl rand -hex 6 | cut -c 1-10)"
echo "Using lexicon name: $LEXICON_NAME" | tee -a "$LOG_FILE"

# Create a lexicon file
echo "Creating lexicon file..." | tee -a "$LOG_FILE"
LEXICON_FILE="${WORK_DIR}/example.pls"
cat > "$LEXICON_FILE" << 'EOF'
<?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>AWS</grapheme>
    <alias>Amazon Web Services</alias>
  </lexeme>
</lexicon>
EOF

# Upload the lexicon
echo "Uploading lexicon..." | tee -a "$LOG_FILE"
LEXICON_ARN=$(aws polly put-lexicon --name "$LEXICON_NAME" --content
  file://"${LEXICON_FILE}" --query 'LexiconArn' --output text 2>&1) || true
if [[ -n "$LEXICON_ARN" && "$LEXICON_ARN" != "" && ! "$LEXICON_ARN" =~ error ]];
then
  echo "Lexicon uploaded with ARN: $LEXICON_ARN" | tee -a "$LOG_FILE"
  aws polly tag-resource --resource-arn "$LEXICON_ARN" --tags
  Key=project,Value=doc-smith Key=tutorial,Value=amazon-polly-gs 2>&1 | tee -a
  "$LOG_FILE" || true
else
  echo "Lexicon uploaded." | tee -a "$LOG_FILE"
fi

# List available lexicons
echo "Listing available lexicons..." | tee -a "$LOG_FILE"
log_cmd "aws polly list-lexicons --output text --query 'Lexicons[*].[Name]'" ||
  true

# Get details about the lexicon
```

```

echo "Getting details about the lexicon..." | tee -a "$LOG_FILE"
log_cmd "aws polly get-lexicon --name '$LEXICON_NAME' --output text --query
'Lexicon.Name'" || true

# Use the lexicon when synthesizing speech
echo "Using the lexicon for speech synthesis..." | tee -a "$LOG_FILE"
LEXICON_OUTPUT="{WORK_DIR}/lexicon-output.mp3"
LEXICON_TEXT="I work with AWS every day."
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --
lexicon-names '$LEXICON_NAME' --text '$LEXICON_TEXT' '$LEXICON_OUTPUT'" || true

if [[ -f "$LEXICON_OUTPUT" ]]; then
    echo "Successfully created lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
"$LOG_FILE"
    cp "$LEXICON_OUTPUT" lexicon-output.mp3
else
    echo "Failed to create lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Summary of created resources
echo "" | tee -a "$LOG_FILE"
echo "======" | tee -a
"$LOG_FILE"
echo "TUTORIAL SUMMARY" | tee -a "$LOG_FILE"
echo "======" | tee -a
"$LOG_FILE"
echo "Created resources:" | tee -a "$LOG_FILE"
echo "1. Lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
echo "2. Audio files:" | tee -a "$LOG_FILE"
echo "  - output.mp3" | tee -a "$LOG_FILE"
echo "  - ssm1-output.mp3" | tee -a "$LOG_FILE"
echo "  - lexicon-output.mp3" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Cleanup with auto-confirmation
echo "" | tee -a "$LOG_FILE"
echo "======" | tee -a
"$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "======" | tee -a
"$LOG_FILE"
echo "Cleaning up all created resources..." | tee -a "$LOG_FILE"

```

```
cleanup
```

```
echo "" | tee -a "$LOG_FILE"  
echo "Tutorial completed successfully!" | tee -a "$LOG_FILE"  
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。
  - [DeleteLexicon](#)
  - [DescribeVoices](#)
  - [GetLexicon](#)
  - [帮助](#)
  - [ListLexicons](#)
  - [PutLexicon](#)
  - [SynthesizeSpeech](#)

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将 Amazon Polly 与软件开发工具包配合使用 AWS](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

# Amazon Polly 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。[责任共担模式](#)将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。AWS 还为您提供可以安全使用的服务。Third-party 作为[AWS 合规计划](#)的一部分，审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon Polly 的合规计划，请参阅按合规计划提供的[范围内的 AWS 服务按合规计划](#)。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

该文档帮助您了解如何在使用 Amazon Polly 时应用责任共担模式。以下主题说明如何配置 Amazon Polly 以实现您的安全性和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 Amazon Polly 资源。

## 主题

- [Amazon Polly 中的数据保护](#)
- [Amazon Polly 中的 Identity and Access Management](#)
- [Amazon Polly 中的日志记录和监控](#)
- [Amazon Polly 的合规性验证](#)
- [Amazon Polly 中的恢复能力](#)
- [Amazon Polly 中的基础设施安全性](#)
- [Amazon Polly 的安全最佳实践](#)
- [将 Amazon Polly 与接口 VPC 终端节点一起使用](#)

## Amazon Polly 中的数据保护

Amazon Polly 符合[责任 AWS 共担模式](#)，其中包括数据保护的法规和指南。AWS 负责保护运行所有 AWS 服务的全球基础架构。AWS 保持对托管在此基础架构上的数据的控制，包括用于处理客户内容和个人数据的安全配置控制。AWS 客户和 APN 合作伙伴，无论是作为数据控制者还是数据处理者，都应对他们在 AWS 云端存储的任何个人数据负责。

出于数据保护目的，我们建议您保护 AWS 账户凭证并使用 AWS Identity and Access Management (IAM) 设置个人用户，以便仅向每个用户提供履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 ( MFA )。
- 用于 SSL/TLS 与 AWS 资源通信。
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如名称字段）。这包括您使用控制台、API 或软件开发工具包使用 Amazon Polly 或其他 AWS 服务时。AWS CLI 您输入到 Amazon Polly 或其它服务中的任何数据都可能被选取以包含在诊断日志中。当您向外部服务器提供网址时，请勿在网址中包含凭证信息来验证您对该服务器的请求。

有关数据保护的更多信息，请参阅 AWS 安全性博客 上的 [AWS 责任共担模式和 GDPR](#) 博客文章。

## 静态加密

您的 Amazon Polly 语音合成输出可以保存在您自己的系统中。您还可以调用 Amazon Polly，然后使用您选择的任何加密密钥对文件进行加密，并将其存储在 Amazon Simple Storage Service (Amazon S3) 或其他安全存储中。Amazon Polly [the section called “SynthesizeSpeech”](#) 操作是无状态的，并且与客户身份无关。您之后无法从 Amazon Polly 中检索它。

## 传输中加密

在传输过程中，所有文本提交都受 TLS 保护。Amazon Polly 不保留所提交文本的内容。

## 互连网络流量保密性

通过控制台、CLI 或软件开发工具包访问 Amazon Polly。通信利用传输层安全性 (TLS) 会话加密来实现保密性，并使用 [数字签名](#) 来实现身份验证和完整性。

## Amazon Polly 中的 Identity and Access Management

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以通过身份验证（登录）和获得授权（具有权限）来使用 Amazon Polly 资源。您可以使用 IAM AWS 服务，无需支付额外费用。

## 主题

- [受众](#)
- [使用身份进行身份验证](#)
- [使用策略管理访问](#)
- [Amazon Polly 如何与 IAM 配合使用](#)
- [Identity-based Amazon Polly 的政策示例](#)
- [Amazon Polly API 权限：操作、权限和资源参考](#)
- [Amazon Polly 身份和访问问题排查](#)

## 受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅[Amazon Polly 身份和访问问题排查](#)）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅[Amazon Polly 如何与 IAM 配合使用](#)）
- IAM 管理员：编写用于管理访问权限的策略（请参阅[Identity-based Amazon Polly 的政策示例](#)）

## 使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center（例如（IAM Identity Center））、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录用户指南》中的[如何登录您的 AWS 账户](#)。

对于编程访问，AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的[适用于 API 请求的 AWS 签名版本 4](#)。

## AWS 账户 根用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的[需要根用户凭证的任务](#)。

## 联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。AWS IAM Identity Center 有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的[什么是 IAM Identity Center ?](#)。

## IAM 用户和群组

[IAM 用户](#)是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的[要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问](#)。

[IAM 组](#)指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的[IAM 用户使用案例](#)。

## IAM 角色

[IAM 角色](#)是具有特定权限的身份，可提供临时凭证。您可以通过[从用户切换到 IAM 角色 \(控制台\)](#) 或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的[担任角色的方法](#)。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的[IAM 中的跨账户资源访问](#)。

## 使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的[JSON 策略概述](#)。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

## Identity-based 政策

Identity-based 策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的[使用客户管理型策略定义自定义 IAM 权限](#)。

Identity-based 策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的[在托管策略与内联策略之间进行选择](#)。

## Resource-based 政策

Resource-based 策略是您附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中[指定主体](#)。

Resource-based 策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

## 其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《IAM 用户指南》中的[IAM 实体的权限边界](#)。
- 服务控制策略 (SCP) – 指定 AWS Organizations 中组织或组织单元的最大权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的[服务控制策略](#)。
- 资源控制策略 (RCP) – 设置对账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的[资源控制策略 \(RCP\)](#)。
- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅《IAM 用户指南》中的[会话策略](#)。

## 多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的[策略评估逻辑](#)。

## Amazon Polly 如何与 IAM 配合使用

在使用 IAM 管理对 Amazon Polly 的访问权限之前，您应该了解哪些 IAM 功能可用于 Amazon Polly。

## 将 IAM 功能与 Amazon Polly 一起使用

IAM 功能	Amazon Polly 支持
<a href="#">Identity-based 策略</a>	是
<a href="#">Resource-based 策略</a>	否
<a href="#">策略操作</a>	是
<a href="#">策略资源</a>	是
<a href="#">策略条件键 ( 特定于服务 )</a>	否
<a href="#">ACL</a>	否
<a href="#">ABAC ( 策略中的标签 )</a>	否
<a href="#">临时凭证</a>	是
<a href="#">Amazon Polly 的转发访问会话 ( FAS )</a>	是
<a href="#">服务角色</a>	否
<a href="#">Service-linked 角色</a>	否

要全面了解 Amazon Polly 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的[AWS 服务](#)。

### Identity-based Amazon Polly 的政策

支持基于身份的策略：是

Identity-based 策略是您可以附加到身份（例如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的[使用客户管理型策略定义自定义 IAM 权限](#)。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的[IAM JSON 策略元素引用](#)。

### Identity-based Amazon Polly 的政策示例

要查看 Amazon Polly 基于身份的策略的示例，请参阅 [Identity-based Amazon Polly 的政策示例](#)。

## Resource-based 亚马逊 Polly 内部的政策

支持基于资源的策略：否

Resource-based 策略是您附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中[指定主体](#)。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。有关更多信息，请参阅《IAM 用户指南》中的 [IAM 中的跨账户资源访问](#)。

## Amazon Polly 的策略操作

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 Action 元素描述可用于在策略中允许或拒绝访问的操作。在策略中包含操作以授予执行关联操作的权限。

有关 Amazon Polly 操作的列表，请参阅服务授权参考中的 [Amazon Polly 定义的操作](#)。

Amazon Polly 中的策略操作在操作前面使用以下前缀：

```
polly
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [  
  "polly:action1",  
  "polly:action2"  
]
```

要查看 Amazon Polly 基于身份的策略的示例，请参阅 [Identity-based Amazon Polly 的政策示例](#)。

## Amazon Polly 的策略资源

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 [Amazon 资源名称 \( ARN \)](#) 指定资源。对于不支持资源级权限的操作，请使用通配符 (\*) 指示语句应用于所有资源。

```
"Resource": "*" 
```

要查看 Amazon Polly 的资源类型及其 ARN 的列表，请参阅服务授权参考中的 [由 Amazon Polly 定义的资源](#)。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 [Amazon Polly 定义的操作](#)。

要查看 Amazon Polly 基于身份的策略的示例，请参阅 [Identity-based Amazon Polly 的政策示例](#)。

## Amazon Polly 的策略条件键

支持特定于服务的策略条件键：否

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

Condition 元素根据定义的条件指定语句何时执行。您可以创建使用 [条件运算符](#) ( 例如，等于或小于 ) 的条件表达式，以使策略中的条件与请求中的值相匹配。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的 [AWS 全局条件上下文密钥](#)。

要查看 Amazon Polly 条件键的列表，请参阅服务授权参考中的 [Amazon Polly 的条件键](#)。要了解您可以对哪些操作和资源使用条件键，请参阅 [Amazon Polly 定义的操作](#)。

要查看 Amazon Polly 基于身份的策略的示例，请参阅 [Identity-based Amazon Polly 的政策示例](#)。

## Amazon Polly 中的 ACL

支持 ACL：否

访问控制列表 ( ACL ) 控制哪些主体 ( 账户成员、用户或角色 ) 有权访问资源。ACL 与基于资源的策略类似，但它们不使用 JSON 策略文档格式。

## 使用 Amazon Polly 的 ABAC

支持 ABAC ( 策略中的标签 ) : 否

Attribute-based 访问控制 (ABAC) 是一种授权策略，它根据称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的 [条件元素](#) 中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的 [使用 ABAC 授权定义权限](#)。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的 [使用基于属性的访问权限控制 \( ABAC \)](#)。

## 将临时凭证用于 Amazon Polly

支持临时凭证 : 是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 [IAM 中的临时安全凭证](#) 和 [使用 IAM 的。AWS 服务](#)

## Cross-service 适用于 Amazon Polly 的转发访问会话 (FAS)

支持转发访问会话 ( FAS ) : 是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅 [转发访问会话](#)。

## Amazon Polly 的服务角色

支持服务角色 : 否

服务角色是由一项服务担任、代表您执行操作的 [IAM 角色](#)。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的 [创建向 AWS 服务委派权限的角色](#)。

### ⚠ Warning

更改服务角色的权限可能会破坏 Amazon Polly 的功能。仅当 Amazon Polly 提供相关指导时才编辑服务角色。

## Service-linked Amazon Polly 的角色

支持服务相关角色：否

服务相关角色是一种与服务相关联的 AWS 服务角色。该服务可以代替您执行操作。Service-linked 角色出现在您的，AWS 账户 并且归服务所有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅[能够与 IAM 搭配使用的 AWS 服务](#)。在表中查找 Service-linked 角色列 Yes 中包含的服务。选择是链接以查看该服务的服务相关角色文档。

## Amazon Polly IAM 角色

您可以将基于身份的权限策略附加到 IAM 角色以授予跨账户权限。例如，账户 A 中的管理员可以创建一个角色来向另一个 AWS 账户（例如账户 B）或 AWS 服务授予跨账户权限，如下所示：

1. 账户 A 管理员可以创建一个 IAM 角色，然后向该角色附加授予其访问账户 A 中资源的权限策略。
2. 账户 A 管理员可以向角色挂载信任策略，将账户 B 标识为能够担任该角色的委托人。
3. 然后，账户 B 管理员可以将代入该角色的权限委托给账户 B 中的任何用户。这样，账户 B 中的用户就可以创建或访问账户 A 中的资源。如果您想向 AWS 服务授予担任该角色的权限，则信任策略中的委托人也可以是 AWS 服务委托人。

有关使用 IAM 委托权限的更多信息，请参阅《IAM 用户指南》中的[访问权限管理](#)。

下面是一个示例策略，该策略授予相应的权限，用于放置和获取字典以及列出这些当前可用的词典。

Amazon Polly 支持资源级别的操作 Identity-based 策略。在某些情况下，资源可能受 ARN 的限制。这适用于 SynthesizeSpeech、StartSpeechSynthesisTask、PutLexicon、GetLexicon 和 DeleteLexicon 操作。在这些情况下，Resource 值由 ARN 指示。例如：`arn:aws:polly:us-east-2:account-id:lexicon/*` 是 Resource 值在 us-east-2 区域内所有词典上指定的权限。

但是，并非所有操作都使用

ARN。DescribeVoices、ListLexicons、GetSpeechSynthesisTasks 和 ListSpeechSynthesisTasks 操作就是这种情况。

有关用户、组、角色和权限的更多信息，请参阅《IAM 用户指南》<https://docs.aws.amazon.com/IAM/latest/UserGuide/id.html>中的身份（用户、组和角色）。

## Identity-based Amazon Polly 的政策示例

原定设置情况下，用户和角色没有创建或修改 Amazon Polly 资源的权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的[创建 IAM 策略（控制台）](#)。

有关 Amazon Polly 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅服务授权参考中的[Amazon Polly 的操作、资源和条件键](#)。

### 主题

- [策略最佳实践](#)
- [使用 Amazon Polly 控制台](#)
- [允许用户查看他们自己的权限](#)
- [AWS 适用于 Amazon Polly 的托管（预定义）策略](#)
- [Amazon Polly 更新至 AWS 托管策略](#)
- [Customer-managed 策略示例](#)

## 策略最佳实践

Identity-based 策略决定了是否有人可以在您的账户中创建、访问或删除 Amazon Polly 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的 AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的[AWS 托管策略](#)或[工作职能的 AWS 托管策略](#)。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的[IAM 中的策略和权限](#)。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使

用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 [IAM JSON 策略元素：条件](#)。

- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性：IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 [使用 IAM Access Analyzer 验证策略](#)。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 [使用 MFA 保护 API 访问](#)。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 [IAM 中的安全最佳实践](#)。

## 使用 Amazon Polly 控制台

要访问 Amazon Polly 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 Amazon Polly 资源的详细信息。AWS 账户如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用 Amazon Polly 控制台，还需要为实体附加 Amazon *ConsoleAccess* Polly *ReadOnly* AWS 或托管策略。有关更多信息，请参阅《IAM 用户指南》中的 [为用户添加权限](#)。

要使用 Amazon Polly 控制台，请向所有 Amazon Polly API 授予权限。不需要额外的权限。要获取完整的控制台功能，您可以使用以下策略：

### 允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
```

```

        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
},
{
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
    ],
    "Resource": "*"
}
]
}

```

## AWS 适用于 Amazon Polly 的托管（预定义）策略

AWS 通过提供由创建和管理的独立 IAM 策略来解决许多常见用例 AWS。这些 AWS 托管策略为常见用例授予必要的权限，这样您就可以不必调查需要哪些权限。有关更多信息，请参阅《IAM 用户指南》中的 [AWS 托管式策略](#)。

以下 AWS 托管政策仅适用于 Amazon Polly，您可以将其附加到账户中的用户：

- **AmazonPollyReadOnlyAccess**— 授予对资源的只读访问权限，允许列出词典、获取词典、列出可用语音和合成语音（包括将词典应用于合成语音）。

要查看有关策略（包括 JSON 策略文档的最新版本）的更多信息，请参阅《AWS 托管式策略参考指南》中的 [AmazonPollyReadOnlyAccess](#)。

- **AmazonPollyFullAccess**— 授予对资源和所有支持的操作的完全访问权限。

要查看有关策略（包括 JSON 策略文档的最新版本）的更多信息，请参阅《AWS 托管式策略参考指南》中的 [AmazonPollyFullAccess](#)。

**Note**

您可以通过登录到 IAM 控制台并在该控制台中搜索特定策略来查看这些权限策略。

此外，您还可以创建您自己的自定义 IAM 策略，以授予 Amazon Polly 操作和资源的相关权限。您可以将这些自定义策略附加到需要这些权限的 IAM 用户或组。

## Amazon Polly 更新至 AWS 托管策略

查看自该服务开始跟踪这些更改以来，Amazon Polly AWS 托管策略更新的详细信息。要获取有关此页面变更的自动提醒，请订阅 Amazon Polly 文档历史记录页面上的 RSS 提要。

### Amazon Polly 更新至 AWS 托管策略

Policy	更改	日期
AmazonPollyReadOnlyAccess	更新了托管策略-增加了支持双向流式语音合成功能的权限。polly:StartSpeechSynthesisStream	2026 年 3 月 19 日

## Customer-managed 策略示例

本节的用户策略示例介绍如何授予各 Amazon Polly 操作的权限。当您使用 AWS 软件开发工具包或 CLI 时，这些策略会起作用。当您使用控制台时，请向所有 Amazon Polly API 授予权限。

**Note**

所有示例都使用 us-east-2 区域和虚构的账户 ID。

### 示例

- [示例 1：允许所有 Amazon Polly 操作](#)
- [示例 2：允许所有 Amazon Polly 操作，但以下操作除外 DeleteLexicon](#)
- [示例 3：允许 DeleteLexicon](#)
- [示例 4：允许在指定区域中删除词典](#)

- [示例 5：允许使用 DeleteLexicon 指定的词典](#)

### 示例 1：允许所有 Amazon Polly 操作

在您注册后（请参阅 [Amazon Polly 入门](#)），创建管理员用户来管理您的账户，包括创建用户和管理用户权限。

您可以创建对所有 Amazon Polly 操作具有权限的用户。将此用户视为特定于服务的管理员，以便使用 Amazon Polly。您可以将以下权限策略附加到该用户。

#### JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowAllPollyActions",
    "Effect": "Allow",
    "Action": [
      "polly:*"
    ],
    "Resource": "*"
  }
]
```

### 示例 2：允许所有 Amazon Polly 操作，但以下操作除外 DeleteLexicon

以下权限策略用于授予用户执行所有操作（DeleteLexicon 除外）的用户权限，以及在所有区域中删除显式拒绝的权限。

### 示例 3：允许 DeleteLexicon

以下权限策略用于授予删除任何词典用户的权限，无论其位于哪个项目或区域中。

#### JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowDeleteLexicon",
```

```
    "Effect": "Allow",
    "Action": [
      "polly:DeleteLexicon"],
    "Resource": "*"
  }
]
```

#### 示例 4：允许在指定区域中删除词典

以下权限策略授予用户删除您位于单个区域（在本例中为 us-east-2）的任何项目中您拥有的任何词典的权限。

#### JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowDeleteSpecifiedRegion",
    "Effect": "Allow",
    "Action": [
      "polly:DeleteLexicon"],
    "Resource": "arn:aws:polly:us-east-2:111122223333:lexicon/*"
  ]
}
```

#### 示例 5：允许使用 DeleteLexicon 指定的词典

以下权限策略授予用户删除您在特定区域（在本例中为 us-east-2）中拥有的特定词典（在本例中为 myLexicon）的权限。

#### JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowDeleteForSpecifiedLexicon",
    "Effect": "Allow",
```

```
"Action": [
  "polly:DeleteLexicon"],
"Resource": "arn:aws:polly:us-east-2:111122223333:lexicon/myLexicon"
}
]
```

## Amazon Polly API 权限：操作、权限和资源参考

在设置可附加到 IAM 身份的权限策略（基于身份的策略）时，可使用下面的列表作为参考。包括每个 Amazon Polly API 操作、您可以为其授予执行该操作的权限的相应操作以及您可以为其授予权限的 AWS 资源。您可以在策略的 Action 字段中指定这些操作，并在策略的 Resource 字段中指定资源值。

您可以在 Amazon Polly 政策中使用 AWS 全局条件键来表达条件。有关 AWS 范围密钥的完整列表，请参阅 IAM 用户指南中的 [可用密钥](#)。

### Note

要指定操作，请在 API 操作名称之前使用 polly 前缀（例如，polly:GetLexicon）。

Amazon Polly 支持资源级别的操作 Identity-based 策略。因此，Resource 由 ARN 指定。例如：arn:aws:polly:us-east-2:account-id:lexicon/\* 是 Resource 值在 us-east-2 区域内所有词典上指定的权限。

由于 Amazon Polly 不支持资源级的操作权限，大多数策略指定通配符 (\*) 作为 Resource 值。但是，如果需要将权限应限制到特定区域，则可将通配符替换为适当的 ARN：arn:aws:polly:region:account-id:lexicon/\*。

### Amazon Polly API 和所需的操作权限

API 操作：[DeleteLexicon](#)

所需权限（API 操作）：polly:DeleteLexicon

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

**API 操作：** [DescribeVoices](#)

所需权限 ( API 操作 ) : polly:DescribeVoices

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

**API 操作：** [GetLexicon](#)

所需权限 ( API 操作 ) : polly:GetLexicon

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

**API 操作：** [ListLexicons](#)

所需权限 ( API 操作 ) : polly:ListLexicons

资源 : arn:aws:polly:*region*:*account-id*:lexicon/\*

**API 操作：** [PutLexicon](#)

所需权限 ( API 操作 ) : polly:ListLexicons

资源 : \*

**API 操作：** [SynthesizeSpeech](#)

所需权限 ( API 操作 ) : polly:SynthesizeSpeech

资源 : \*

## Amazon Polly 身份和访问问题排查

可以使用以下信息，以帮助您诊断和修复在使用 Amazon Polly 和 IAM 时可能遇到的常见问题。

### 主题

- [我无权在 Amazon Polly 中执行操作](#)
- [我无权执行 iam : PassRole](#)
- [我想允许我以外的人进入 AWS 账户 访问我的 Amazon Polly 资源](#)

### 我无权在 Amazon Polly 中执行操作

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 polly:*GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
polly:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 polly:*GetWidget* 操作访问 *my-example-widget* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

## 我无权执行 iam : PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 Amazon Polly。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 marymajor 的 IAM 用户尝试使用控制台在 Amazon Polly 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

## 我想允许我以外的人进入 AWS 账户 访问我的 Amazon Polly 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 ( ACL ) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon Polly 是否支持这些功能，请参阅 [Amazon Polly 如何与 IAM 配合使用](#)。

- 要了解如何提供对您拥有的资源的访问权限 AWS 账户，请参阅 [IAM 用户指南中的向您拥有 AWS 账户的另一个 IAM 用户提供访问权限](#)。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 [IAM 用户指南中的向第三方提供访问权限](#)。AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的 [为经过外部身份验证的用户 \(身份联合验证\) 提供访问权限](#)。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 [IAM 中的跨账户资源访问](#)。

## Amazon Polly 中的日志记录和监控

要保持 Amazon Polly 应用程序的可靠性、可用性和性能，监控是一个重要环节。要监控 Amazon Polly API 调用，你可以使用。AWS CloudTrail 要监控您的任务状态，请使用 Amazon CloudWatch 日志。

- Amazon CloudWatch Alarms — 使用 CloudWatch 警报，您可以监控您指定的时间段内的单个指标。如果指标超过给定阈值，则会向 Amazon 简单通知服务主题或 AWS Auto Scaling 政策发送通知。CloudWatch 当指标处于特定状态时，警报不会调用操作。而是必须在状态已改变并在指定的若干个时间段内保持不变后才调用。有关更多信息，请参阅 [将 CloudWatch 与 Amazon Polly 集成](#)。
- CloudTrail 日志 — CloudTrail 提供用户、角色或 AWS 服务在 Amazon Polly 中执行的操作的记录。使用收集的信息 CloudTrail，您可以确定向 Amazon Polly 提出的请求。还可以确定发出请求的源 IP 地址、请求方、请求时间以及其他详细信息。有关更多信息，请参阅 [使用 AWS CloudTrail 记录 Amazon Polly API 调用](#)。

## Amazon Polly 的合规性验证

Third-party 作为多项合规计划的一部分，审计师评估 Amazon Polly 的安全与 AWS 合规性。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

有关特定合规计划范围内的 AWS 服务列表，请参阅按合规计划划分的 [范围内的 AWS AWS 服务按合规计划](#)。有关一般信息，请参阅 [AWS 合规计划 AWS](#)。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅 [在 AWS Artifact 中下载报告](#)。

您在使用 Amazon Polly 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。AWS 提供了以下资源来帮助实现合规性：

- [安全性与合规性快速入门指南](#) - 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- [HIPAA 安全与合规架构白皮书 — 本白皮书](#) 描述了公司如何使用来创建应用程序。AWS HIPAA-compliant
- [AWS 合规资源 AWS](#) — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- [使用 AWS Config 开发人员指南中的规则评估资源](#) — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- [AWS Security Hub CSPM](#) — 此 AWS 服务可全面了解您的安全状态 AWS，帮助您检查是否符合安全行业标准和最佳实践。

## Amazon Polly 中的恢复能力

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。AWS 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错能力和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅[AWS 全球基础设施](#)。

## Amazon Polly 中的基础设施安全性

作为一项托管服务，Amazon Polly 受[亚马逊网络服务：安全流程概述白皮书中描述的 AWS 全球网络安全程序](#)的保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 Amazon Polly。客户端必须支持传输层安全性协议 ( TLS ) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如短暂的 (DHE) 或椭圆曲线短暂的 Diffie-Hellman (ECDHE)。Diffie-Hellman 大多数现代系统 ( 如 Java 7 及更高版本 ) 都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用[AWS Security Token Service](#) ( AWS STS ) 生成临时安全凭证来对请求进行签名。

## Amazon Polly 的安全最佳实践

您的信任、隐私和内容的安全性是我们最重视的问题。我们会实施负责任的、先进的技术和物理控制，以防止对您的内容进行未经授权的访问或披露，并确保我们依照对您的承诺使用您的内容。有关更多信息，请参阅[AWS 数据隐私常见问题](#)。

Amazon Polly 不保留所提交文本的内容。

要全面了解 AWS 安全性，包括合规性、渗透测试、公告和资源，请访问[AWS 云安全](#)网站。

## 将 Amazon Polly 与接口 VPC 终端节点一起使用

如果您使用亚马逊虚拟私有云 ( 亚马逊 VPC ) 托管 AWS 资源，则可以在您的 VPC 和 Amazon Polly 之间建立私有连接。您可以使用此连接与 Amazon Polly 合成语音，而无需遍历公共互联网。

Amazon VPC 是一项 AWS 服务，可用于在您定义的虚拟网络中启动 AWS 资源。借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。要将您的 VPC 连接到 Amazon Polly，请为 Amazon Polly 定义一个接口 VPC 终端节点。这种类型的终端节点使您能够将 VPC 连接到 AWS 服务。该终端节点提供了到 Amazon Polly 的可靠、可扩展的连接，无需互联网网关、网络地址转换 (NAT) 实例或 VPN 连接。有关更多信息，请参阅 Amazon VPC 用户指南中的[什么是 Amazon VPC](#)。

接口 VPC 终端节点由 AWS PrivateLink 一种 AWS 技术提供支持，该技术允许在 AWS 服务使用弹性网络接口与私有 IP 地址之间进行私有通信。有关更多信息，请参阅[新建-f AWS PrivateLink or AWS 服务](#)。

以下步骤适用于 Amazon VPC 的用户。有关更多信息，请参阅 Amazon VPC 用户指南中的[入门](#)。

### 可用性

所有[支持 Amazon Polly 的区域](#)均支持 VPC 终端节点。有关 AWS 区域和可用区的更多信息，请参阅[AWS 全球基础设施](#)。

FIPS 端点在以下区域可用：

- 美国东部 ( 弗吉尼亚州北部 ) ( us-east-1 )
- 美国东部 ( 俄亥俄州 ) (us-east-2)
- 美国西部 ( 加利福尼亚北部 ) (us-west-1)
- 美国西部 ( 俄勒冈州 ) (us-west-2)
- AWS GovCloud (US-West) (us-gov-west-1)
- 加拿大 ( 中部 ) (ca-central-1)

FIPS 端点的格式为 `com.amazonaws.REGION.polly-fips`。

## 为 Amazon Polly 创建 VPC 终端节点

要开始将您的 Amazon Polly 与 VPC 一起使用，请为 Amazon Polly 创建接口 VPC 终端节点。可供选择的服务是 `com.amazonaws.Region.polly`。您不需要更改任何 Amazon Polly 设置。有关更多信息，请参阅 Amazon VPC 用户指南中的[创建接口终端节点](#)。

## 测试您的 VPC 和 Amazon Polly 之间的连接

创建端点后，您可以测试连接。

测试 VPC 和 Amazon Polly 终端节点之间的连接

1. 连接到位于您的 VPC 中的 Amazon EC2 实例。有关连接的信息，请参阅 Amazon EC2 文档中的[连接到您的 Linux 实例](#)或[连接到您的 Windows 实例](#)。
2. 在实例中，使用 AWS CLI 中的 `aws polly describe-voices` 来列出可用的 Amazon Polly 语音。

如果对该命令的响应包含可用的 Amazon Polly 语音列表，则该命令已成功执行，并且您的 VPC 终端节点正常运行。

## 控制对 Amazon Polly 终端节点的访问

VPC 终端节点策略是一种 IAM 资源策略，您在创建或修改端点时可将它附加到端点。如果在创建端点时未附加策略，我们将为您附加默认策略以允许对服务进行完全访问。端点策略不会覆盖或替换 IAM 用户策略或服务特定的策略。这是一个单独的策略，用于控制从端点中对指定服务进行的访问。

端点策略必须采用 JSON 格式编写。

有关更多信息，请参阅 Amazon VPC 用户指南中的[使用 VPC 终端节点控制对服务的访问](#)。

下面是用于 Amazon Polly 的端点策略示例。该策略允许通过 VPC 连接到 Amazon Polly 的用户描述语音和合成语音，并禁止他们执行其他 Amazon Polly 操作。

```
{
  "Statement": [
    {
      "Sid": "SynthesisAndDescribeVoicesOnly",
      "Principal": "*",
      "Action": [
        "polly:DescribeVoices",
```

```
    "polly:SynthesizeSpeech"  
  ],  
  "Effect": "Allow",  
  "Resource": "*"   
}   
]   
}
```

## 修改 Amazon Polly 的 VPC 终端节点策略

1. 打开位于 <https://console.aws.amazon.com/vpc> 的 Amazon VPC 控制台。
2. 在导航窗格中，选择端点。
3. 如果还没有为 Amazon Polly 创建终端节点，请选择创建终端节点。然后选择 com.amazonaws。  
**Region**.polly 并选择创建端点。
4. 选择 com.amazonaws。**Region**.polly 端点，然后选择屏幕下半部分的“策略”选项卡。
5. 选择编辑策略并对策略进行更改。

## 对 VPC 上下文键的支持

Amazon Polly 支持可用于限制对特定 VPC 或特定 VPC 终端节点进行访问的 `aws:SourceVpc` 和 `aws:SourceVpce` 上下文键。这些键仅当用户使用 VPC 终端节点时才起作用。有关更多信息，请参阅 IAM 用户指南中的 [可用于部分服务的键](#)。

# 使用 AWS CloudTrail 记录 Amazon Polly API 调用

Amazon Polly 与 AWS CloudTrail 集成，后者是在 Amazon Polly 中提供用户、角色或 AWS 服务所采取操作的记录的服务。CloudTrail 将 Amazon Polly 的所有 API 调用作为事件捕获。捕获调用中包括通过 Amazon Polly 控制台的调用和对 Amazon Polly API 操作的代码调用。如果您创建跟踪，则可以让 CloudTrail 事件持续传送到 Amazon S3 存储桶，包括 Amazon Polly 的事件。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的事件历史记录中查看最新事件。使用通过 CloudTrail 收集的信息，您可以确定向 Amazon Polly 发出了什么请求、发出请求时使用的 IP 地址、何人发出的请求、请求的发出时间以及其它详细信息。

要了解有关 CloudTrail 的更多信息（包括如何对其进行配置和启用），请参阅 [AWS CloudTrail 用户指南](#)。

## CloudTrail 中的 Amazon Polly 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 Amazon Polly 中发生受支持的事件活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在事件历史记录中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅 [使用 CloudTrail 事件历史记录查看事件](#)。

要持续记录 AWS 账户中的事件（包括 Amazon Polly 的事件），请创建跟踪。通过跟踪记录，CloudTrail 可将日志文件传送到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 AWS 区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- [创建跟踪概述](#)
- [CloudTrail 支持的服务和集成](#)
- [为 CloudTrail 配置 Amazon SNS 通知](#)
- [从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件](#)

Amazon Polly 支持将以下操作记录为 CloudTrail 日志文件中的事件：

- [DeleteLexicon](#)
- [DescribeVoices](#)
- [GetLexicon](#)

- [GetSpeechSynthesisTask](#)
- [ListLexicons](#)
- [ListSpeechSynthesisTasks](#)
- [PutLexicon](#)
- [StartSpeechSynthesisTask](#)
- [SynthesizeSpeech](#)

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是 AWS Identity and Access Management (IAM) 用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 [CloudTrail userIdentity 元素](#)。

## 示例：Amazon Polly 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日记账条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 SynthesizeSpeech。

```
{
  "Records": [
    {
      "awsRegion": "us-east-2",
      "eventID": "19bd70f7-5e60-4cdc-9825-936c552278ae",
      "eventName": "SynthesizeSpeech",
      "eventSource": "polly.amazonaws.com",
      "eventTime": "2016-11-02T03:49:39Z",
      "eventType": "AwsApiCall",
      "eventVersion": "1.05",
      "recipientAccountId": "123456789012",
      "requestID": "414288c2-a1af-11e6-b17f-d7cfc06cb461",
      "requestParameters": {
```

```
"lexiconNames": [
    "SampleLexicon"
],
"engine": "neural",
"outputFormat": "mp3",
"sampleRate": "22050",
"text": "*****",
"textType": "text",
"voiceId": "Kendra"
},
"responseElements": null,
"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"
}
}
]
}
```

## 将 CloudWatch 与 Amazon Polly 集成

当您与 Amazon Polly 交互时，它每一分钟向 CloudWatch 发送一次以下指标和维度。您可以使用以下流程查看 Amazon Polly 的指标。

您可以使用 Amazon CloudWatch 监控 Amazon Polly，此工具可从 Amazon Polly 收集原始数据，并将其处理为易读的近乎实时的指标。这些统计数据会保存两周，从而使您能够访问 `historical information`，并能够更好地了解您的 Web 应用程序或服务的执行情况。默认情况下，Amazon Polly 指标数据以 1 分钟为间隔发送到 CloudWatch。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的[什么是 Amazon CloudWatch？](#)。

### 获取 CloudWatch 指标（控制台）

1. 通过 <https://console.aws.amazon.com/cloudwatch/> 打开 CloudWatch 控制台。
2. 在导航窗格中，选择指标。
3. 在按类别显示的 CloudWatch 指标窗格中，在 Amazon Polly 的指标类别下，选择一个指标类别，然后在上方窗格中，向下滚动以查看完整的指标列表。

### 在 AWS CLI 上获取 CloudWatch 指标

以下代码显示 Amazon Polly 的可用指标。

```
aws cloudwatch list-metrics --namespace "AWS/Polly"
```

前面的命令会返回类似于以下内容的 Amazon Polly 指标的列表。此 `MetricName` 元素标识出是哪种参数。

```
{
  "Metrics": [
    {
      "Namespace": "AWS/Polly",
      "Dimensions": [
        {
          "Name": "Operation",
          "Value": "SynthesizeSpeech"
        }
      ]
    }
  ],
```

```

    "MetricName": "ResponseLatency"
  },
  {
    "Namespace": "AWS/Polly",
    "Dimensions": [
      {
        "Name": "Operation",
        "Value": "SynthesizeSpeech"
      }
    ],
    "MetricName": "RequestCharacters"
  }
}

```

有关更多信息，请参阅 Amazon CloudWatch API 参考中的 [GetMetricStatistics](#)。

## Amazon Polly 指标

Amazon Polly 为每个请求生成以下指标。这些指标经过汇总后每隔一分钟会发送到 CloudWatch，这些指标在 CloudWatch 处可用。

指标	描述
RequestCharacters	<p>请求中的字符数。这仅是可计费字符，不包括 SSML 标签。</p> <p>有效维度：操作</p> <p>有效统计数据：平均值、最小值、最大值、样本计数、总和</p> <p>单位：个</p>
ResponseLatency	<p>从发出请求到开始流式处理响应之间的延迟。</p> <p>有效的维度：操作</p> <p>有效统计数据：最小值、最大值、平均值、样本计数</p> <p>单位：微秒</p>
2XXCount	成功响应后返回的 HTTP 200 级代码。

指标	描述
	<p>有效的维度：操作</p> <p>有效统计数据：平均值、样本计数、总和</p> <p>单位：个</p>
4XXCount	<p>出现错误时返回的 HTTP 400 级错误代码。每一次成功响应都会发出一个零 (0)。</p> <p>有效的维度：操作</p> <p>有效统计数据：平均值、样本计数、总和</p> <p>单位：个</p>
5XXCount	<p>出现错误时返回的 HTTP 500 级错误代码。每一次成功响应都会发出一个零 (0)。</p> <p>有效的维度：操作</p> <p>有效统计数据：平均值、样本计数、总和</p> <p>单位：个</p>

## Amazon Polly 指标的维度

Amazon Polly 指标使用 AWS/Polly 命名空间并提供以下维度的指标：

维度	描述
Operation	指标按其所指的 API 方法进行分组。可能的值为 <code>SynthesizeSpeech</code> 、 <code>PutLexicon</code> 、 <code>DescribeVoices</code> 等。

# Amazon Polly API 参考

Amazon Polly 是一项 Web 服务，可让您轻松地用文本合成语音。

Amazon Polly 服务提供 API 操作，能够用纯文本和语音合成标记语言 ( SSML ) 来合成高质量语音，并管理发音词典，使您能够在自己的应用领域中获得极佳结果。

已验证的 API 调用都必须使用签名版本 4 签名流程签名。有关更多信息，请参阅中的[签署 AWS API 请求 Amazon Web Services 一般参考](#)。

## 主题

- [操作](#)
- [数据类型](#)
- [常见错误类型](#)
- [常见参数](#)

## 操作

支持以下操作：

- [DeleteLexicon](#)
- [DescribeVoices](#)
- [GetLexicon](#)
- [GetSpeechSynthesisTask](#)
- [ListLexicons](#)
- [ListSpeechSynthesisTasks](#)
- [PutLexicon](#)
- [StartSpeechSynthesisStream](#)
- [StartSpeechSynthesisTask](#)
- [SynthesizeSpeech](#)

## DeleteLexicon

删除 AWS 区域中存储的指定的发音词典。已删除的词典无法用于语音合成，也无法使用 `GetLexicon` 或 `ListLexicon` APIs 进行检索。

有关更多信息，请参阅 [管理词典](#)。

### 请求语法

```
DELETE /v1/lexicons/LexiconName HTTP/1.1
```

### URI 请求参数

请求使用以下 URI 参数。

#### LexiconName

要删除的词典的名称。必须是该地区的现有词典。

模式：`[0-9A-Za-z]{1,20}`

必需：是

### 请求体

该请求没有请求正文。

### 响应语法

```
HTTP/1.1 200
```

### 响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

### 错误

#### LexiconNotFoundException

Amazon Polly 找不到指定的词典。这可能是由于词典缺失、名称拼写错误或指定了位于不同区域的词典所致。

验证词典是否存在、位于该区域 ( 参阅 [ListLexicons](#) ) ，以及词典名称拼写是否正确。然后请重试。

HTTP 状态代码 : 404

ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码 : 500

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs ，请参阅以下内容：

- [AWS 命令行界面 V2](#)
- [AWS 适用于 .NET 的 SDK V4](#)
- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版软件开发工具包](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)
- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## DescribeVoices

返回在请求语音合成时可用的语音列表。每个语音都说一种特定的语言，可以是男性语音也可以是女性语音，并由 ID（即语音名称的 ASCII 版本）标识。

合成语音 (SynthesizeSpeech) 时，您需要从 DescribeVoices 返回的语音列表中提供所需的语音 ID。

例如，您希望新闻阅读器应用程序以特定语言阅读新闻，但允许用户选择语音。使用 DescribeVoices 操作，您可以为用户提供可供选择的可用语音列表。

您可以选择指定语言代码以筛选可用语音。例如，如果您指定 en-US，则该操作将返回所有可用美国英语语音的列表。

此操作需要执行 polly:DescribeVoices 操作的权限。

### 请求语法

```
GET /v1/voices?  
Engine=Engine&IncludeAdditionalLanguageCodes=IncludeAdditionalLanguageCodes&LanguageCode=LanguageCode  
HTTP/1.1
```

### URI 请求参数

请求使用以下 URI 参数。

#### [Engine](#)

指定 Amazon Polly 在处理输入文本进行语音合成时使用的引擎 ( standard、neural、long-form 或 generative )。

有效值 : standard | neural | long-form | generative

#### [IncludeAdditionalLanguageCodes](#)

布尔值，表示是否返回使用指定语言作为附加语言的双语语音。例如，如果您请求所有使用美国英语 (es-US) 的语言，并且有一个意大利语语音同时说意大利语 (it-IT) 和美国英语，那么如果您指定 yes，则将包含该语音，但如果您指定 no，则不会包含该语音。

#### [LanguageCode](#)

用于筛选返回的语音列表的语言标识标签 ( 语言名称的 ISO 639 代码 : ISO 3166 国家/地区代码 )。如果不指定此可选参数，则返回所有可用的语音。

有效值 : arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS  
| en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT  
| ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO  
| ru-RU | sv-SE | tr-TR | en-NZ | en-ZA | ca-ES | de-AT | yue-CN | ar-AE  
| fi-FI | en-IE | nl-BE | fr-BE | cs-CZ | de-CH | en-SG

## [NextToken](#)

从上一次 DescribeVoices 操作返回的不透明分页标记。如果存在，则表示在哪里继续列出。

长度限制：最小长度为 0。最大长度为 4096。

## 请求正文

该请求没有请求正文。

## 响应语法

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

{
  "NextToken": "string",
  "Voices": [
    {
      "AdditionalLanguageCodes": [ "string" ],
      "Gender": "string",
      "Id": "string",
      "LanguageCode": "string",
      "LanguageName": "string",
      "Name": "string",
      "SupportedEngines": [ "string" ]
    }
  ]
}
```

## 响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

## NextToken

在下一个请求中使用的分页标记，用于继续列出语音。仅当响应被截断时才会返回 NextToken。

类型：字符串

长度限制：最小长度为 0。最大长度为 4096。

## Voices

语音及其属性的列表。

类型：[Voice](#) 对象数组

## 错误

### InvalidNextTokenException

NextToken 无效。确保拼写正确，然后请重试。

HTTP 状态代码：400

### ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码：500

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 命令行界面 V2](#)
- [AWS 适用于 .NET 的 SDK V4](#)
- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版软件开发工具包](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)

- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

# GetLexicon

返回 AWS 区域中存储的指定的发音词典的内容。有关更多信息，请参阅[管理词典](#)。

## 请求语法

```
GET /v1/lexicons/LexiconName HTTP/1.1
```

## URI 请求参数

请求使用以下 URI 参数。

### LexiconName

词典的名称。

模式：`[0-9A-Za-z]{1,20}`

必需：是

## 请求体

该请求没有请求正文。

## 响应语法

```
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
  }
}
```

```
}  
}
```

## 响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

### [Lexicon](#)

提供词典名称和字符串内容的词典对象。

类型：[Lexicon](#) 对象

### [LexiconAttributes](#)

词典的元数据，包括使用的音标字母、语言代码、词典 ARN、词典中定义的词素数量以及以字节为单位的词典大小。

类型：[LexiconAttributes](#) 对象

## 错误

### LexiconNotFoundException

Amazon Polly 找不到指定的词典。这可能是由于词典缺失、名称拼写错误或指定了位于不同区域的词典所致。

验证词典是否存在、位于该区域（参阅 [ListLexicons](#)），以及词典名称拼写是否正确。然后请重试。

HTTP 状态代码：404

### ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码：500

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 命令行界面 V2](#)
- [AWS 适用于 .NET 的 SDK V4](#)
- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版 SDK](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)
- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## GetSpeechSynthesisTask

根据特定 SpeechSynthesisTask 对象的 taskID 检索该对象。此对象包含有关给定语音合成任务的信息，包括任务的状态，以及指向包含任务输出的 S3 存储桶的链接。

### 请求语法

```
GET /v1/synthesisTasks/TaskId HTTP/1.1
```

### URI 请求参数

请求使用以下 URI 参数。

#### TaskId

Amazon Polly 生成的语音合成任务的标识符。

模式：`^[a-zA-Z0-9_-]{1,100}$`

必需：是

### 请求体

该请求没有请求正文。

### 响应语法

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

{
  "SpeechSynthesisTask": {
    "CreationTime": number,
    "Engine": "string",
    "LanguageCode": "string",
    "LexiconNames": [ "string" ],
    "OutputFormat": "string",
    "OutputUri": "string",
    "RequestCharacters": number,
    "SampleRate": "string",
    "SnsTopicArn": "string",
    "SpeechMarkTypes": [ "string" ],
```

```
    "TaskId": "string",
    "TaskStatus": "string",
    "TaskStatusReason": "string",
    "TextType": "string",
    "VoiceId": "string"
  }
}
```

## 响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

### [SynthesisTask](#)

SynthesisTask 对象，它提供所请求任务的信息，包括输出格式、创建时间、任务状态等。

类型：[SynthesisTask](#) 对象

## 错误

### InvalidTaskIdException

提供的任务 ID 无效。请提供有效的任务 ID，然后重试。

HTTP 状态代码：400

### ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码：500

### SynthesisTaskNotFoundException

找不到具有请求任务 ID 的语音合成任务。

HTTP 状态代码：400

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 命令行界面 V2](#)
- [AWS 适用于 .NET 的 SDK V4](#)
- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版软件开发工具包](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)
- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## ListLexicons

返回 AWS 区域中存储的发音词典的列表。有关更多信息，请参阅[管理词典](#)。

### 请求语法

```
GET /v1/lexicons?NextToken=NextToken HTTP/1.1
```

### URI 请求参数

请求使用以下 URI 参数。

#### [NextToken](#)

从上一次 ListLexicons 操作返回的不透明分页标记。如果存在，则表示在哪里继续列出词典。

长度限制：最小长度为 0。最大长度为 4096。

### 请求正文

该请求没有请求正文。

### 响应语法

```
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"  
}
```

## 响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

### [Lexicons](#)

词典名称和属性的列表。

类型：[LexiconDescription](#) 对象数组

### [NextToken](#)

在下一个请求中使用的分页标记，用于继续列出词典。仅当响应被截断时才会返回 NextToken。

类型：字符串

长度限制：最小长度为 0。最大长度为 4096。

## 错误

### InvalidNextTokenException

NextToken 无效。确保拼写正确，然后请重试。

HTTP 状态代码：400

### ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码：500

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 命令行界面 V2](#)

- [AWS 适用于 .NET 的 SDK V4](#)
- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版软件开发工具包](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)
- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## ListSpeechSynthesisTasks

返回按创建日期排序的 `SpeechSynthesisTask` 对象列表。此操作可以按任务状态筛选任务，例如，允许用户仅列出已完成的任務。

### 请求语法

```
GET /v1/synthesisTasks?MaxResults=MaxResults&NextToken=NextToken&Status=Status HTTP/1.1
```

### URI 请求参数

请求使用以下 URI 参数。

#### MaxResults

在列表操作中返回的最大语音合成任务数。

有效范围：最小值为 1。最大值为 100。

#### NextToken

在下一个请求中使用的分页标记，用于继续列出语音合成任务。

长度限制：最小长度为 0。最大长度为 4096。

#### Status

在列表操作中返回的语音合成任务的状态

有效值：`scheduled` | `inProgress` | `completed` | `failed`

### 请求正文

该请求没有请求正文。

### 响应语法

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

```
{
  "NextToken": "string",
  "SynthesisTasks": [
    {
      "CreationTime": number,
      "Engine": "string",
      "LanguageCode": "string",
      "LexiconNames": [ "string" ],
      "OutputFormat": "string",
      "OutputUri": "string",
      "RequestCharacters": number,
      "SampleRate": "string",
      "SnsTopicArn": "string",
      "SpeechMarkTypes": [ "string" ],
      "TaskId": "string",
      "TaskStatus": "string",
      "TaskStatusReason": "string",
      "TextType": "string",
      "VoiceId": "string"
    }
  ]
}
```

## 响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

### [NextToken](#)

此请求中从上一次列表操作返回的不透明分页标记。如果存在，则表示在哪里继续列出。

类型：字符串

长度限制：最小长度为 0。最大长度为 4096。

### [SynthesisTasks](#)

提供列表请求中指定任务信息的 SynthesisTask 对象列表，包括输出格式、创建时间、任务状态等。

类型：[SynthesisTask](#) 对象数组

## 错误

### InvalidNextTokenException

NextToken 无效。确保拼写正确，然后请重试。

HTTP 状态代码：400

### ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码：500

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 命令行界面 V2](#)
- [AWS 适用于 .NET 的 SDK V4](#)
- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版软件开发工具包](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)
- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## PutLexicon

在 AWS 区域中存储一个发音词典。如果区域中已存在同名词典，则新词典将覆盖该词典。词典操作具有最终一致性，因此，可能需要一段时间才能使用该词典。SynthesizeSpeech

有关更多信息，请参阅[管理词典](#)。

### 请求语法

```
PUT /v1/lexicons/LexiconName HTTP/1.1
Content-type: application/json

{
  "Content": "string"
}
```

### URI 请求参数

请求使用以下 URI 参数。

#### LexiconName

词典的名称。名称必须遵循正则表达式格式 `[0-9A-Za-z]{1,20}`。也就是说，该名称是一个区分大小写的字母数字字符串，长度最多为 20 个字符。

模式：`[0-9A-Za-z]{1,20}`

必需：是

### 请求体

请求接受采用 JSON 格式的以下数据。

#### Content

PLS 词典的内容作为字符串数据。

类型：字符串

是否必需：是

## 响应语法

```
HTTP/1.1 200
```

## 响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

## 错误

### InvalidLexiconException

Amazon Polly 找不到指定的词典。确保词典名称拼写正确，然后请重试。

HTTP 状态代码：400

### LexiconSizeExceededException

此操作将超过指定词典的最大大小。

HTTP 状态代码：400

### MaxLexemeLengthExceededException

此操作将超过词素的最大大小。

HTTP 状态代码：400

### MaxLexiconsNumberExceededException

此操作将超过最大词典数量。

HTTP 状态代码：400

### ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码：500

### UnsupportedPIsAlphabetException

词典指定的字母不受支持。有效值为 x-sampa 和 ipa。

HTTP 状态代码：400

## UnsupportedPlsLanguageException

不支持词典中指定的语言。有关受支持的语言的列表，请参阅[词典属性](#)。

HTTP 状态代码：400

### 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 命令行界面 V2](#)
- [AWS 适用于 .NET 的 SDK V4](#)
- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版软件开发工具包](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)
- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## StartSpeechSynthesisStream

通过双向流媒体连接合成 UTF-8 输入、纯文本或 SSML。在 HTTP/2 标头中指定合成参数，将文本作为输入流上的事件递增发送，并在合成音频可用时接收合成的音频。

此操作可作为双向SynthesizeSpeech对应物：

- [SynthesizeSpeech](#)

### 请求语法

```
POST /v1/synthesisStream HTTP/1.1
x-amzn-Engine: Engine
x-amzn-LanguageCode: LanguageCode
x-amzn-LexiconNames: LexiconNames
x-amzn-OutputFormat: OutputFormat
x-amzn-SampleRate: SampleRate
x-amzn-VoiceId: VoiceId
Content-type: application/json

{
  "CloseStreamEvent": {
  },
  "TextEvent": {
    "FlushStreamConfiguration": {
      "Force": boolean
    },
    "Text": "string",
    "TextType": "string"
  }
}
```

### URI 请求参数

请求使用以下 URI 参数。

#### [Engine](#)

指定 Amazon Polly 在处理语音合成输入文本时使用的引擎。当前，仅支持该generative引擎。如果您指定了所选引擎不支持的声音，Amazon Polly 将返回错误。

有效值：standard | neural | long-form | generative

是否必需：是

### [LanguageCode](#)

一个可选参数，用于设置语音合成请求的语言代码。只有在使用双语语音时才指定此参数。如果使用双语语音但未指定语言代码，则 Amazon Polly 将使用双语语音的默认语言。

有效值：arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR | en-NZ | en-ZA | ca-ES | de-AT | yue-CN | ar-AE | fi-FI | en-IE | nl-BE | fr-BE | cs-CZ | de-CH | en-SG

### [LexiconNames](#)

服务在合成过程中要应用的一个或多个发音词典的名称。只有当词典语言与语音语言匹配时，Amazon Polly 才会应用词典。

数组成员：最多 5 项。

模式：`[0-9A-Za-z]{1,20}`

### [OutputFormat](#)

合成语音的音频格式。目前，Amazon Polly 不支持 JSON 语音标记。

有效值：json | mp3 | ogg\_opus | ogg\_vorbis | pcm | mulaw | alaw

是否必需：是

### [SampleRate](#)

音频频率，以 Hz 为单位指定。

### [Voiceld](#)

合成中要使用的声音。要获取可用语音列表 IDs，请使用[DescribeVoices](#)操作。

有效值：Aditi | Amy | Astrid | Bianca | Brian | Camila | Carla | Carmen | Celine | Chantal | Conchita | Cristiano | Dora | Emma | Enrique | Ewa | Filiz | Gabrielle | Geraint | Giorgio | Gwyneth | Hans | Ines | Ivy | Jacek | Jan | Joanna | Joey | Justin | Karl | Kendra | Kevin | Kimberly | Lea | Liv | Lotte | Lucia | Lupe | Mads | Maja | Marlene | Mathieu | Matthew | Maxim | Mia | Miguel | Mizuki | Naja | Nicole | Olivia | Penelope | Raveena | Ricardo | Ruben | Russell | Salli | Seoyeon |

Takumi | Tatyana | Vicki | Vitoria | Zeina | Zhiyu | Aria | Ayanda |  
Arlet | Hannah | Arthur | Daniel | Liam | Pedro | Kajal | Hiujin | Laura  
| Elin | Ida | Suvi | Ola | Hala | Andres | Sergio | Remi | Adriano  
| Thiago | Ruth | Stephen | Kazuha | Tomoko | Niamh | Sofie | Lisa |  
Isabelle | Zayd | Danielle | Gregory | Burcu | Jitka | Sabrina | Jasmine  
| Jihye | Ambre | Beatrice | Florian | Lennart | Lorenzo | Tiffany

是否必需：是

## 请求体

请求接受采用 JSON 格式的以下数据。

### CloseStreamEvent

一个表示输入流结束的事件。

类型：[CloseStreamEvent](#) 对象

必需：否

### TextEvent

包含待合成内容的文本事件。

类型：[TextEvent](#) 对象

必需：否

## 响应语法

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

{
  "AudioEvent": {
    "AudioChunk": blob
  },
  "ServiceFailureException": {
  },
  "ServiceQuotaExceededException": {
  },
}
```

```
"StreamClosedEvent": {
  "RequestCharacters": number
},
"ThrottlingException": {
},
"ValidationException": {
}
}
```

## 响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

### [AudioEvent](#)

包含合成语音的音频事件。

类型：[AudioEvent](#) 对象

### [ServiceFailureException](#)

未知情况导致服务故障。

类型：异常

HTTP 状态代码：500

### [ServiceQuotaExceededException](#)

表示将超过服务配额的异常。

类型：异常

HTTP 状态代码：402

### [StreamClosedEvent](#)

带有摘要信息的事件，表示直播已关闭。

类型：[StreamClosedEvent](#) 对象

### [ThrottlingException](#)

表示请求已被限制的异常。

类型：异常

HTTP 状态代码：400

### [ValidationException](#)

表示输入未通过验证的异常。

类型：异常

HTTP 状态代码：400

## 错误

### ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码：500

### ServiceQuotaExceededException

该请求会导致超出服务限额。

quotaCode

标识特定配额的配额代码。

serviceCode

标识源服务的服务代码。

HTTP 状态代码：402

### ThrottlingException

由于请求限制，该请求被拒绝。

throttlingReasons

解释请求被限制的原因列表。

HTTP 状态代码：400

### ValidationException

输入未能满足 服务指定的约束。

## fields

导致验证错误的字段。

## reason

请求验证失败的原因。

HTTP 状态代码：400

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 命令行界面 V2](#)
- [AWS 适用于 .NET 的 SDK V4](#)
- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版软件开发工具包](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)
- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## StartSpeechSynthesisTask

允许通过启动新 `SpeechSynthesisTask` 来创建异步合成任务。此操作需要语音合成所需的所有标准信息，以及用于存储合成任务输出的 Amazon S3 存储桶的名称和两个可选参数 (`OutputS3KeyPrefix` 和 `SnsTopicArn`)。合成任务创建后，此操作将返回一个 `SpeechSynthesisTask` 对象，其中将包括该任务的标识符以及当前状态。启动异步合成任务后，该 `SpeechSynthesisTask` 对象在 72 小时内可用。

### 请求语法

```
POST /v1/synthesisTasks HTTP/1.1
Content-type: application/json

{
  "Engine": "string",
  "LanguageCode": "string",
  "LexiconNames": [ "string" ],
  "OutputFormat": "string",
  "OutputS3BucketName": "string",
  "OutputS3KeyPrefix": "string",
  "SampleRate": "string",
  "SnsTopicArn": "string",
  "SpeechMarkTypes": [ "string" ],
  "Text": "string",
  "TextType": "string",
  "VoiceId": "string"
}
```

### URI 请求参数

该请求不使用任何 URI 参数。

### 请求正文

请求接受采用 JSON 格式的以下数据。

#### Engine

指定 Amazon Polly 在处理输入文本进行语音合成时使用的引擎 (`standard`、`neural`、`long-form` 或 `generative`)。使用不受所选引擎支持的语音会导致错误。

类型：字符串

有效值：standard | neural | long-form | generative

必需：否

### LanguageCode

语音合成请求的可选语言代码。只有在使用双语语音（例如 Aditi）时才需设置，Aditi 可用于印度英语 (en-IN) 或印地语 (hi-IN)。

如果使用双语语音但未指定语言代码，则 Amazon Polly 将使用双语语音的默认语言。任何语音的默认语言都是 [DescribeVoices](#) 操作返回的 LanguageCode 参数语言。例如，如果未指定语言代码，Aditi 将使用印度英语而不是印地语。

类型：字符串

有效值：arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS  
| en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT  
| ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO  
| ru-RU | sv-SE | tr-TR | en-NZ | en-ZA | ca-ES | de-AT | yue-CN | ar-AE  
| fi-FI | en-IE | nl-BE | fr-BE | cs-CZ | de-CH | en-SG

必需：否

### LexiconNames

您希望服务在合成过程中应用的一个或多个发音词典名称的列表。仅当词典的语言与语音的语言相同时，才会应用词典。

类型：字符串数组

数组成员：最多 5 项。

模式：`[0-9A-Za-z]{1,20}`

必需：否

### OutputFormat

返回的输出将采用的编码格式。对于音频流，这将是 mp3、ogg\_vorbis、ogg\_opus、mu-law、a-law 或 pcm。对于语音标记，格式为 JSON。

类型：字符串

有效值：json | mp3 | ogg\_opus | ogg\_vorbis | pcm | mulaw | alaw

是否必需：是

### OutputS3BucketName

将保存输出文件的 Amazon S3 存储桶名称。

类型：字符串

模式：`^[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9]$`

是否必需：是

### OutputS3KeyPrefix

输出语音文件的 Amazon S3 键前缀。

类型：字符串

模式：`^[0-9a-zA-Z\^!\-_\.\*\'\(\):;\$@=+\,\?&]{0,800}$`

必需：否

### SampleRate

指定的音频频率，单位为 Hz。

对于 MP3 和 OGG Vorbis，有效值为“8000”、“16000”、“22050”和“24000”。标准语音的默认值为“22050”。神经语音的默认值为“24000”。长篇语音的默认值为“24000”。生成式语音的默认值为“24000”。

对于 PCM，有效值为“8000”和“16000”。默认值为“16000”。

ogg\_opus 的有效值为“48000”。

mu-law 和 a-law 的有效值为“8000”。

类型：字符串

必需：否

### SnsTopicArn

SNS 主题的 ARN，可用于为语音合成任务提供状态通知。

类型：字符串

模式：`^arn:aws(-(cn|iso(-b)?|us-gov))?:sns:[a-z0-9_-]{1,50}:\d{12}:[a-zA-Z0-9_-]{1,251}([a-zA-Z0-9_-]{0,5}|\.fifo)$`

必需：否

### [SpeechMarkTypes](#)

为输入文本返回的语音标记的类型。

类型：字符串数组

数组成员：最多 4 项。

有效值：`sentence | ssm1 | viseme | word`

必需：否

### [Text](#)

要合成的输入文本。如果您将 `ssml` 指定为 `TextType`，请按照 SSML 格式输入文本。

类型：字符串

是否必需：是

### [TextType](#)

指定输入文本是纯文本还是 SSML。默认值为纯文本。

类型：字符串

有效值：`ssml | text`

必需：否

### [Voiceld](#)

要用于合成的语音 ID。

类型：字符串

有效值：`Aditi | Amy | Astrid | Bianca | Brian | Camila | Carla | Carmen | Celine | Chantal | Conchita | Cristiano | Dora | Emma | Enrique | Ewa | Filiz | Gabrielle | Geraint | Giorgio | Gwyneth | Hans | Ines | Ivy | Jacek | Jan | Joanna | Joey | Justin | Karl | Kendra | Kevin | Kimberly | Lea | Liv | Lotte | Lucia | Lupe | Mads | Maja | Marlene | Mathieu`

| Matthew | Maxim | Mia | Miguel | Mizuki | Naja | Nicole | Olivia  
| Penelope | Raveena | Ricardo | Ruben | Russell | Salli | Seoyeon |  
Takumi | Tatyana | Vicki | Vitoria | Zeina | Zhiyu | Aria | Ayanda |  
Arlet | Hannah | Arthur | Daniel | Liam | Pedro | Kajal | Hiujin | Laura  
| Elin | Ida | Suvi | Ola | Hala | Andres | Sergio | Remi | Adriano  
| Thiago | Ruth | Stephen | Kazuha | Tomoko | Niamh | Sofie | Lisa |  
Isabelle | Zayd | Danielle | Gregory | Burcu | Jitka | Sabrina | Jasmine  
| Jihye | Ambre | Beatrice | Florian | Lennart | Lorenzo | Tiffany

是否必需：是

## 响应语法

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

{
  "SynthesisTask": {
    "CreationTime": number,
    "Engine": "string",
    "LanguageCode": "string",
    "LexiconNames": [ "string" ],
    "OutputFormat": "string",
    "OutputUri": "string",
    "RequestCharacters": number,
    "SampleRate": "string",
    "SnsTopicArn": "string",
    "SpeechMarkTypes": [ "string" ],
    "TaskId": "string",
    "TaskStatus": "string",
    "TaskStatusReason": "string",
    "TextType": "string",
    "VoiceId": "string"
  }
}
```

## 响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

## SynthesisTask

SynthesisTask 对象，提供有关新提交的语音合成任务的信息和属性。

类型：[SynthesisTask](#) 对象

### 错误

#### EngineNotSupportedException

此引擎与您指定的语音不兼容。选择与引擎兼容的新语音，或者更换引擎并重新开始操作。

HTTP 状态代码：400

#### InvalidS3BucketException

提供的 Amazon S3 存储桶名称无效。请检查您输入的 S3 存储桶命名要求并重试。

HTTP 状态代码：400

#### InvalidS3KeyException

提供的 Amazon S3 键前缀无效。请提供有效的 S3 对象键名称。

HTTP 状态代码：400

#### InvalidSampleRateException

指定的采样率无效。

HTTP 状态代码：400

#### InvalidSnsTopicArnException

提供的 SNS 主题 ARN 无效。请提供有效的 SNS 主题 ARN 并重试。

HTTP 状态代码：400

#### InvalidSsmlException

您提供的 SSML 无效。验证 SSML 语法、标签和值拼写，然后重试。

HTTP 状态代码：400

#### LanguageNotSupportedException

Amazon Polly 目前不支持使用指定的语言。

HTTP 状态代码：400

### LexiconNotFoundExpection

Amazon Polly 找不到指定的词典。这可能是由于词典缺失、名称拼写错误或指定了位于不同区域的词典所致。

验证词典是否存在、位于该区域（参阅 [ListLexicons](#)），以及词典名称拼写是否正确。然后请重试。

HTTP 状态代码：404

### MarksNotSupportedForFormatExpection

所选 `OutputFormat` 不支持语音标记。语音标记仅适用于 `json` 格式的内容。

HTTP 状态代码：400

### ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码：500

### SsmlMarksNotSupportedForTextTypeException

纯文本类型输入不支持 SSML 语音标记。

HTTP 状态代码：400

### TextLengthExceededException

“文本”参数的值长于可接受的限制值。对于 `SynthesizeSpeech` API，输入文本的总长度限制为最多 6000 个字符，其中计费字符不能超过 3000 个。对于 `StartSpeechSynthesisTask` API，最大值为 20 万个字符，其中计费字符数不得超过 10 万个。SSML 标签不会算作计费字符。

HTTP 状态代码：400

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 命令行界面 V2](#)
- [AWS 适用于 .NET 的 SDK V4](#)

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版软件开发工具包](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)
- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## SynthesizeSpeech

将 UTF-8 输入、纯文本或 SSML 合成为字节流。SSML 输入必须是有效的、格式正确的 SSML。除非使用音素映射，否则某些字母可能无法用于所有语音（例如，英语语音可能根本无法读取西里尔语）。有关更多信息，请参阅[工作原理](#)。

### 请求语法

```
POST /v1/speech HTTP/1.1
Content-type: application/json

{
  "Engine": "string",
  "LanguageCode": "string",
  "LexiconNames": [ "string" ],
  "OutputFormat": "string",
  "SampleRate": "string",
  "SpeechMarkTypes": [ "string" ],
  "Text": "string",
  "TextType": "string",
  "VoiceId": "string"
}
```

### URI 请求参数

该请求不使用任何 URI 参数。

### 请求正文

请求接受采用 JSON 格式的以下数据。

#### Engine

指定 Amazon Polly 在处理输入文本进行语音合成时使用的引擎（standard、neural、long-form 或 generative）。提供您选定的语音所支持的引擎。如果您未提供引擎，则默认情况下选择标准引擎。如果标准引擎不支持所选语音，则会导致出错。有关 Amazon Polly 语音以及每个引擎可用的语音的信息，请参阅[可用语音](#)。

类型：字符串

有效值：standard | neural | long-form | generative

必需：否

## LanguageCode

合成语音请求的可选语言代码。只有在使用双语语音（例如 Aditi）时才需设置，Aditi 可用于印度英语 (en-IN) 或印地语 (hi-IN)。

如果使用双语语音但未指定语言代码，则 Amazon Polly 将使用双语语音的默认语言。任何语音的默认语言都是 [DescribeVoices](#) 操作返回的 LanguageCode 参数语言。例如，如果未指定语言代码，Aditi 将使用印度英语而不是印地语。

类型：字符串

有效值：arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO | ru-RU | sv-SE | tr-TR | en-NZ | en-ZA | ca-ES | de-AT | yue-CN | ar-AE | fi-FI | en-IE | nl-BE | fr-BE | cs-CZ | de-CH | en-SG

必需：否

## LexiconNames

您希望服务在合成过程中应用的一个或多个发音词典名称的列表。仅当词典的语言与语音的语言相同时，才会应用词典。有关存储词典的信息，请参见 [PutLexicon](#)。

类型：字符串数组

数组成员：最多 5 项。

模式：`[0-9A-Za-z]{1,20}`

必需：否

## OutputFormat

返回的输出将采用的编码格式。对于音频流，这将是 mp3、ogg\_vorbis、ogg\_opus、mu-law、a-law 或 pcm。对于语音标记，格式为 JSON。

使用 pcm 时，返回的内容采用带符号的 audio/pcm 16 位、1 通道 (mono)、小端格式。

类型：字符串

有效值：json | mp3 | ogg\_opus | ogg\_vorbis | pcm | mulaw | alaw

是否必需：是

### SampleRate

指定的音频频率，单位为 Hz。

对于 mp3 和 ogg\_vorbis，有效值为“8000”、“16000”、“22050”、“24000”、“44100”和“48000”。标准语音的默认值为“22050”。神经语音的默认值为“24000”。长篇语音的默认值为“24000”。生成式语音的默认值为“24000”。

对于 PCM，有效值为“8000”和“16000”。默认值为“16000”。

ogg\_opus 的有效值为“48000”。

mu-law 和 a-law 的有效值为“8000”。

类型：字符串

必需：否

### SpeechMarkTypes

为输入文本返回的语音标记的类型。

类型：字符串数组

数组成员：最多 4 项。

有效值：sentence | ssm1 | viseme | word

必需：否

### Text

输入要合成的文本。如果指定 ssm1 为 TextType，请按照 SSML 格式输入文本。

类型：字符串

是否必需：是

### TextType

指定输入文本是纯文本还是 SSML。默认值为纯文本。有关更多信息，请参阅[使用 SSML](#)。

类型：字符串

有效值：ssm1 | text

必需：否

## Voiceld

要用于合成的语音 ID。您可以 IDs 通过拨打[DescribeVoices](#)操作来获取可用语音列表。

类型：字符串

有效值：Aditi | Amy | Astrid | Bianca | Brian | Camila | Carla | Carmen | Celine | Chantal | Conchita | Cristiano | Dora | Emma | Enrique | Ewa | Filiz | Gabrielle | Geraint | Giorgio | Gwyneth | Hans | Ines | Ivy | Jacek | Jan | Joanna | Joey | Justin | Karl | Kendra | Kevin | Kimberly | Lea | Liv | Lotte | Lucia | Lupe | Mads | Maja | Marlene | Mathieu | Matthew | Maxim | Mia | Miguel | Mizuki | Naja | Nicole | Olivia | Penelope | Raveena | Ricardo | Ruben | Russell | Salli | Seoyeon | Takumi | Tatyana | Vicki | Vitoria | Zeina | Zhiyu | Aria | Ayanda | Arlet | Hannah | Arthur | Daniel | Liam | Pedro | Kajal | Hiujin | Laura | Elin | Ida | Suvi | Ola | Hala | Andres | Sergio | Remi | Adriano | Thiago | Ruth | Stephen | Kazuha | Tomoko | Niamh | Sofie | Lisa | Isabelle | Zayd | Danielle | Gregory | Burcu | Jitka | Sabrina | Jasmine | Jihye | Ambre | Beatrice | Florian | Lennart | Lorenzo | Tiffany

是否必需：是

## 响应语法

```
HTTP/1.1 200
Content-Type: ContentType
x-amzn-RequestCharacters: RequestCharacters

AudioStream
```

## 响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

响应将返回以下 HTTP 标头。

## ContentType

指定音频流的类型。这应该反映您的请求中的 `OutputFormat` 参数。

- 如果您请求 mp3 为 OutputFormat，则 ContentType 返回的是音频/MPEG。
- 如果您请求 ogg\_vorbis 为 OutputFormat，则 ContentType 返回的是音频/OGG。
- 如果您请求 ogg\_opus 为 OutputFormat，则 ContentType 返回的是音频/OGG。
- 如果您请求 pcm 为 OutputFormat，则 ContentType 返回的格式为带符号的 audio/pcm 16 位、1 通道 (mono)、小端字节序格式。
- 如果您请求 mu-law 为 OutputFormat，则 ContentType 返回的是音频/mulaw。
- 如果您 a-law 以的身份请求 OutputFormat，则 ContentType 返回的是音频/alaw。
- 如果您请求 json 为 OutputFormat，则 ContentType 返回的是应用程序/ x-json-stream。

## RequestCharacters

合成的字符数。

响应将以下内容作为 HTTP 正文返回。

## AudioStream

包含合成语音的流。

## 错误

### EngineNotSupportedException

此引擎与您指定的语音不兼容。选择与引擎兼容的新语音，或者更换引擎并重新开始操作。

HTTP 状态代码：400

### InvalidSampleRateException

指定的采样率无效。

HTTP 状态代码：400

### InvalidSsmlException

您提供的 SSML 无效。验证 SSML 语法、标签和值拼写，然后重试。

HTTP 状态代码：400

### LanguageNotSupportedException

Amazon Polly 目前不支持使用指定的语言。

HTTP 状态代码：400

### LexiconNotFoundException

Amazon Polly 找不到指定的词典。这可能是由于词典缺失、名称拼写错误或指定了位于不同区域的词典所致。

验证词典是否存在、位于该区域（参阅 [ListLexicons](#)），以及词典名称拼写是否正确。然后请重试。

HTTP 状态代码：404

### MarksNotSupportedForFormatException

所选 `OutputFormat` 不支持语音标记。语音标记仅适用于 `json` 格式的内容。

HTTP 状态代码：400

### ServiceFailureException

未知情况导致服务故障。

HTTP 状态代码：500

### SsmlMarksNotSupportedForTextTypeException

纯文本类型输入不支持 SSML 语音标记。

HTTP 状态代码：400

### TextLengthExceededException

“文本”参数的值长于可接受的限制值。对于 `SynthesizeSpeech` API，输入文本的总长度限制为最多 6000 个字符，其中计费字符不能超过 3000 个。对于 `StartSpeechSynthesisTask` API，最大值为 20 万个字符，其中计费字符数不得超过 10 万个。SSML 标签不会算作计费字符。

HTTP 状态代码：400

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 命令行界面 V2](#)
- [AWS 适用于 .NET 的 SDK V4](#)

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Go v2 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS JavaScript V3 版软件开发工具包](#)
- [AWS 适用于 Kotlin 的 SDK](#)
- [AWS 适用于 PHP 的 SDK V3](#)
- [AWS Python 软件开发工具包](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## 数据类型

支持以下数据类型：

- [AudioEvent](#)
- [CloseStreamEvent](#)
- [FlushStreamConfiguration](#)
- [Lexicon](#)
- [LexiconAttributes](#)
- [LexiconDescription](#)
- [StartSpeechSynthesisStreamActionStream](#)
- [StartSpeechSynthesisStreamEventStream](#)
- [StreamClosedEvent](#)
- [SynthesisTask](#)
- [TextEvent](#)
- [ThrottlingReason](#)
- [ValidationExceptionField](#)
- [Voice](#)

# AudioEvent

包含一大块合成的音频数据。

## 内容

### AudioChunk

以参数指定的格式编码的合成音频数据块。OutputFormat

类型：Base64 编码的二进制数据对象

必需：否

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## CloseStreamEvent

表示输入流的结束。发送此事件后，输入流将被关闭并返回所有音频。

### 内容

此例外结构的成员取决于上下文。

### 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

# FlushStreamConfiguration

控制何时在输出流上发送合成音频数据的配置。

## 内容

### Force

指定是否强制合成引擎立即将缓冲的音频数据写入输出流。

类型：布尔值

必需：否

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

# Lexicon

以字符串格式提供词典名称和词典内容。有关更多信息，请参阅[发音词典规范 \(PLS\) 1.0 版](#)。

## 目录

### Content

字符串格式的词典内容。词典的内容必须采用 PLS 格式。

类型：字符串

必需：否

### Name

词典的名称。

类型：字符串

模式：`[0-9A-Za-z]{1,20}`

必需：否

## 另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- [适用于 C++ 的 AWS SDK](#)
- [适用于 Java V2 的 AWS SDK](#)
- [适用于 Ruby V3 的 AWS SDK](#)

## LexiconAttributes

包含描述词典的元数据，例如词素数量、语言代码等。有关更多信息，请参阅[管理词典](#)。

### 内容

#### Alphabet

词典中使用的语音字母。有效值为 ipa 和 x-sampa。

类型：字符串

必需：否

#### LanguageCode

词典适用的语言代码。例如，语言代码为“en”的词典将应用于所有英语语言（en-GB、en-US、en-AUS、en-WLS 等）。

类型：字符串

有效值：arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS  
| en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT  
| ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO  
| ru-RU | sv-SE | tr-TR | en-NZ | en-ZA | ca-ES | de-AT | yue-CN | ar-AE  
| fi-FI | en-IE | nl-BE | fr-BE | cs-CZ | de-CH | en-SG

必需：否

#### LastModified

上次修改词典的日期（时间戳值）。

类型：时间戳

必需：否

#### LexemesCount

词典中的词素数量。

类型：整数

必需：否

## LexiconArn

词典的 Amazon 资源名称 (ARN)。

类型：字符串

必需：否

## Size

词典的总大小 (以字符为单位)。

类型：整数

必需：否

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

# LexiconDescription

描述词典的内容。

## 目录

### Attributes

提供词典元数据。

类型：[LexiconAttributes](#) 对象

必需：否

### Name

词典的名称。

类型：字符串

模式：`[0-9A-Za-z]{1,20}`

必需：否

## 另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- [适用于 C++ 的 AWS SDK](#)
- [适用于 Java V2 的 AWS SDK](#)
- [适用于 Ruby V3 的 AWS SDK](#)

# StartSpeechSynthesisStreamActionStream

入站事件流，用于发送输入和控制事件以管理双向语音合成。

## 内容

### CloseStreamEvent

一个表示输入流结束的事件。

类型：[CloseStreamEvent](#) 对象

必需：否

### TextEvent

包含待合成内容的文本事件。

类型：[TextEvent](#) 对象

必需：否

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

# StartSpeechSynthesisStreamEventStream

包含合成音频数据和流状态事件的出站事件流。

## 内容

### AudioEvent

包含合成语音的音频事件。

类型：[AudioEvent](#) 对象

必需：否

### ServiceFailureException

未知情况导致服务故障。

类型：异常

HTTP 状态代码：500

必需：否

### ServiceQuotaExceededException

表示将超过服务配额的异常。

类型：异常

HTTP 状态代码：402

必需：否

### StreamClosedEvent

带有摘要信息的事件，表示直播已关闭。

类型：[StreamClosedEvent](#) 对象

必需：否

### ThrottlingException

表示请求已被限制的异常。

类型：异常

HTTP 状态代码：400

必需：否

### ValidationException

表示输入未通过验证的异常。

类型：异常

HTTP 状态代码：400

必需：否

### 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## StreamClosedEvent

表示合成流已关闭并提供摘要信息。

### 内容

#### RequestCharacters

直播会话期间合成的角色总数。

类型：整数

必需：否

### 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

# SynthesisTask

SynthesisTask 提供有关语音合成任务信息的对象。

## 内容

### CreationTime

合成任务启动时间的时间戳。

类型：时间戳

必需：否

### Engine

指定 Amazon Polly 在处理输入文本进行语音合成时使用的引擎 ( standard、neural、long-form 或 generative )。使用不受所选引擎支持的语音会导致错误。

类型：字符串

有效值：standard | neural | long-form | generative

必需：否

### LanguageCode

合成任务的可选语言代码。只有在使用双语语音 ( 例如 Aditi ) 时才需设置，Aditi 可用于印度英语 (en-IN) 或印地语 (hi-IN)。

如果使用双语语音但未指定语言代码，则 Amazon Polly 将使用双语语音的默认语言。任何语音的默认语言都是 [DescribeVoices](#) 操作返回的 LanguageCode 参数语言。例如，如果未指定语言代码，Aditi 将使用印度英语而不是印地语。

类型：字符串

有效值：arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS  
| en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT  
| ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO  
| ru-RU | sv-SE | tr-TR | en-NZ | en-ZA | ca-ES | de-AT | yue-CN | ar-AE  
| fi-FI | en-IE | nl-BE | fr-BE | cs-CZ | de-CH | en-SG

必需：否

## LexiconNames

您希望服务在合成过程中应用的一个或多个发音词典名称的列表。仅当词典的语言与语音的语言相同时，才会应用词典。

类型：字符串数组

数组成员：最多 5 项。

模式：`[0-9A-Za-z]{1,20}`

必需：否

## OutputFormat

返回的输出将采用的编码格式。对于音频流，这将是 mp3、ogg\_vorbis、ogg\_opus、mu-law、a-law 或 pcm。对于语音标记，格式为 JSON。

类型：字符串

有效值：`json | mp3 | ogg_opus | ogg_vorbis | pcm | mulaw | alaw`

必需：否

## OutputUri

输出语音文件的路径。

类型：字符串

必需：否

## RequestCharacters

合成的可计费字符数。

类型：整数

必需：否

## SampleRate

指定的音频频率，单位为 Hz。

对于 MP3 和 OGG Vorbis，有效值为“8000”、“16000”、“22050”和“24000”。标准语音的默认值为“22050”。神经语音的默认值为“24000”。长篇语音的默认值为“24000”。生成式语音的默认值为“24000”。

对于 PCM，有效值为“8000”和“16000”。默认值为“16000”。

ogg\_opus 的有效值为“48000”。

mu-law 和 a-law 的有效值为“8000”。

类型：字符串

必需：否

### SnsTopicArn

SNS 主题的 ARN，可用于为语音合成任务提供状态通知。

类型：字符串

模式：`^arn:aws(-(cn|iso(-b)?|us-gov))?:sns:[a-z0-9_-]{1,50}:\d{12}:[a-zA-Z0-9_-]{1,251}([a-zA-Z0-9_-]{0,5}|\.fifo)$`

必需：否

### SpeechMarkTypes

为输入文本返回的语音标记的类型。

类型：字符串数组

数组成员：最多 4 项。

有效值：`sentence | ssm1 | viseme | word`

必需：否

### TaskId

Amazon Polly 生成的语音合成任务的标识符。

类型：字符串

模式：`^[a-zA-Z0-9_-]{1,100}$`

必需：否

### TaskStatus

单个语音合成任务的当前状态。

类型：字符串

有效值：scheduled | inProgress | completed | failed

必需：否

### TaskStatusReason

特定语音合成任务当前状态的原因，包括任务失败时出现的错误。

类型：字符串

必需：否

### TextType

指定输入文本是纯文本还是 SSML。默认值为纯文本。

类型：字符串

有效值：ssml | text

必需：否

### Voiceld

要用于合成的语音 ID。

类型：字符串

有效值：Aditi | Amy | Astrid | Bianca | Brian | Camila | Carla | Carmen | Celine | Chantal | Conchita | Cristiano | Dora | Emma | Enrique | Ewa | Filiz | Gabrielle | Geraint | Giorgio | Gwyneth | Hans | Ines | Ivy | Jacek | Jan | Joanna | Joey | Justin | Karl | Kendra | Kevin | Kimberly | Lea | Liv | Lotte | Lucia | Lupe | Mads | Maja | Marlene | Mathieu | Matthew | Maxim | Mia | Miguel | Mizuki | Naja | Nicole | Olivia | Penelope | Raveena | Ricardo | Ruben | Russell | Salli | Seoyeon | Takumi | Tatyana | Vicki | Vitoria | Zeina | Zhiyu | Aria | Ayanda | Arlet | Hannah | Arthur | Daniel | Liam | Pedro | Kajal | Hiujin | Laura | Elin | Ida | Suvi | Ola | Hala | Andres | Sergio | Remi | Adriano | Thiago | Ruth | Stephen | Kazuha | Tomoko | Niamh | Sofie | Lisa | Isabelle | Zayd | Danielle | Gregory | Burcu | Jitka | Sabrina | Jasmine | Jihye | Ambre | Beatrice | Florian | Lennart | Lorenzo | Tiffany

必需：否

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## TextEvent

包含要合成语音的文本内容。

### 内容

#### Text

要合成的文本内容。如果指定 `ssml` 为 `TextType`，请按照 SSML 格式输入文本。

类型：字符串

是否必需：是

#### FlushStreamConfiguration

用于控制合成音频何时刷新到输出流的配置。

类型：[FlushStreamConfiguration](#) 对象

必需：否

#### TextType

指定输入文本是纯文本还是 SSML。默认：纯文本。

类型：字符串

有效值：`ssml` | `text`

必需：否

### 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

# ThrottlingReason

提供有关特定限制原因的信息。

## 内容

### reason

解释请求受限原因的原因代码。

类型：字符串

必需：否

### resource

导致限制的资源。

类型：字符串

必需：否

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## ValidationExceptionField

有关验证失败的字段的信息。

### 内容

#### message

一条消息，描述该字段验证失败的原因。

类型：字符串

是否必需：是

#### name

验证失败的字段的名称。

类型：字符串

必需：是

### 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## Voice

语音描述。

### 内容

#### AdditionalLanguageCodes

除默认语言外，指定语音还可用的其他语言代码。

例如，Aditi 的默认语言是印度英语 (en-IN)，因为它最初用于该语言。由于 Aditi 会说双语，并且精通印度英语和印地语，因而此参数将显示代码 hi-IN。

类型：字符串数组

有效值：arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS  
| en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT  
| ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO  
| ru-RU | sv-SE | tr-TR | en-NZ | en-ZA | ca-ES | de-AT | yue-CN | ar-AE  
| fi-FI | en-IE | nl-BE | fr-BE | cs-CZ | de-CH | en-SG

必需：否

#### Gender

语音的性别。

类型：字符串

有效值：Female | Male

必需：否

#### Id

Amazon Polly 分配的语音 ID。这是您调用 SynthesizeSpeech 操作时指定的 ID。

类型：字符串

有效值：Aditi | Amy | Astrid | Bianca | Brian | Camila | Carla | Carmen  
| Celine | Chantal | Conchita | Cristiano | Dora | Emma | Enrique | Ewa  
| Filiz | Gabrielle | Geraint | Giorgio | Gwyneth | Hans | Ines | Ivy |

Jacek | Jan | Joanna | Joey | Justin | Karl | Kendra | Kevin | Kimberly  
 | Lea | Liv | Lotte | Lucia | Lupe | Mads | Maja | Marlene | Mathieu  
 | Matthew | Maxim | Mia | Miguel | Mizuki | Naja | Nicole | Olivia  
 | Penelope | Raveena | Ricardo | Ruben | Russell | Salli | Seoyeon |  
 Takumi | Tatyana | Vicki | Vitoria | Zeina | Zhiyu | Aria | Ayanda |  
 Arlet | Hannah | Arthur | Daniel | Liam | Pedro | Kajal | Hiujin | Laura  
 | Elin | Ida | Suvi | Ola | Hala | Andres | Sergio | Remi | Adriano  
 | Thiago | Ruth | Stephen | Kazuha | Tomoko | Niamh | Sofie | Lisa |  
 Isabelle | Zayd | Danielle | Gregory | Burcu | Jitka | Sabrina | Jasmine  
 | Jihye | Ambre | Beatrice | Florian | Lennart | Lorenzo | Tiffany

必需：否

### LanguageCode

语音的语言代码。

类型：字符串

有效值：arb | cmn-CN | cy-GB | da-DK | de-DE | en-AU | en-GB | en-GB-WLS  
 | en-IN | en-US | es-ES | es-MX | es-US | fr-CA | fr-FR | is-IS | it-IT  
 | ja-JP | hi-IN | ko-KR | nb-NO | nl-NL | pl-PL | pt-BR | pt-PT | ro-RO  
 | ru-RU | sv-SE | tr-TR | en-NZ | en-ZA | ca-ES | de-AT | yue-CN | ar-AE  
 | fi-FI | en-IE | nl-BE | fr-BE | cs-CZ | de-CH | en-SG

必需：否

### LanguageName

语言的用户可读英文名称。

类型：字符串

必需：否

### Name

语音的名称（例如 Salli、Kendra 等）。提供了用户可读的语音名称，您可以在应用程序中显示该名称。

类型：字符串

必需：否

## SupportedEngines

指定给定语音支持哪些引擎 ( standard、neural、long-form 或 generative )。

类型：字符串数组

有效值：standard | neural | long-form | generative

必需：否

## 另请参阅

有关以特定语言之一使用此 API 的更多信息 AWS SDKs，请参阅以下内容：

- [AWS 适用于 C++ 的 SDK](#)
- [AWS 适用于 Java 的 SDK V2](#)
- [AWS 适用于 Ruby V3 的 SDK](#)

## 常见错误类型

本节列出了此 AWS 服务可能返回的常见错误类型。并非所有服务都会返回此处列出的所有错误类型。对于特定于此服务的 API 操作的错误，请参阅该 API 操作的主题。

### AccessDeniedException

您无权执行此操作。验证您的 IAM 策略是否包含所需的权限。

HTTP 状态代码：403

### ExpiredTokenException

请求中包含的安全令牌已过期。申请新的安全令牌并重试。

HTTP 状态代码：403

### IncompleteSignature

请求签名不符合 AWS 标准。确认您使用的是有效的 AWS 凭证，并且您的请求格式是否正确。如果您使用的是 SDK，请确保它是最新的。

HTTP 状态代码：403

## InternalFailure

由于内部服务器问题，该请求现在无法处理。请稍后重试。如果问题仍然存在，请联系 Supp AWS ort。

HTTP 状态代码：500

## MalformedHttpRequestException

无法处理请求正文。当无法使用指定的内容编码算法对请求正文进行解压缩时，通常会发生这种情况。验证内容编码标头是否与使用的压缩格式相匹配。

HTTP 状态代码：400

## NotAuthorized

您无权执行此操作。验证您的 IAM 策略是否包含所需的权限。

HTTP 状态代码：401

## OptInRequired

您的 AWS 账户需要订阅此服务。确认您已在账户中启用该服务。

HTTP 状态代码：403

## RequestAbortedException

在返回响应之前，请求已中止。这通常发生在客户端关闭连接时。

HTTP 状态代码：400

## RequestEntityTooLargeException

请求实体太大。请缩小请求正文的大小，然后重试。

HTTP 状态代码：413

## RequestTimeoutException

请求超时。服务器未在预期的时间范围内收到完整的请求。请重试。

HTTP 状态代码：408

## ServiceUnavailable

服务暂时不可用。请稍后重试。

HTTP 状态代码：503

#### ThrottlingException

您的请求率太高。收到此异常的 AWS SDKs 自动重试请求。请降低请求频率。

HTTP 状态代码：400

#### UnknownOperationException

无法识别该操作或操作。确认操作名称拼写正确，并且您使用的 API 版本是否支持该名称。

HTTP 状态代码：404

#### UnrecognizedClientException

您提供的 X.509 证书或 AWS 访问密钥 ID 不在我们的记录中。确认您使用的是有效的凭证，并且这些凭证尚未过期。

HTTP 状态代码：403

#### ValidationError

输入不符合要求的格式或限制。检查是否包含所有必需的参数以及值是否有效。

HTTP 状态代码：400

## 常见参数

以下列表包含所有操作用于使用查询字符串对 Signature Version 4 请求进行签名的参数。任何特定于操作的参数都列在该操作的主题中。有关签名版本 4 的更多信息，请参阅 IAM 用户指南中的[签署 AWS API 请求](#)。

#### X-Amz-Algorithm

您用于创建请求签名的哈希算法。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

有效值：AWS4-HMAC-SHA256

必需：条件

## X-Amz-Credential

凭证范围值，该值是一个字符串，其中包含您的访问密钥、日期、您要定位的区域、您请求的服务以及终止字符串（“aws4\_request”）。值采用以下格式表示：`access_key/YYYYMMDD/region/service/aws4_request`。

有关更多信息，请参阅 IAM 用户指南中的[创建已签名的 AWS API 请求](#)。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

必需：条件

## X-Amz-Date

用于创建签名的日期。格式必须为 ISO 8601 基本格式 (YYYYMMDD'T'HHMMSS'Z')。例如，以下日期时间是有效 X-Amz-Date 值：`20120325T120000Z`。

条件：X-Amz-Date 对于所有请求都是可选的；它可用于覆盖用于签署请求的日期。如果日期标题以 ISO 8601 基本格式指定，则 X-Amz-Date 不是必填项。使用 X-Amz-Date 时，它总是会覆盖 Date 标题的值。有关更多信息，请参阅 IAM 用户指南中的[AWS API 请求签名元素](#)。

类型：字符串

必需：条件

## X-Amz-Security-Token

通过调用 AWS Security Token Service (AWS STS) 获得的临时安全令牌。有关支持来自 AWS STS 的临时安全凭证的服务列表，请参阅《IAM 用户指南》中的[使用 IAM 的 AWS 服务](#)。

条件：如果您使用的是中的临时安全证书 AWS STS，则必须包含安全令牌。

类型：字符串

必需：条件

## X-Amz-Signature

指定从要签名的字符串和派生的签名密钥计算的十六进制编码签名。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

必需：条件

## X-Amz-SignedHeaders

指定作为规范请求的一部分包含的所有 HTTP 标头。有关指定签名标头的更多信息，请参阅 IAM 用户指南中的[创建已签名的 AWS API 请求](#)。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

必需：条件

# Amazon Polly 文件历史记录

下表描述此 Amazon Polly 开发人员指南每次发布时进行的重要修改。要获得本文档的更新通知，您可以订阅 RSS 源。

- 文档最新更新时间：2025 年 2 月 18 日

变更	说明	日期
<a href="#">双向直播在新地区推出</a>	Amazon Polly 双向直播现已在另外两个 AWS 地区推出：欧洲（伦敦）和加拿大（中部）。有关更多信息，请参阅 <a href="#">生成语音</a> 。	2026年4月20日
<a href="#">更新了托管策略</a>	Amazon Polly 更新了托管策略 AmazonPollyReadOnlyAccess，增加了双向流式语音合成的权限。有关信息，请参阅 <a href="#">Amazon Polly 对 AWS 托管策略的更新</a> 。	2026 年 3 月 19 日
<a href="#">为生成添加了新的声音 text-to-speech</a>	Amazon Polly 现在还提供了十种额外的生成声音：Brian、Aria、Jasmine、Tiffany、Ambre、Florian、Sabrina、Lennart、Beatrice、Lennart、Beatrice、Len 有关生成式 TTS 语音的列表，请参阅 <a href="#">生成式语音</a> 。	2026 年 3 月 19 日
<a href="#">为生成语音添加了新的区域</a>	Amazon Polly 生成语音现已在另外两个 AWS 地区推出：欧洲（伦敦）和加拿大（中部	2026 年 3 月 19 日

	<p>)。有关更多信息，请参阅<a href="#">生成语音</a>。</p>	
<a href="#">为生成语音添加了新的区域</a>	<p>Amazon Polly 生成语音现已在另外三个 AWS 地区推出：亚太地区（东京）、亚太地区（首尔）和亚太地区（新加坡）。有关更多信息，请参阅<a href="#">生成语音</a>。</p>	2025 年 11 月 18 日
<a href="#">为生成添加了新的声音 text-to-speech</a>	<p>Amazon Polly 现在提供了另外六种生成声音：Seoyeon、Camila、Hannah、Niamh、Laura 和 Lisa。有关生成式 TTS 语音的列表，请参阅<a href="#">生成式语音</a>。</p>	2025 年 11 月 14 日
<a href="#">为神经语音添加了新区域</a>	<p>Amazon Polly 现已在欧洲（苏黎世）AWS 地区上市。该区域支持神经 TTS (NTTS) 语音。有关更多信息，请参阅<a href="#">神经语音</a>。</p>	2025 年 10 月 20 日
<a href="#">为生成添加了新的声音 text-to-speech</a>	<p>Amazon Polly 现在提供了另七种生成式语音：Salli、Isabelle、Céline、Liam、Gabrielle、Ola 和 Ewa。有关生成式 TTS 语音的列表，请参阅<a href="#">生成式语音</a>。</p>	2025 年 8 月 26 日
<a href="#">为神经语音和标准语音添加了新区域</a>	<p>Amazon Polly 现已在亚太地区（马来西亚）AWS 区域推出。该区域支持神经 TTS (NTTS) 和标准语音。有关更多信息，请参阅<a href="#">神经语音</a>和<a href="#">标准语音</a>。</p>	2025 年 3 月 27 日

<a href="#">为神经添加了新的声音 text-to-speech</a>	Amazon Polly 现在提供了另一种韩语语音：Jihye。有关 NTTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2025 年 3 月 26 日
<a href="#">为神经语音和标准语音添加了新区域</a>	Amazon Polly 现已在欧洲（西班牙）AWS 区域提供。该区域支持神经 TTS（NTTTS）和标准语音。有关更多信息，请参阅 <a href="#">神经语音</a> 和 <a href="#">标准语音</a> 。	2025 年 2 月 18 日
<a href="#">为神经添加了新的声音 text-to-speech</a>	Amazon Polly 现在提供了另一种英语（新加坡）语音：Jasmine。有关 NTTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2025 年 2 月 11 日
<a href="#">为生成添加了新的声音 text-to-speech</a>	Amazon Polly 现在提供了七种新的生成式语音：Pedro、Andrés、Sergio、Daniel、Kajal、Rémi、Bianca。有关生成式 TTS 语音的列表，请参阅 <a href="#">生成式语音</a> 。	2024 年 11 月 21 日
<a href="#">添加了新的长篇语音</a>	Amazon Polly 现在提供了更长篇的语音。另外增加了一种英语（美国）和两种新的西班牙语（西班牙）长篇语音：Patrick、Alba 和 Raúl。有关所有长篇语音的列表，请参阅 <a href="#">长篇语音</a> 。	2024 年 11 月 14 日
<a href="#">为生成添加了新的声音 text-to-speech</a>	Amazon Polly 现在提供了六种新的生成式语音：Aynda、Léa、Lucia、Mía、Lupe 和 Vicki。有关生成式 TTS 语音的列表，请参阅 <a href="#">生成式语音</a> 。	2024 年 11 月 6 日

<a href="#">为生成添加了新的声音 text-to-speech</a>	Amazon Polly 现在提供了四种新的生成式语音：Olivia、Danielle、Joanna 和 Stephen。有关生成式 TTS 语音的列表，请参阅 <a href="#">生成式语音</a> 。	2024 年 10 月 10 日
<a href="#">添加了新的语言 text-to-speech</a>	Amazon Polly 现在提供了两种新的 TTS 语言：捷克语 (cs-CZ) 和德语 (瑞士) (de-CH)。有关语言的列表，请参阅 <a href="#">支持的语言</a> 。	2024 年 9 月 26 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在提供了两种新的 NTTS 语音：Jitka 和 Sabrina。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2024 年 8 月 27 日
<a href="#">添加了新的生成式语音引擎</a>	Amazon Polly 现在提供了专为长篇内容设计的生成式语音引擎，生成式变体中有三种英语语音：Amy、Matthew 和 Ruth。有关更多信息，请参阅 <a href="#">生成式语音</a> 。	2024 年 3 月 28 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在提供了 NTTS 土耳其语音 Burcu。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2024 年 2 月 14 日
<a href="#">添加了新的长篇语音引擎</a>	Amazon Polly 现在提供专为较长内容设计的长篇语音引擎，有三种 en-US 语音：Danielle、Gregory 和 Ruth。有关更多信息，请参阅 <a href="#">长篇语音</a> 。	2023 年 11 月 16 日

<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供两种新的 NTTS 美国英语语音：Danielle 和 Gregory。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2023 年 10 月 5 日
<a href="#">适用于 Windows 的 Amazon Polly</a>	将不再支持 Amazon Polly Windows 语音应用程序编程接口 (SAPI) 插件。	2023 年 9 月 26 日
<a href="#">更新了 Amazon Polly 的配额指南</a>	更新了 Amazon Polly 配额指南。添加了术语示例和说明。有关更新内容，请参阅 <a href="#">Amazon Polly 中的配额</a> 。	2023 年 8 月 17 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供海湾阿拉伯语 NTTS 语音，Zayd。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2023 年 8 月 16 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供比利时法语 NTTS 语音，Isabelle。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2023 年 8 月 1 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供比利时荷兰语（佛兰芒语）NTTS 语音，Lisa。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2023 年 6 月 7 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供两种新的 NTTS 语音：爱尔兰英语 (Niamh) 和丹麦语 (Sofie)。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2023 年 5 月 30 日

<a href="#">更新了 Amazon Polly 的 IAM 指南</a>	更新了指南，使其符合 IAM 最佳实践。有关更多信息，请参阅 <a href="#">IAM 安全最佳实践</a> 。	2023 年 4 月 19 日
<a href="#">WordPress update</a>	将不再支持 Amazon Polly WordPress 插件。	2023 年 4 月 6 日
<a href="#">添加了新区域</a>	Amazon Polly 现已在亚太地区（大阪）AWS 区域推出。该区域支持神经 TTS (NTTS)。有关更多信息，请参阅 <a href="#">功能和区域兼容性</a> 以获取支持 NTTS 的区域列表。	2023 年 4 月 5 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供两种新的日语 NTTS 语音：Kazuha 和 Tomoko。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2023 年 2 月 7 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供两种新的美国英语 NTTS 语音：Stephen 和 Ruth。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2023 年 1 月 31 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可为巴西葡萄牙语 (Thiago)、卡斯蒂利亚西班牙语 (Sergio)、法语 (Rémi)、意大利语 (Adriano) 和墨西哥西班牙语 (Andrés) 提供新的 NTTS 语音。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2023 年 1 月 24 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可为阿拉伯语 (Hala) 和波兰语 (Ola) 提供 NTTS 语音。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 11 月 17 日

<a href="#">发布 AWS PrivateLink 支持</a>	Amazon Polly 现在提供 AWS PrivateLink 支持。要了解更多信息，请参阅 <a href="#">将 Amazon Polly 与 VPC 终端节点一起使用</a> 。	2022 年 11 月 9 日
<a href="#">为 NTTS 添加了新语音和语言</a>	Amazon Polly 现在可为芬兰语 (Suvi)、挪威语 (Ida) 和瑞典语 (Elin) 提供 NTTS 语音。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 11 月 8 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供荷兰语 NTTS 语音，Laura。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 11 月 2 日
<a href="#">添加了新区域</a>	Amazon Polly 现已在欧洲地区（巴黎）AWS 区域中提供。该区域支持神经 TTS (NTTS)。有关更多信息，请参阅 <a href="#">功能和区域兼容性</a> 以获取支持 NTTS 的区域列表。	2022 年 9 月 22 日
<a href="#">为 NTTS 添加了新语音和语言</a>	Amazon Polly 现在可提供粤语 NTTS 语音，Hiujin。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 9 月 20 日
<a href="#">添加了新区域</a>	Amazon Polly 现已在亚太地区（孟买）AWS 区域中提供。该区域支持神经 TTS (NTTS)。有关更多信息，请参阅 <a href="#">功能和区域兼容性</a> 以获取支持 NTTS 的区域列表。	2022 年 9 月 1 日

<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供普通话语音 Zhiyu 作为 NTTS 语音。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 8 月 23 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供印地语 NTTS 语音，Kajal。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 7 月 27 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可为美国西班牙语 (Pedro)、德语 (Daniel)、加拿大法语 (Liam) 和英国英语 (Arthur) 提供 NTTS 语音。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 6 月 28 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供葡萄牙语 (巴西) 语音 Vitória 作为 NTTS 语音。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 4 月 27 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供葡萄牙语 (欧洲) 语音 Inès 作为 NTTS 语音。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 4 月 26 日
<a href="#">为 NTTS 添加了新语音和语言</a>	Amazon Polly 现在可提供德语 (奥地利) 语言和 NTTS 语音 Hannah。有关 NTTS 语音的列表，请参阅 <a href="#">神经语音</a> 。	2022 年 4 月 19 日

<a href="#">为 NTTS 添加了新语音和语言</a>	Amazon Polly 现在可提供西班牙语 (墨西哥) 语音 Mia 作为 NTTS 语音。添加了一种新的语言, 即加泰罗尼亚语, 以及 NTTS 语音 Arlet。有关 NTTS 语音的列表, 请参阅 <a href="#">神经语音</a> 。	2022 年 3 月 22 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供日语语音 Takumi 作为 NTTS 语音。有关 NTTS 语音的列表, 请参阅 <a href="#">神经语音</a> 。	2021 年 12 月 6 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供法语语音 Léa 作为 NTTS 语音。有关 NTTS 语音的列表, 请参阅 <a href="#">神经语音</a> 。	2021 年 11 月 18 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供意大利语语音 Bianca 和欧洲西班牙语语音 Lucia 作为 NTTS 语音。有关 NTTS 语音的列表, 请参阅 <a href="#">神经语音</a> 。	2021 年 11 月 8 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供一种新的南非英语语音, Ayanda。该语音仅可用作 NTTS 语音。有关 NTTS 语音的列表, 请参阅 <a href="#">神经语音</a> 。	2021 年 9 月 1 日
<a href="#">添加了新区域</a>	Amazon Polly 现已在非洲 (开普敦) AWS 区域中推出。该区域支持神经 TTS (NTTS)。有关更多信息, 请参阅 <a href="#">功能和区域兼容性</a> 以获取支持 NTTS 的区域列表。	2021 年 9 月 1 日

<a href="#">添加了新语言和语音</a>	Amazon Polly 现在可支持新西兰英语 (en-NZ)。新的 NTTS 语音 Aria 会说新西兰英语和精选的毛利语单词。	2021 年 8 月 24 日
<a href="#">新特征</a>	Amazon Polly 将对话式说话风格作为神经语音 Matthew 和 Joanna 的默认版本。我们删除了对于对话式说话风格的引用。	2021 年 6 月 28 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供德语语音 Vicki 作为 NTTS 语音。	2021 年 6 月 15 日
<a href="#">添加了新语音</a>	向法语 ( 加拿大 ) (fr-CA) 区域中填加了新的女性语音, Gabrielle。此语音的质量高, 并且只能用作 NTTS 语音。像所有神经语音一样, 它仅在某些区域可用。有关区域的列表, 请参阅 <a href="#">功能和区域兼容性</a> 。	2021 年 6 月 1 日
<a href="#">为 NTTS 添加了新语音</a>	Amazon Polly 现在可提供韩语语音 Seoyeon 作为 NTTS 语音。	2021 年 5 月 11 日
<a href="#">为 NTTS 添加了新区域</a>	Amazon Polly 现在支持加拿大 ( 中部 ) 地区的神经 TTS (NTTS)。AWS 有关更多信息, 请参阅 NTTS 的 <a href="#">功能和区域兼容性</a> 。	2021 年 3 月 17 日

## [适用于新闻播音员风格的新语音](#)

除了新闻播音员风格的 Matthew、Joanna 和 Lupe 语音之外，Amazon Polly 现在为这种说话风格提供了一个额外的选项。通过神经引擎，您可以将英式英语语音 Amy 用于新闻播音员风格。有关更多信息，请参阅 [NTTS 说话样式](#)。

2020 年 11 月 10 日

## [为 NTTS 添加了新区域](#)

除了 NTTS 的现有区域 ( us-east-1、us-west-2、eu-west-1 和 ap-southeast-2 ) 之外，现在还增加了另外四个受支持区域：ap-northeast-1 ( 东京 )、ap-southeast-1 ( 新加坡 )、eu-central-1 ( 法兰克福 ) 和 eu-west-2 ( 伦敦 )。有关更多信息，请参阅 NTTS 的 [功能和区域兼容性](#)。

2020 年 9 月 3 日

## [添加了新语音](#)

除了儿童语音 Ivy 和 Justin 之外，还向美国英语 (en-US) 版中添加了一种新的男童语音 Kevin。此新语音的质量非常高，并且只能用作 NTTS 语音。与所有神经语音一样，此新语音仅在四个区域内受支持：us-east-1 ( 弗吉尼亚北部 )、us-west-2 ( 俄勒冈 )、eu-west-1 ( 爱尔兰 ) 和 ap-southeast-2 ( 悉尼 )。有关更多信息，请参阅 [NTTS 语音](#)。

2020 年 6 月 16 日

## [适用于新闻播音员风格的新语音](#)

除了新闻播音员风格的 Matthew 和 Joanna 语音之外，Amazon Polly 现在为这种说话风格提供了一个额外的选项。通过神经引擎，您可以将西班牙语（美国）版的 Lupe 语音用于新闻播音员风格。有关更多信息，请参阅 [NTTS 说话样式](#)。

2020 年 4 月 16 日

## [新特征](#)

除了新闻播音员风格之外，Amazon Polly 现在还提供了第二种 NTTS 说话风格，可以帮助您将更好的文本合成到语音段落中。对话式风格使用神经系统，以更加友好和富有表现力的对话式风格制作语音，这种风格可以在许多使用案例中使用。有关更多信息，请参阅 [NTTS 说话样式](#)。

2019 年 11 月 25 日

## [添加了新语音](#)

增加了两个新语音：  
Camila（女，葡萄牙语-巴西）  
和 Lupe（女，西班牙语-美国）。

2019 年 10 月 23 日

## [添加了新功能](#)

添加了[适用于 Windows 的 Amazon Polly 插件](#)，以便将全套 Amazon Polly 语音并入 Windows SAPI 兼容应用程序。

2019 年 9 月 26 日

<a href="#">主要新功能</a>	除了自推出以来由Amazon Polly支持的标准 text-to-speech ( TTS ) 语音外，Amazon Polly现在还提供了改进的神经TTS ( NTTS ) 系统，该系统可以提供更高质量的声音，从而为您提供最自然、最像人类的声音。text-to-speech有关更多信息，请参阅 <a href="#">神经 Text-to-Speech</a> 。	2019 年 7 月 30 日
<a href="#">添加了新语音</a>	添加了新声音：Lucia ( 女声，西班牙语 ) 和 Bianca ( 女声，意大利语 )。	2018 年 8 月 2 日
<a href="#">添加了新语言</a>	添加了新语言：墨西哥西班牙语 (es-MX)。此语言使用 Mia 的女声。	2018 年 8 月 2 日
<a href="#">添加了新语言</a>	增加了新语言：印地语 (hi-IN)。此语音使用 Aditi 的女性语音，它还用于印度英语，这使 Aditi 成为 Amazon Polly 的第一个双语语音。	2018 年 8 月 2 日
<a href="#">添加了新功能</a>	增加了 <a href="#">长文本段的语音合成</a> ( 最多 100,000 个计费字符 )。	2018 年 7 月 17 日
<a href="#">添加了新的 SSML 功能</a>	增加了 <a href="#">合成语音的最长持续时间</a> 。	2018 年 7 月 17 日
<a href="#">添加了新语音</a>	增加了新语音：Léa ( 女性，法语 )。	2018 年 6 月 5 日
<a href="#">区域扩展</a>	Amazon Polly 到所有商业区域的扩展。	2018 年 4 月 6 日

---

<a href="#">添加了新语言</a>	增加了新语言：韩语 (ko-KR)。	2018 年 4 月 6 日
<a href="#">扩展了功能</a>	Amazon Polly WordPress 插件功能，包括新增的 Amazon Translate 功能。	2018 年 4 月 6 日
<a href="#">添加了新语音</a>	增加了两种新语音：Aditi (印度英语女声) 和 Seoyeon (韩语女声)。	2017 年 11 月 15 日
<a href="#">新特征</a>	增加了新的 <a href="#">语音标记</a> 功能，并扩展了 <a href="#">SSML</a> 功能。	2017 年 4 月 19 日
<a href="#">新指南</a>	本指南是 Amazon Polly 开发人员指南的第一个版本。	2016 年 11 月 30 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。