aws

开发人员指南

Amazon Transcribe

Amazon Transcribe: 开发人员指南

Table of Contents

什么是 Amazon Transcribe？

Amazon Transcribe是一种自动语音识别服务，它使用机器学习模型将音频转换为文本。您可以用Amazon Transcribe作独立的转录服务，也可以向任何应用程序添加speech-to-text功能。

借Amazon Transcribe助，您可以通过自定义语言来提高特定用例的准确性，筛选内容以确保客户隐私或适合受众的语言，分析多声道音频中的内容，对单个发言者的语音进行分区等。

您可以实时转录媒体（流式传输），也可以转录Amazon S3存储桶中的媒体文件（批处理）。要查看每种转录类型支持哪些语言，请参阅下支持的语言和特定语言的特征表。

主题

- Amazon Transcribe和 HIPAA 资格
- 定价
- 区域可用性和配额

看看是什么Amazon Transcribe？ 观看此服务的简短视频导览。

要了解更多信息，请参阅Amazon Transcribe 的工作原理和Amazon Transcribe 入门。

> **Tip**
>
> 有关 Amazon TranscribeAPI 的信息位于 API 参考中。

Amazon Transcribe和 HIPAA 资格

Amazon Transcribe受到 HIPAA 资格和 BAA AWS 的保障，后者要求 BAA 客户在使用时对所有静态和传输中的 PHI 进行加密。在所有Amazon Transcribe运营地区均可免费使用自动 PHI 识别。有关更多信息，请参阅 HIPAA 资格和 BAA。

定价

Amazon Transcribe是一项pay-as-you-go服务；定价基于转录的音频的秒数，按月计费。

使用量以一秒的增量进行计费，时长最少 15 秒。请注意，诸如 PII 内容编辑和自定义语言模型之类的功能需要支付额外费用。

有关每项的成本信息AWS 区域，请参阅定Amazon Transcribe价。

区域可用性和配额

Amazon Transcribe支持以下内容AWS 区域：

区域	转录类型
a-south----------1	批量和流批量的
ap-east-east-1（香港）	批处理
ap-northeast-1 (东京)	批量和流批量的
ap-northeast-2 (首尔)	批量和流批量的
ap-south----1（孟买）	批量和流批量的
ap-southeast-1 (新加坡)	批量和流批量的
ap-southeast-2 (悉尼)	批量和流批量的
central-sentral-1（加拿大，中部）（加拿大	批量和流批量的
eu-central-1 (法兰克福)	批量和流批量的
eu-north-1 (斯德哥尔摩	批处理
eu-west-1 (爱尔兰)	批量和流批量的
eu-weu-we-we-we-s	批量和流批量的
eu-weu-weu-we-we-	批处理
me-south-----1	批处理
sa-east-1（圣保罗）（圣保罗）	批量和流批量的
us-east-east-s-east-1	批量和流批量的
us-east-east-east-2	批量和流批量的

区域	转录类型
us-gov-east-1（GovCloud美国东部）	批量和流批量的
us-gov-west-1（GovCloud美国西部）	批量和流批量的
us-west-west-west-s-w	批处理
us-west-2 (俄勒冈)	批量和流批量的

> **Important**
>
> Amazon Transcribe、Amazon Transcribe Medical和呼叫分析的区域支持有所不同。

要获取每个支持区域的终端节点，请参阅AWS一般参考中的服务终端节点。

有关与您的转录相关的配额列表，请参阅AWS一般参考中的服务配额。某些配额可以根据要求更改。如果 “可调整” 列包含 “是”，则可以请求增加。为此，请选择提供的链接。

Amazon Transcribe features

为了帮助您确定哪种 Amazon Transcribe 解决方案最适合您的用例，下表提供了功能对比。

请注意，“批量转录”和 'post-call' 指转录位于 Amazon S3 存储桶中的文件，“流式转录”和 'real-time' 指实时转录媒体。

功能	Amazon Transcribe	Amazon Transcribe Medical[1]	Amazon Transcribe 呼叫分析
配置选项			
备选转录	批量转录，流式转录	批量转录，流式转录	否
声道识别	批量转录，流式转录	批量转录，流式转录	post-call, real-time
作业队列	批处理	否	post-call
语言识别	批量转录，流式转录	否	post-call
多语言识别	批量转录，流式转录	否	否
发言者分类	批量转录，流式转录	批量转录，流式转录	post-call
转录数字[2]	批量转录，流式转录	批量转录，流式转录	post-call, real-time
对话分析			
通话特点	否	否	post-call
通话摘要[2]	否	否	post-call
自定义分类	否	否	post-call
实时类别事件	否	否	real-time
实时问题检测[2]	否	否	real-time
实时发言者情绪	否	否	real-time
发言者情绪	否	否	post-call

功能	Amazon Transcribe	Amazon Transcribe Medical[1]	Amazon Transcribe 呼叫分析
语言自定义			
自定义语言模型[2]	批量转录，流式转录	否	post-call, real-time
自定义词汇	批量转录，流式转录	批量转录，流式转录	post-call, real-time
资源组织			
标记	批处理	批处理	post-call
敏感数据			
识别个人健康信息[2]	否	批量转录，流式转录	否
识别个人身份信息[2]	流式处理	否	real-time
编辑音频[2]	否	否	post-call, real-time
编辑转录[2]	批量转录，流式转录	否	post-call, real-time
词汇表筛选	批量转录，流式转录	否	post-call, real-time
视频			
字幕	批处理	否	否

[1] Amazon Transcribe 医疗仅提供美式英语版本。
[2] 此特征并非适用于所有语言；请查看支持的语言和特定语言的特征表格以了解更多详细信息。

支持的语言和特定语言的特征

除了 Amazon Transcribe 支持的语言，下表还列出了特定语言的特征。在继续转录之前，请确认您的媒体语言支持您要使用的特征。

要查看 Amazon Transcribe 特征的完整列表，请参阅特征摘要。

在下表中，“批量转录”是指转录位于 Amazon S3 存储桶中的媒体文件，“流式转录”是指实时转录流媒体。对于通话分析转录，'post-call' 是指转录位于 Amazon S3 存储桶中的媒体文件，'real-time' 指的是实时转录流媒体。

语言	语言代码	数据输入	转录数字	首字母缩略词	自定义语言模型	编辑	通话分析*
阿布哈兹语	ab-GE	批处理	否	批处理	否	否	否
南非荷兰语	af-ZA	批处理	否	批处理	否	否	否
阿拉伯语，海湾阿拉伯语	ar-AE	批处理	否	否	否	否	post-call
阿拉伯语，现代标准	ar-SA	批处理	否	否	否	否	否
亚美尼亚语	hy-AM	批处理	否	批处理	否	否	否
阿斯图里亚斯语	ast-ES	批处理	否	批处理	否	否	否
阿塞拜疆语	az-AZ	批处理	否	批处理	否	否	否
巴什基尔语	ba-RU	批处理	否	批处理	否	否	否

语言	语言代码	数据输入	转录数字	首字母缩略词	自定义语言模型	编辑	通话分析 *
巴斯克语	eu-ES	批处理	否	批处理	否	否	否
白俄罗斯语	be-BY	批处理	否	批处理	否	否	否
孟加拉语	bn-IN	批处理	否	批处理	否	否	否
波斯尼亚语	bs-BA	批处理	否	批处理	否	否	否
保加利亚语	bg-BG	批处理	否	批处理	否	否	否
加泰罗尼亚语	ca-ES	批处理	否	批处理	否	否	否
中库尔德语，伊朗	ckb-IR	批处理	否	批处理	否	否	否
中库尔德语，伊拉克	ckb-IQ	批处理	否	批处理	否	否	否
简体中文	zh-CN	批量转录，流式转录	否	否	否	否	post-call
繁体中文	zh-TW	批处理	否	否	否	否	否
克罗地亚语	hr-HR	批处理	否	批处理	否	否	否
捷克语	cs-CZ	批处理	否	批处理	否	否	否
丹麦语	da-DK	批处理	否	批处理	否	否	否
荷兰语	nl-NL	批处理	否	批处理	否	否	否

语言	语言代码	数据输入	转录数字	首字母缩略词	自定义语言模型	编辑	通话分析 *
英语，澳大利亚英语	en-AU	批量转录，流式转录	批量转录，流式转录	批量转录，流式转录	批量转录，流式转录	流式处理	post-call, real-time
英语，英国英语	en-GB	批量转录，流式转录	批量转录，流式转录	批量转录，流式转录	批量转录，流式转录	流式处理	post-call, real-time
英语，印度英语	en-IN	批处理	批处理	批处理	否	否	post-call
英语，爱尔兰英语	en-IE	批处理	批处理	批处理	否	否	post-call
英语，新西兰英语	en-NZ	批处理	批处理	批处理	否	否	否
英语，苏格兰英语	en-AB	批处理	批处理	批处理	否	否	post-call
英语，南非英语	en-ZA	批处理	批处理	批处理	否	否	否
英语，美国英语	en-US	批量转录，流式转录	批量转录，流式转录	批量转录，流式转录	批量转录，流式转录	批量转录，流式转录	post-call, real-time
英语，威尔士英语	en-WL	批处理	批处理	批处理	否	否	post-call
爱沙尼亚语	et-ET	批处理	否	批处理	否	否	否
波斯语	fa-IR	批处理	否	否	否	否	否
芬兰语	fi-FI	批处理	否	批处理	否	否	否

语言	语言代码	数据输入	转录数字	首字母缩略词	自定义语言模型	编辑	通话分析 *
法语	fr-FR	批量转录，流式转录	否	批量转录，流式转录	否	否	post-call, real-time
法语，加拿大法语	fr-CA	批量转录，流式转录	否	批量转录，流式转录	否	否	post-call, real-time
加利西亚语	gl-ES	批处理	否	批处理	否	否	否
格鲁吉亚语	ka-GE	批处理	否	批处理	否	否	否
德语	de-DE	批量转录，流式转录	批量转录，流式转录	批量转录，流式转录	批量转录，流式转录	否	post-call, real-time
德语，瑞士德语	de-CH	批处理	批处理	批处理	否	否	post-call
希腊语	el-GR	批处理	否	批处理	否	否	否
古吉拉特语	gu-IN	批处理	否	批处理	否	否	否
豪萨语	ha-NG	批处理	否	批处理	否	否	否
希伯来语	he-IL	批处理	否	否	否	否	否
印地语，印度英语	hi-IN	批量转录，流式转录	否	批量转录，流式转录	批处理	否	post-call
匈牙利语	hu-HU	批处理	否	批处理	否	否	否
冰岛语	is-IS	批处理	否	批处理	否	否	否

语言	语言代码	数据输入	转录数字	首字母缩略词	自定义语言模型	编辑	通话分析 *
印度尼西亚语	id-ID	批处理	否	批处理	否	否	否
意大利语	it-IT	批量转录，流式转录	否	批量转录，流式转录	否	否	post-call, real-time
日语	ja-JP	批量转录，流式转录	否	否	批量转录，流式转录	否	post-call
卡布列语	kab-DZ	批处理	否	批处理	否	否	否
卡纳达语	kn-IN	批处理	否	批处理	否	否	否
哈萨克语	kk-KZ	批处理	否	批处理	否	否	否
基尼亚卢旺达语	rw-RW	批处理	否	批处理	否	否	否
韩语	ko-KR	批量转录，流式转录	否	否	否	否	post-call
吉尔吉斯语	ky-KG	批处理	否	批处理	否	否	否
拉脱维亚语	lv-LV	批处理	否	批处理	否	否	否
立陶宛语	lt-LT	批处理	否	批处理	否	否	否
卢干达语	lg-IN	批处理	否	批处理	否	否	否
马其顿语	mk-MK	批处理	否	批处理	否	否	否
马来语	ms-MY	批处理	否	批处理	否	否	否

语言	语言代码	数据输入	转录数字	首字母缩略词	自定义语言模型	编辑	通话分析 *
马拉雅拉姆语	ml-IN	批处理	否	批处理	否	否	否
马耳他语	mt-MT	批处理	否	批处理	否	否	否
马拉地语	mr-IN	批处理	否	批处理	否	否	否
草甸马里语	mhr-RU	批处理	否	批处理	否	否	否
蒙古语	mn-MN	批处理	否	批处理	否	否	否
挪威布克莫尔语	no-NO	批处理	否	批处理	否	否	否
奥迪亚/奥里亚语	or-IN	批处理	否	批处理	否	否	否
普什图语	ps-AF	批处理	否	批处理	否	否	否
波兰语	pl-PL	批处理	否	批处理	否	否	否
葡萄牙语	pt-PT	批处理	否	批处理	否	否	post-call
葡萄牙语，巴西葡萄牙语	pt-BR	批量转录，流式转录	否	批量转录，流式转录	否	否	post-call, real-time
旁遮普语	pa-IN	批处理	否	批处理	否	否	否
罗马尼亚语	ro-RO	批处理	否	批处理	否	否	否
俄语	ru-RU	批处理	否	否	否	否	否
塞尔维亚语	sr-RS	批处理	否	批处理	否	否	否

语言	语言代码	数据输入	转录数字	首字母缩略词	自定义语言模型	编辑	通话分析 *
僧伽罗语	si-LK	批处理	否	批处理	否	否	否
斯洛伐克语	sk-SK	批处理	否	批处理	否	否	否
斯洛文尼亚语	sl-SI	批处理	否	批处理	否	否	否
索马里语	so-SO	批处理	否	批处理	否	否	否
西班牙语	es-ES	批处理	否	批处理	否	否	post-call
西班牙语，美国英语	es-US	批量转录，流式转录	否	批量转录，流式转录	批量转录，流式转录	否	post-call, real-time
巽他语	su-ID	批处理	否	批处理	否	否	否
斯瓦西里语，肯尼亚	sw-KE	批处理	否	批处理	否	否	否
斯瓦西里语，布隆迪	sw-BI	批处理	否	批处理	否	否	否
斯瓦西里语，卢旺达	sw-RW	批处理	否	批处理	否	否	否
斯瓦西里语，坦桑尼亚	sw-TZ	批处理	否	批处理	否	否	否

语言	语言代码	数据输入	转录数字	首字母缩略词	自定义语言模型	编辑	通话分析 *
斯瓦西里语，乌干达	sw-UG	批处理	否	批处理	否	否	否
瑞典语	sv-SE	批处理	否	批处理	否	否	否
塔加洛语/菲律宾语	tl-PH	批处理	否	批处理	否	否	否
泰米尔语	ta-IN	批处理	否	否	否	否	否
鞑靼语	tt-RU	批处理	否	批处理	否	否	否
泰卢固语	te-IN	批处理	否	否	否	否	否
泰语	th-TH	批量转录，流式转录	否	批量转录，流式转录	否	否	否
土耳其语	tr-TR	批处理	否	批处理	否	否	否
乌克兰语	uk-UA	批处理	否	批处理	否	否	否
维吾尔族	ug-CN	批处理	否	批处理	否	否	否
乌兹别克斯坦语	uz-UZ	批处理	否	批处理	否	否	否
越南语	vi-VN	批处理	否	批处理	否	否	否
威尔士语	cy-WL	批处理	否	批处理	否	否	否
沃洛夫语	wo-SN	批处理	否	批处理	否	否	否
祖鲁语	zu-ZA	批处理	否	批处理	否	否	否

*仅部分英语方言支持以下通话分析见解：

- 通话摘要：en-*（所有英语方言）
- 问题检测：en-AU、en-GB、en-US

支持的编程语言

Amazon Transcribe 支持以下 AWS SDK：

批量转录	流式转录
.NET	流式转录不支持 .NET。
AWS 命令行界面 (CLI)	流式转录不支持 CLI。
C++	C++
Go	Go
Java V2	Java V2
JavaScript	JavaScript V3
PHP V3	PHP V3
Python Boto3	适用于 Amazon Transcribe 的 Python 流式转录 SDK
Ruby V3	Ruby V3
Rust	Rust

有关将 SDK 与 Amazon Transcribe 配合使用的信息，请参阅使用 AWS SDK 进行转录 。

有关所有可用的 AWS SDK 和构建器工具的更多信息，请参阅在 AWS 上构建所需的工具。

> **Tip**
>
> 您可以在以下 GitHub 存储库中找到 SDK 代码示例：
>
> - AWS 代码示例

- Amazon Transcribe 示例

自定义词汇表和词汇表过滤器的字符集

对于 Amazon Transcribe 支持的每种语言，Amazon Transcribe 都有一组特定的字符可以识别。创建自定义词汇表或词汇表过滤器时，请仅使用您的语言字符集中所列的字符。如果您使用不支持的字符，则您的自定义词汇表或词汇表过滤器将失败。

> **Important**
>
> 请务必检查您的自定义词汇表文件是否仅使用以下字符集中所列的支持的 Unicode 代码点和代码点序列。

许多 Unicode 字符即使使用不同的代码点，在常用字体中也可能看起来完全相同。仅支持本指南中所列的代码点。例如，法语单词 déjà 可以使用预先组合的字符（其中一个 Unicode 值表示重音字符）或分解字符（其中两个 Unicode 值代表重音字符，一个值代表基本字符，另一个值代表重音字符）来呈现。

- 预先组合版本：0064 **00E9** 006A **00E0**（呈现为 déjà）
- 分解版本：0064 **0065 0301** 006A **0061 0300**（呈现为 déjà）

主题

- 波斯尼亚语字符集
- 保加利亚语字符集
- 加泰罗尼亚语字符集
- 中库尔德语字符集
- 中文、普通话（中国大陆）、简体字符集
- 中文、普通话（台湾）、繁体字符集
- 克罗地亚语字符集
- 捷克语字符集
- 丹麦语字符集
- 荷兰语字符集
- 英语字符集
- 爱沙尼亚语字符集
- 波斯语字符集
- 芬兰语字符集
- 法语字符集
- 加利西亚语字符集
- 格鲁吉亚语字符集
- 德语字符集
- 希腊语字符集
- 古吉拉特语字符集
- 豪萨语字符集
- 希伯来语字符集
- 印地语字符集
- 匈牙利语字符集
- 冰岛语字符集
- 印度尼西亚语字符集
- 意大利语字符集
- 日文字符集
- 卡布列语字符集
- 卡纳拉语字符集

- 哈萨克语字符集
- 基尼亚卢旺达语字符集
- 韩语字符集
- 吉尔吉斯语字符集
- 拉脱维亚语字符集
- 立陶宛语字符集
- 卢干达语字符集
- 马其顿语字符集
- 马来语字符集
- 马拉雅拉姆语字符集
- 马耳他语字符集
- 马拉地语字符集
- 草甸马里语字符集
- 蒙古语字符集
- 挪威布克莫尔语字符集
- 奥迪亚/奥里亚语字符集
- 普什图语字符集
- 波兰语字符集
- 葡萄牙语字符集
- 旁遮普语字符集
- 罗马尼亚语字符集
- 俄语字符集
- 塞尔维亚语字符集
- 僧伽罗语字符集
- 斯洛伐克语字符集
- 斯洛文尼亚语字符集
- 索马里语字符集
- 西班牙语字符集
- 巽他语字符集
- 斯瓦西里语字符集

- 瑞典语字符集
- 塔加洛语/菲律宾语字符集
- 泰米尔语字符集
- 鞑靼语字符集
- 泰卢固语字符集
- 泰语字符集
- 土耳其语字符集
- 乌克兰语字符集
- 维吾尔语字符集
- 乌兹别克语字符集
- 越南语字符集
- 威尔士语字符集
- 沃洛夫语字符集
- 祖鲁语字符集

阿布哈兹语字符集

对于阿布哈兹语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C

字符	代码	字符	代码
д	0434	ҥ	045D
е	0435	ў	045E
ж	0436	ҧ	045F
з	0437	ґ	0491
и	0438	ғ	0493
й	0439	җ	0497
к	043A	ҙ	0499
л	043B	қ	049B
м	043C	ҝ	049F
н	043D	ҡ	04A1
о	043E	ң	04A3
п	043F	ҥ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB
т	0442	ҭ	04AD
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7

字符	代码	字符	代码
ш	0448	h	04BB
щ	0449	ҽ	04BD
ъ	044A	ҿ	04BF
ы	044B	#	04CA
ь	044C	ӑ	04D1
э	044D	ӓ	04D3
ю	044E	ӗ	04D7
я	044F	ә	04D9
#	0450	ӡ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
s	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	#	04F7
ј	0458	ӹ	04F9
#	0525		

南非荷兰语字符集

对于南非荷兰语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
á	00E1	ï	00EF
è	00E8	ó	00F3
é	00E9	ô	00F4
ê	00EA	ö	00F6
ë	00EB	ú	00FA
í	00ED	û	00FB
î	00EE	ü	00FC

阿拉伯语字符集

对于阿拉伯语自定义词汇表，您可以在 Phrase 字段中使用以下 Unicode 字符。您也可以使用连字符 (-) 来分隔单词。

字符	代码	字符	代码
ء	0621	س	0633
آ	0622	ش	0634
أ	0623	ص	0635
ؤ	0624	ض	0636
إ	0625	ط	0637

字符	代码	字符	代码
ئ	0626	ظ	0638
ا	0627	ع	0639
ب	0628	غ	063A
ة	0629	ف	0641
ت	062A	ق	0642
ث	062B	ك	0643
ج	062C	ل	0644
ح	062D	م	0645
خ	062E	ن	0646
د	062F	ه	0647
ذ	0630	و	0648
ر	0631	ى	0649
ز	0632	ي	064A

阿斯图里亚斯语字符集

对于阿斯图里亚斯语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
á	00E1	ñ	00F1
é	00E9	ó	00F3
í	00ED	ú	00FA
ü	00FC		

阿塞拜疆语字符集

对于阿塞拜疆语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ä	00E4	ğ	011F
ç	00E7	ı	0131
ö	00F6	ş	015F
ü	00FC	ə	0259
◌̇	0307		

亚美尼亚语字符集

对于亚美尼亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z

- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ա	0561	մ	0574
բ	0562	յ	0575
գ	0563	ն	0576
դ	0564	շ	0577
ե	0565	ո	0578
զ	0566	չ	0579
է	0567	պ	057A
ը	0568	ջ	057B
թ	0569	ռ	057C
ժ	056A	ս	057D
ի	056B	վ	057E
լ	056C	տ	057F
խ	056D	ր	0580
ծ	056E	ց	0581
կ	056F	ւ	0582
հ	0570	փ	0583
ձ	0571	ք	0584

字符	代码	字符	代码
ɳ	0572	ɵ	0585
ɦ	0573	ɸ	0586

巴什基尔语字符集

对于巴什基尔语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C
д	0434	#	045D
е	0435	ў	045E
ж	0436	џ	045F
з	0437	ґ	0491
и	0438	ғ	0493
й	0439	ҷ	0497
к	043A	ҙ	0499

字符	代码	字符	代码
л	043B	қ	049B
м	043C	ҝ	049F
н	043D	ҡ	04A1
о	043E	ң	04A3
п	043F	ҥ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB
т	0442	ҭ	04AD
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7
ш	0448	h	04BB
щ	0449	ҽ	04BD
ъ	044A	ҿ	04BF
ы	044B	#	04CA
ь	044C	ӑ	04D1
э	044D	ӓ	04D3
ю	044E	ӗ	04D7

字符	代码	字符	代码
я	044F	ҙ	04D9
ѐ	0450	ҙ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
ѕ	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	ӵ	04F7
ј	0458	ӹ	04F9

巴斯克语字符集

对于巴斯克语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
á	00E1	ñ	00F1
é	00E9	ó	00F3
í	00ED	ú	00FA

字符	代码	字符	代码
ü	00FC		

白俄罗斯语字符集

对于白俄罗斯语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	с	0441
б	0431	т	0442
в	0432	у	0443
г	0433	ф	0444
д	0434	х	0445
е	0435	ц	0446
ж	0436	ч	0447
з	0437	ш	0448
й	0439	ы	044B
к	043A	ь	044C
л	043B	э	044D
м	043C	ю	044E

字符	代码	字符	代码
н	043D	я	044F
о	043E	ё	0451
п	043F	і	0456
р	0440	ў	045E

孟加拉语字符集

对于孟加拉语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ঁ	0981	দ	09A6
ং	0982	ধ	09A7
ঃ	0983	ন	09A8
অ	0985	প	09AA
আ	0986	ফ	09AB
ই	0987	ব	09AC
ঈ	0988	ভ	09AD
উ	0989	ম	09AE
ঊ	098A	য	09AF

字符	代码	字符	代码
ঋ	098B	র	09B0
এ	098F	ল	09B2
ঐ	0990	শ	09B6
ও	0993	ষ	09B7
ঔ	0994	স	09B8
ক	0995	হ	09B9
খ	0996	়	09BC
গ	0997	#	09BD
ঘ	0998	া	09BE
ঙ	0999	ি	09BF
চ	099A	ী	09C0
ছ	099B	ু	09C1
জ	099C	ূ	09C2
ঝ	099D	ৃ	09C3
ঞ	099E	ৄ	09C4
ট	099F	ে	09C7
ঠ	09A0	ৈ	09C8
ড	09A1	ো	09CB
ঢ	09A2	ৌ	09CC
ণ	09A3	্	09CD

字符	代码	字符	代码
ত	09A4	#	09CE
থ	09A5	ৗ	09D7

波斯尼亚语字符集

对于波斯尼亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ć	0107	đ	0111
č	010D	š	0161
ž	017E		

保加利亚语字符集

对于保加利亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	п	043F
б	0431	р	0440
в	0432	с	0441
г	0433	т	0442
д	0434	у	0443
е	0435	ф	0444
ж	0436	х	0445
з	0437	ц	0446
и	0438	ч	0447
й	0439	ш	0448
к	043A	щ	0449
л	043B	ъ	044A
м	043C	ь	044C
н	043D	ю	044E
о	043E	я	044F

加泰罗尼亚语字符集

对于加泰罗尼亚语自定义词汇表，您可以在 `Phrase` 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
à	00E0	ï	00EF
ç	00E7	ò	00F2
è	00E8	ó	00F3
é	00E9	ú	00FA
í	00ED	ü	00FC
l·	0140		

中库尔德语字符集

对于中库尔德语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ئ	0626	م	0645
ا	0627	ن	0646
ب	0628	و	0648
ت	062A	پ	067E
ج	062C	چ	0686
ح	062D	ڕ	0695

字符	代码	字符	代码
خ	062E	ژ	0698
د	062F	ڤ	06A4
ر	0631	ک	06A9
ز	0632	گ	06AF
س	0633	ڵ	06B5
ش	0634	ھ	06BE
ع	0639	ۆ	06C6
غ	063A	ۇ	06C7
ف	0641	ی	06CC
ق	0642	ێ	06CE
ل	0644	ە	06D5

中文、普通话（中国大陆）、简体字符集

对于简体中文自定义词汇表，Phrase 字段可以使用下列文件中列出的任何字符。

- zh-cn-character-set

SoundsLike 字段可以包含下列文件中列出的拼音音节。

- pinyin-character-set

当您在SoundsLike 字段中使用拼音音节时，请使用连字符 (-) 分隔音节。

Amazon Transcribe 使用数值表示简体中文中的四种音调。下表显示了如何为“ma”字映射音调符号。

音调	音调符号	音调编号
一声	mā	ma1
二声	má	ma2
三声	mǎ	ma3
四声	mà	ma4

> **Note**
>
> 对于第 5 个（中性）音调，您可以使用音调 1，但“er”除外，其必须映射到音调 2。例如，“打转儿”会以“da3-zhuan4-er2”的形式表示。

简体中文自定义词汇表不使用 IPA 字段，但您仍必须在自定义词汇表中包含 IPA 标头。

以下示例是文本格式的输入文件。该示例使用空格对齐列。您的输入文件应使用制表符字符分隔列。仅在 DisplayAs 列中包含空格。

```
Phrase       SoundsLike               IPA     DisplayAs
##           kang1-jian4
##           qian3-ze2
####         guo2-fang2-da4-chen2
#####      shi4-jie4-bo2-lan3-hui4          ###
```

中文、普通话（台湾）、繁体字符集

对于繁体中文自定义词汇表，Phrase 字段可以使用下列文件中列出的任何字符。

- zh-tw-character-set

SoundsLike 字段可以包含下列文件中列出的注音音节。

- zhuyin-character-set

当您在 SoundsLike 字段中使用注音音节时，请使用连字符 (-) 分隔音节。

Amazon Transcribe 使用数值表示繁体中文中的四种音调。下表显示了如何为“ㄇㄚ”字映射音调符号。

音调	音调符号	
一声	ㄇㄚ	
二声	ㄇㄚˊ	
三声	ㄇㄚˇ	
四声	ㄇㄚˋ	

繁体中文自定义词汇表不使用 IPA 字段，但您仍必须在自定义词汇表中包含 IPA 标头。

以下示例是文本格式的输入文件。该示例使用空格对齐列。您的输入文件应使用制表符字符分隔列。仅在 DisplayAs 列中包含空格。

```
Phrase        SoundsLike                    IPA        DisplayAs
##           ###ˋ-##
##           ###ˇ-##ˊ
####        ###ˊ-##ˊ-##ˋ-##ˇ
#####     #ˋ-###ˋ-##ˊ-##ˇ-###ˋ            ###
```

克罗地亚语字符集

对于克罗地亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ć	0107	đ	0111
č	010D	š	0161

字符	代码	字符	代码
ž	017E		

捷克语字符集

对于捷克语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
á	00E1	ď	010F
é	00E9	ě	011B
í	00ED	ň	0148
ó	00F3	ř	0159
ú	00FA	š	0161
ý	00FD	ť	0165
č	010D	ů	016F
ž	017E		

丹麦语字符集

对于丹麦语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z

- A - Z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
Å	00C5	æ	00E6
Æ	00C6	é	00E9
Ø	00D8	ø	00F8
å	00E5		

荷兰语字符集

对于荷兰语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
à	00E0	î	00EE
á	00E1	ï	00EF
â	00E2	ñ	00F1

字符	代码	字符	代码
ä	00E4	ò	00F2
ç	00E7	ó	00F3
è	00E8	ô	00F4
é	00E9	ö	00F6
ê	00EA	ù	00F9
ë	00EB	ú	00FA
ì	00EC	û	00FB
í	00ED	ü	00FC

英语字符集

对于英语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

爱沙尼亚语字符集

对于爱沙尼亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ä	00E4	ü	00FC
õ	00F5	š	0161
ö	00F6	ž	017E

波斯语字符集

对于波斯语自定义词汇表，您可以在 Phrase 字段中使用以下字符。

字符	代码	字符	代码
ء	0621	ظ	0638
آ	0622	ع	0639
أ	0623	غ	063A
ؤ	0624	ف	0641
ئ	0626	ق	0642
ا	0627	ل	0644
ب	0628	م	0645
ت	062A	ن	0646
ث	062B	ه	0647
ج	062C	و	0648
ح	062D	ـَ	064E
خ	062E	ـُ	064F
د	062F	ـِ	0650

字符	代码	字符	代码
ذ	0630	ّ	0651
ر	0631	پ	067E
ز	0632	چ	0686
س	0633	ژ	0698
ش	0634	ک	06A9
ص	0635	گ	06AF
ض	0636	ی	06CC
ط	0637		

芬兰语字符集

对于芬兰语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ä	00E4	ö	00F6
å	00E5	š	0161
ž	017E		

法语字符集

对于法语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
À	00C0	à	00E0
Â	00C2	â	00E2
Ç	00C7	ç	00E7
È	00C8	è	00E8
É	00C9	é	00E9
Ê	00CA	ê	00EA
Ë	00CB	ë	00EB
Î	00CE	î	00EE
Ï	00CF	ï	00EF
Ô	00D4	ô	00F4
Ö	00D6	ö	00F6
Ù	00D9	ù	00F9
Û	00DB	û	00FB

字符	代码	字符	代码
Ü	00DC	ü	00FC

加利西亚语字符集

对于加利西亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
á	00E1	ñ	00F1
é	00E9	ó	00F3
í	00ED	ú	00FA
ü	00FC		

格鲁吉亚语字符集

对于格鲁吉亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ა	10D0	რ	10E0
ბ	10D1	ს	10E1
გ	10D2	ტ	10E2
დ	10D3	უ	10E3
ე	10D4	ფ	10E4
ვ	10D5	ქ	10E5
ზ	10D6	ღ	10E6
თ	10D7	ყ	10E7
ი	10D8	შ	10E8
კ	10D9	ჩ	10E9
ლ	10DA	ც	10EA
მ	10DB	ძ	10EB
ნ	10DC	წ	10EC
ო	10DD	ჭ	10ED
პ	10DE	ხ	10EE
ჟ	10DF	ჯ	10EF
ს	10F0		

德语字符集

对于德语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ä	00E4	Ä	00C4
ö	00F6	Ö	00D6
ü	00FC	Ü	00DC
ß	00DF		

希腊语字符集

对于希腊语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ά	03AC	ν	03BD
έ	03AD	ξ	03BE
ή	03AE	ο	03BF

字符	代码	字符	代码
ί	03AF	π	03C0
ΰ	03B0	ρ	03C1
α	03B1	ς	03C2
β	03B2	σ	03C3
γ	03B3	τ	03C4
δ	03B4	υ	03C5
ε	03B5	φ	03C6
ζ	03B6	χ	03C7
η	03B7	ψ	03C8
θ	03B8	ω	03C9
ι	03B9	ϊ	03CA
κ	03BA	ϋ	03CB
λ	03BB	ό	03CC
μ	03BC	ώ	03CE
ΐ	0390		

古吉拉特语字符集

对于古吉拉特语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ં	0A81	દ	0AA6
ઃ	0A82	ધ	0AA7
:	0A83	ન	0AA8
અ	0A85	પ	0AAA
આ	0A86	ફ	0AAB
ઇ	0A87	બ	0AAC
ઈ	0A88	ભ	0AAD
ઉ	0A89	મ	0AAE
ઊ	0A8A	ય	0AAF
ઋ	0A8B	ર	0AB0
ઍ	0A8D	લ	0AB2
એ	0A8F	ળ	0AB3
ઐ	0A90	વ	0AB5
ઑ	0A91	શ	0AB6
ઓ	0A93	ષ	0AB7
ઔ	0A94	સ	0AB8
ક	0A95	હ	0AB9
ખ	0A96	઼	0ABC
ગ	0A97	ા	0ABE

字符	代码	字符	代码
ઘ	0A98	િ	0ABF
ઙ	0A99	ી	0AC0
ચ	0A9A	ુ	0AC1
છ	0A9B	ૂ	0AC2
જ	0A9C	ૃ	0AC3
ઝ	0A9D	ૅ	0AC5
ઞ	0A9E	ે	0AC7
ટ	0A9F	ૈ	0AC8
ઠ	0AA0	ૉ	0AC9
ડ	0AA1	ો	0ACB
ઢ	0AA2	ૌ	0ACC
ણ	0AA3	્	0ACD
ત	0AA4	ૐ	0AD0
થ	0AA5	ૠ	0AE0

豪萨语字符集

对于豪萨语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ƙ	0199	ɓ	0253
ƴ	01B4	ɗ	0257
̃	0303		

希伯来语字符集

对于希伯来语自定义词汇表，您可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
-	002D	ם	05DD
א	05D0	מ	05DE
ב	05D1	ן	05DF
ג	05D2	נ	05E0
ד	05D3	ס	05E1
ה	05D4	ע	05E2
ו	05D5	ף	05E3
ז	05D6	פ	05E4
ח	05D7	ץ	05E5
ט	05D8	צ	05E6
י	05D9	ק	05E7
ך	05DA	ר	05E8
כ	05DB	ש	05E9

字符	代码	字符	代码
ׇ	05DC	ת	05EA

印地语字符集

对于印地语自定义词汇表，您可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
-	002D	थ	0925
.	002E	द	0926
ँ	0901	ध	0927
ं	0902	न	0928
ः	0903	प	092A
अ	0905	फ	092B
आ	0906	ब	092C
इ	0907	भ	092D
ई	0908	म	092E
उ	0909	य	092F
ऊ	090A	र	0930
ऋ	090B	ल	0932
ए	090F	व	0935
ऐ	0910	श	0936
ऑ	0911	ष	0937

字符	代码	字符	代码
ओ	0913	स	0938
औ	0914	ह	0939
क	0915	ा	093E
ख	0916	ि	093F
ग	0917	ी	0940
घ	0918	ु	0941
ङ	0919	ू	0942
च	091A	ृ	0943
छ	091B	ॅ	0945
ज	091C	ॉ	0947
झ	091D	ॊ	0948
ञ	091E	ॐ	0949
ट	091F	ो	094B
ठ	0920	ौ	094C
ड	0921	्	094D
ढ	0922	ज़	095B
ण	0923	ड़	095C
त	0924	ढ़	095D

Amazon Transcribe 映射以下字符：

字符	映射到
ऩ (0929)	न (0928)
ऱ (0931)	र (0930)
क़ (0958)	क (0915)
ख़ (0959)	ख (0916)
ग़ (095A)	ग (0917)
फ़ (095E)	फ (092B)
य़ (095F)	य (092F)

匈牙利语字符集

对于匈牙利语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
á	00E1	ö	00F6
é	00E9	ú	00FA
í	00ED	ü	00FC
ó	00F3	ő	0151
ű	0171		

冰岛语字符集

对于冰岛语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
á	00E1	ú	00FA
é	00E9	ý	00FD
ð	00F0	þ	00FE
í	00ED	æ	00E6
ó	00F3	ö	00F6

印度尼西亚语字符集

对于印度尼西亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

意大利语字符集

对于意大利语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z

- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
À	00C0	à	00E0
Ä	00C4	ä	00E4
Ç	00C7	ç	00E7
È	00C8	è	00E8
É	00C9	é	00E9
Ê	00CA	ê	00EA
Ë	00CB	ë	00EB
Ì	00CC	ì	00EC
Ò	00D2	ò	00F2
Ù	00D9	ù	00F9
Ü	00DC	ü	00FC

日文字符集

对于日语自定义词汇表，DisplayAs 字段支持所有平假名、片假名和汉字字符，以及全角罗马字大写字母。

Phrase 字段支持以下文件中所列的字符：

- ja-jp-character-set

卡布列语字符集

对于卡布列语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ï	00EF	ḍ	1E0D
č	010D	ḥ	1E25
ř	0159	ṛ	1E5B
ǧ	01E7	ṣ	1E63
ɛ	025B	ṭ	1E6D
ɣ	0263	ẓ	1E93

卡纳拉语字符集

对于卡纳拉语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ಂ	0C82	ಧ	0CA7

字符	代码	字符	代码
ಃ	0C83	ನ	0CA8
ಅ	0C85	ಪ	0CAA
ಆ	0C86	ಫ	0CAB
ಇ	0C87	ಬ	0CAC
ಈ	0C88	ಭ	0CAD
ಉ	0C89	ಮ	0CAE
ಊ	0C8A	ಯ	0CAF
ಋ	0C8B	ರ	0CB0
ಎ	0C8E	ಲ	0CB2
ಏ	0C8F	ಳ	0CB3
ಐ	0C90	ವ	0CB5
ಒ	0C92	ಶ	0CB6
ಓ	0C93	ಷ	0CB7
ಔ	0C94	ಸ	0CB8
ಕ	0C95	ಹ	0CB9
ಖ	0C96	#	0CBC
ಗ	0C97	#	0CBD
ಘ	0C98	ಾ	0CBE
ಙ	0C99	ಿ	0CBF
ಚ	0C9A	ೀ	0CC0

字符	代码	字符	代码
ಛ	0C9B	ಁ	0CC1
ಜ	0C9C	ಂ	0CC2
ಝ	0C9D	ಃ	0CC3
ಞ	0C9E	ಆ	0CC6
ಟ	0C9F	ಈ	0CC7
ಠ	0CA0	ಉ	0CC8
ಡ	0CA1	ಊ	0CCA
ಢ	0CA2	ಋ	0CCB
ಣ	0CA3	ಌ	0CCC
ತ	0CA4	ಎ	0CCD
ಥ	0CA5	ಏ	0CD5
ದ	0CA6	ಐ	0CD6
ೱ	0CE0		

哈萨克语字符集

对于哈萨克语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
т	0442	ы	044B
б	0431	я	044F
о	043E	с	0441
п	043F	h	04BB
ш	0448	д	0434
и	0438	р	0440
ч	0447	г	0433
н	043D	ё	0451
қ	049B	й	0439
і	0456	ө	04E9
щ	0449	в	0432
е	0435	э	044D
ә	04D9	ң	04A3
ю	044E	л	043B
з	0437	ф	0444
х	0445	к	043A
ц	0446	у	0443
ү	04AF	ж	0436
м	043C	ғ	0493
ь	044C	а	0430

字符	代码	字符	代码
ъ	044A	ӳ	04B1

基尼亚卢旺达语字符集

对于卢旺达语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
á	00E1	ó	00F3
â	00E2	ô	00F4
ã	00E3	ú	00FA
ç	00E7	ü	00FC
è	00E8	ā	0101
é	00E9	ē	0113
ê	00EA	ī	012B
ë	00EB	ō	014D
í	00ED	ū	016B
ï	00EF	´	0301

韩语字符集

对于韩语自定义词汇表，您可以在 Phrase 字段中使用任何朝鲜文音节。有关更多信息，请参阅维基百科上的朝鲜文音节。

吉尔吉斯语字符集

对于吉尔吉斯语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C
д	0434	#	045D
е	0435	ў	045E
ж	0436	џ	045F
з	0437	ґ	0491
и	0438	ғ	0493
й	0439	җ	0497
к	043A	ҙ	0499
л	043B	қ	049B

字符	代码	字符	代码
м	043C	ҝ	049F
н	043D	ҟ	04A1
о	043E	ҥ	04A3
п	043F	ҧ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB
т	0442	ҭ	04AD
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7
ш	0448	h	04BB
щ	0449	ҽ	04BD
ъ	044A	ҿ	04BF
ы	044B	#	04CA
ь	044C	ӑ	04D1
э	044D	ӓ	04D3
ю	044E	ӗ	04D7
я	044F	ә	04D9

字符	代码	字符	代码
#	0450	ʒ	04E1
ë	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
s	0455	ӱ	04F1
i	0456	ӳ	04F3
ї	0457	#	04F7
j	0458	ы	04F9

拉脱维亚语字符集

对于拉脱维亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ā	0101	ķ	0137
č	010D	ļ	013C
ē	0113	ņ	0146
ģ	0123	š	0161

字符	代码	字符	代码
ī	012B	ū	016B
ž	017E		

立陶宛语字符集

对于立陶宛语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ą	0105	į	012F
č	010D	š	0161
ę	0119	ų	0173
ė	0117	ū	016B
ž	017E		

卢干达语字符集

对于卢干达语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
Ÿ	00FF	ŋ	014B

马其顿语字符集

对于马其顿语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C
д	0434	#	045D
е	0435	ў	045E
ж	0436	џ	045F
з	0437	ґ	0491
и	0438	ғ	0493
й	0439	ҷ	0497
к	043A	ҙ	0499

字符	代码	字符	代码
л	043B	қ	049B
м	043C	ҝ	049F
н	043D	ҡ	04A1
о	043E	ң	04A3
п	043F	ҥ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB
т	0442	ҭ	04AD
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7
ш	0448	h	04BB
щ	0449	ҽ	04BD
ъ	044A	ҿ	04BF
ы	044B	#	04CA
ь	044C	ӑ	04D1
э	044D	ӓ	04D3
ю	044E	ӗ	04D7

字符	代码	字符	代码
я	044F	ҙ	04D9
ѐ	0450	ҙ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
ѕ	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	ӵ	04F7
ј	0458	ӹ	04F9

马来语字符集

对于马来语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

马拉雅拉姆语字符集

对于马拉雅拉姆语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)

- .（句点）

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ം	0D02	ന	0D28
ഃ	0D03	പ	0D2A
അ	0D05	ഫ	0D2B
ആ	0D06	ബ	0D2C
ഇ	0D07	ഭ	0D2D
ഈ	0D08	മ	0D2E
ഉ	0D09	യ	0D2F
ഊ	0D0A	ര	0D30
ഋ	0D0B	റ	0D31
എ	0D0E	ല	0D32
ഏ	0D0F	ള	0D33
ഐ	0D10	ഴ	0D34
ഒ	0D12	വ	0D35
ഓ	0D13	ശ	0D36
ഔ	0D14	ഷ	0D37
ക	0D15	സ	0D38
ഖ	0D16	ഹ	0D39
ഗ	0D17	ാ	0D3E

字符	代码	字符	代码
ഘ	0D18	ാ	0D3F
ങ	0D19	ി	0D40
ച	0D1A	ു	0D41
ഛ	0D1B	ൂ	0D42
ജ	0D1C	ൃ	0D43
ഝ	0D1D	െ	0D46
ഞ	0D1E	േ	0D47
ട	0D1F	ൈ	0D48
ഠ	0D20	ൊ	0D4A
ഡ	0D21	ോ	0D4B
ഢ	0D22	ൌ	0D4C
ണ	0D23	്	0D4D
ത	0D24	#	0D7A
ഥ	0D25	#	0D7B
ദ	0D26	#	0D7C
ധ	0D27	#	0D7D

马耳他语字符集

对于马耳他语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)

- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
à	00E0	ù	00F9
è	00E8	ċ	010B
ì	00EC	ġ	0121
ò	00F2	ħ	0127
ż	017C		

马拉地语字符集

对于马拉地语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ं	0901	थ	0925
ँ	0902	द	0926
:	0903	ध	0927
अ	0905	न	0928
आ	0906	प	092A

字符	代码	字符	代码
इ	0907	फ	092B
ई	0908	ब	092C
उ	0909	भ	092D
ऊ	090A	म	092E
ऋ	090B	य	092F
ऍ	090D	र	0930
ए	090F	ल	0932
ऐ	0910	ळ	0933
ऑ	0911	व	0935
ओ	0913	श	0936
औ	0914	ष	0937
क	0915	स	0938
ख	0916	ह	0939
ग	0917	़	093C
घ	0918	ा	093E
ङ	0919	ि	093F
च	091A	ी	0940
छ	091B	ु	0941
ज	091C	ू	0942
झ	091D	ृ	0943

字符	代码	字符	代码
अ	091E	ॅ	0945
ट	091F	ॉ	0947
ठ	0920	ॊ	0948
ड	0921	ॏ	0949
ढ	0922	ॗ	094B
ण	0923	ॗ	094C
त	0924	्	094D
ॐ	0950		

草甸马里语字符集

对于草甸马里语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C
д	0434	#	045D

字符	代码	字符	代码
е	0435	ў	045E
ж	0436	џ	045F
з	0437	ґ	0491
и	0438	ғ	0493
й	0439	ҙ	0497
к	043A	ҝ	0499
л	043B	қ	049B
м	043C	ҟ	049F
н	043D	ҡ	04A1
о	043E	ң	04A3
п	043F	ҥ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB
т	0442	ҭ	04AD
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7
ш	0448	һ	04BB

字符	代码	字符	代码
щ	0449	ҽ	04BD
ъ	044A	ҿ	04BF
ы	044B	#	04CA
ь	044C	ӑ	04D1
э	044D	ӓ	04D3
ю	044E	ӗ	04D7
я	044F	ә	04D9
#	0450	ӡ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
s	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	#	04F7
ј	0458	ӹ	04F9

蒙古语字符集

对于蒙古语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)

- .（句点）

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C
д	0434	#	045D
е	0435	ў	045E
ж	0436	џ	045F
з	0437	ґ	0491
и	0438	ғ	0493
й	0439	җ	0497
к	043A	ҙ	0499
л	043B	қ	049B
м	043C	ҟ	049F
н	043D	ҡ	04A1
о	043E	ң	04A3
п	043F	ҥ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB

字符	代码	字符	代码
т	0442	ҭ	04AD
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7
ш	0448	h	04BB
щ	0449	ҽ	04BD
ъ	044A	ҿ	04BF
ы	044B	#	04CA
ь	044C	ӑ	04D1
э	044D	ӓ	04D3
ю	044E	ӗ	04D7
я	044F	ә	04D9
#	0450	ӟ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
s	0455	ӱ	04F1

字符	代码	字符	代码
i	0456	ӳ	04F3
ï	0457	#	04F7
j	0458	ы	04F9

挪威布克莫尔语字符集

对于挪威布克莫尔语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
å	00E5	æ	00E6
ø	00F8		

奥迪亚/奥里亚语字符集

对于奥迪亚/奥里亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
◌	0B01	ଦ	0B26
ଂ	0B02	ଧ	0B27
ଃ	0B03	ନ	0B28
ଅ	0B05	ପ	0B2A
ଆ	0B06	ଫ	0B2B
ଇ	0B07	ବ	0B2C
ଈ	0B08	ଭ	0B2D
ଉ	0B09	ମ	0B2E
ଊ	0B0A	ଯ	0B2F
ଋ	0B0B	ର	0B30
ଏ	0B0F	ଲ	0B32
ଐ	0B10	ଳ	0B33
ଓ	0B13	ଗ	0B36
ଔ	0B14	ଷ	0B37
କ	0B15	ସ	0B38
ଖ	0B16	ହ	0B39
ଗ	0B17	଼	0B3C
ଘ	0B18	ା	0B3E
ଙ	0B19	◌ି	0B3F
ଚ	0B1A	◌ୀ	0B40

字符	代码	字符	代码
ଛ	0B1B	ୁ	0B41
ଜ	0B1C	ୂ	0B42
ଝ	0B1D	ୃ	0B43
ଞ	0B1E	େ	0B47
ଟ	0B1F	ୈ	0B48
ଠ	0B20	େ#	0B4B
ଡ	0B21	ୋ	0B4C
ଢ	0B22	୍	0B4D
ଣ	0B23	ୖ	0B56
ତ	0B24	ୟ	0B5F
ଥ	0B25	ୠ	0B60
#	0B71		

普什图语字符集

对于普什图语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
آ	0622	و	0648

字符	代码	字符	代码
أ	0623	ي	064A
وْ	0624	ً	064B
ئ	0626	ٌ	064C
ا	0627	ٍ	064D
ب	0628	َ	064E
ت	062A	ُ	064F
ث	062B	ِ	0650
ج	062C	ّ	0651
ح	062D	°	0652
خ	062E	#	0654
د	062F	ٱ	0670
ذ	0630	ټ	067C
ر	0631	پ	067E
ز	0632	ځ	0681
س	0633	څ	0685
ش	0634	چ	0686
ص	0635	ډ	0689
ض	0636	ړ	0693
ط	0637	ږ	0696
ظ	0638	ژ	0698

字符	代码	字符	代码
ع	0639	ښ	069A
غ	063A	ک	06A9
ف	0641	ګ	06AB
ق	0642	گ	06AF
ل	0644	ڼ	06BC
م	0645	ی	06CC
ن	0646	ۍ	06CD
ه	0647	ې	06D0

波兰语字符集

对于波兰语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ó	00F3	ł	0142
ą	0105	ń	0144
ć	0107	ś	015B
ę	0119	ź	017A
ż	017C		

葡萄牙语字符集

对于葡萄牙语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
À	00C0	à	00E0
Á	00C1	á	00E1
Â	00C2	â	00E2
Ã	00C3	ã	00E3
Ä	00C4	ä	00E4
Ç	00C7	ç	00E7
È	00C8	è	00E8
É	00C9	é	00E9
Ê	00CA	ê	00EA
Ë	00CB	ë	00EB
Í	00CD	í	00ED
Ñ	00D1	ñ	00F1
Ó	00D3	ó	00F3

字符	代码	字符	代码
Ô	00D4	ô	00F4
Õ	00D5	õ	00F5
Ö	00D6	ö	00F6
Ú	00DA	ú	00FA
Ü	00DC	ü	00FC

旁遮普语字符集

对于旁遮普语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ਅ	0A05	ਧ	0A27
ਆ	0A06	ਨ	0A28
ਇ	0A07	ਪ	0A2A
ਈ	0A08	ਫ	0A2B
ਉ	0A09	ਬ	0A2C
ਊ	0A0A	ਭ	0A2D
ਏ	0A0F	ਮ	0A2E
ਐ	0A10	ਯ	0A2F

字符	代码	字符	代码
ਓ	0A13	ਰ	0A30
ਔ	0A14	ਲ	0A32
ਕ	0A15	ਵ	0A35
ਖ	0A16	ਸ	0A38
ਗ	0A17	ਹ	0A39
ਘ	0A18	਼	0A3C
ਙ	0A19	ਾ	0A3E
ਚ	0A1A	ਿ	0A3F
ਛ	0A1B	ੀ	0A40
ਜ	0A1C	ੁ	0A41
ਝ	0A1D	ੂ	0A42
ਞ	0A1E	ੇ	0A47
ਟ	0A1F	ੈ	0A48
ਠ	0A20	ੋ	0A4B
ਡ	0A21	ੌ	0A4C
ਢ	0A22	੍	0A4D
ਣ	0A23	ੜ	0A5C
ਤ	0A24	ੰ	0A70
ਥ	0A25	ੱ	0A71
ਦ	0A26	ੲ	0A72

字符	代码	字符	代码
ఀ	0A73		

罗马尼亚语字符集

对于罗马尼亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ă	0103	#	0219
â	00E2	#	021B
î	00EE	ș	015F
ț	0163		

俄语字符集

对于俄语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

字符	代码	字符	代码
'	0027	п	043F
-	002D	р	0440
.	002E	с	0441
а	0430	т	0442

字符	代码	字符	代码
б	0431	у	0443
в	0432	ф	0444
г	0433	х	0445
д	0434	ц	0446
е	0435	ч	0447
ж	0436	ш	0448
з	0437	щ	0449
и	0438	ъ	044A
й	0439	ы	044B
к	043A	ь	044C
л	043B	э	044D
м	043C	ю	044E
н	043D	я	044F
о	043E	ё	0451

塞尔维亚语字符集

对于塞尔维亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ć	0107	і	0456
č	010D	ї	0457
đ	0111	ј	0458
š	0161	љ	0459
ž	017E	њ	045A
а	0430	ћ	045B
б	0431	ќ	045C
в	0432	#	045D
г	0433	ў	045E
д	0434	џ	045F
е	0435	ґ	0491
ж	0436	ғ	0493
з	0437	җ	0497
и	0438	ҙ	0499
й	0439	қ	049B
к	043A	ҝ	049F
л	043B	ҡ	04A1
м	043C	ң	04A3
н	043D	ҥ	04A5
о	043E	ҩ	04A9

字符	代码	字符	代码
п	043F	ҫ	04AB
р	0440	ҭ	04AD
с	0441	ү	04AF
т	0442	ұ	04B1
у	0443	ҳ	04B3
ф	0444	ҵ	04B5
х	0445	ҷ	04B7
ц	0446	h	04BB
ч	0447	ҽ	04BD
ш	0448	ҿ	04BF
щ	0449	#	04CA
ъ	044A	ӑ	04D1
ы	044B	ӓ	04D3
ь	044C	ӗ	04D7
э	044D	ә	04D9
ю	044E	ӟ	04E1
я	044F	ӣ	04E3
#	0450	ӧ	04E7
ё	0451	ө	04E9
ђ	0452	ӯ	04EF

字符	代码	字符	代码
ѓ	0453	ӱ	04F1
є	0454	ӳ	04F3
s	0455	#	04F7
ӹ	04F9		

僧伽罗语字符集

对于僧伽罗语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
#	0D82	#	0DAF
#	0D83	#	0DB0
#	0D85	#	0DB1
#	0D86	#	0DB3
#	0D87	#	0DB4
#	0D88	#	0DB5
#	0D89	#	0DB6
#	0D8A	#	0DB7
#	0D8B	#	0DB8

字符	代码	字符	代码
#	0D8C	#	0DB9
#	0D8D	#	0DBA
#	0D91	#	0DBB
#	0D92	#	0DBD
#	0D93	#	0DC0
#	0D94	#	0DC1
#	0D95	#	0DC2
#	0D96	#	0DC3
#	0D9A	#	0DC4
#	0D9B	#	0DC5
#	0D9C	#	0DC6
#	0D9D	#	0DCA
#	0D9E	#	0DCF
#	0D9F	#	0DD0
#	0DA0	#	0DD1
#	0DA1	#	0DD2
#	0DA2	#	0DD3
#	0DA3	#	0DD4
#	0DA4	#	0DD6
#	0DA5	#	0DD8

字符	代码	字符	代码
#	0DA7	#	0DD9
#	0DA8	##	0DDA
#	0DA9	#	0DDB
#	0DAA	##	0DDC
#	0DAB	###	0DDD
#	0DAC	##	0DDE
#	0DAD	#	0DDF
#	0DAE	#	0DF2

斯洛伐克语字符集

对于斯洛伐克语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
á	00E1	ň	0148
ä	00E4	ó	00F3
č	010D	ô	00F4
ď	010F	ŕ	0155
é	00E9	š	0161

字符	代码	字符	代码
í	00ED	ť	0165
Í	013A	ú	00FA
Ľ	013E	ý	00FD
ž	017E		

斯洛文尼亚语字符集

对于斯洛文尼亚语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
č	010D	š	0161
ž	017E		

索马里语字符集

对于索马里语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
s	0073	d	0064
t	0074	a	0061
a	0061	r	0072
n	006E	d	0064

西班牙语字符集

对于西班牙语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
Á	00C1	á	00E1
É	00C9	é	00E9
Í	00CD	í	00ED
Ó	00D3	ó	0XF3
Ú	00DA	ú	00FA
Ñ	00D1	ñ	0XF1
ü	00FC		

巽他语字符集

对于巽他语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
s	0073	d	0064
t	0074	a	0061
a	0061	r	0072
n	006E	d	0064

斯瓦西里语字符集

对于斯瓦西里语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
s	0073	d	0064
t	0074	a	0061
a	0061	r	0072

字符	代码	字符	代码
n	006E	d	0064

瑞典语字符集

对于瑞典语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
Ä	00C4	ä	00E4
Å	00C5	å	00E5
Ö	00D6	ö	00F6

塔加洛语/菲律宾语字符集

对于塔加洛语/菲律宾语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码		
ñ	00F1		

泰米尔语字符集

对于泰米尔语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

字符	代码	字符	代码
அ	0B85	ர	0BB0
ஆ	0B86	ல	0BB2
இ	0B87	வ	0BB5
ஈ	0B88	ழ	0BB4
உ	0B89	ள	0BB3
ஊ	0B8A	ற	0BB1
எ	0B8E	ன	0BA9
ஏ	0B8F	ஜ	0B9C
ஐ	0B90	#	0BB6
ஒ	0B92	ஷ	0BB7
ஓ	0B93	ஸ	0BB8
ஔ	0B94	ஹ	0BB9
ஃ	0B83	்	0BCD
க	0B95	ா	0BBE
ங	0B99	ி	0BBF

字符	代码	字符	代码
ச	0B9A	ை	0BC0
ஞ	0B9E	⁓	0BC1
ட	0B9F	ொ	0BC2
ண	0BA3	ெ	0BC6
த	0BA4	ே	0BC7
ந	0BA8	ை	0BC8
ப	0BAA	ொ	0BCA
ம	0BAE	ோ	0BCB
ய	0BAF	ௌ	0BCC

鞑靼语字符集

对于鞑靼语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C

字符	代码	字符	代码
д	0434	ѝ	045D
е	0435	ў	045E
ж	0436	џ	045F
з	0437	ґ	0491
и	0438	ғ	0493
й	0439	ҙ	0497
к	043A	ҝ	0499
л	043B	қ	049B
м	043C	ҟ	049F
н	043D	ҡ	04A1
о	043E	ң	04A3
п	043F	ҥ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB
т	0442	ҭ	04AD
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7

字符	代码	字符	代码
ш	0448	h	04BB
щ	0449	ҽ	04BD
ъ	044A	ҿ	04BF
ы	044B	#	04CA
ь	044C	ӑ	04D1
э	044D	ӓ	04D3
ю	044E	ӗ	04D7
я	044F	ә	04D9
#	0450	ӡ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
s	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	#	04F7
ј	0458	ӹ	04F9

泰卢固语字符集

对于泰卢固语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

字符	代码	字符	代码
-	002D	త	0C24
ఁ	0C01	థ	0C25
ం	0C02	ద	0C26
ః	0C03	ధ	0C27
అ	0C05	న	0C28
ఆ	0C06	ప	0C2A
ఇ	0C07	ఫ	0C2B
ఈ	0C08	బ	0C2C
ఉ	0C09	భ	0C2D
ఊ	0C0A	మ	0C2E
ఋ	0C0B	య	0C2F
ర	0C30	ఎ	0C0E
ఱ	0C31	ఏ	0C0F
ల	0C32	ఐ	0C10
ళ	0C33	ఒ	0C12
వ	0C35	ఓ	0C13
శ	0C36	ఔ	0C14
ష	0C37	క	0C15
స	0C38	ఖ	0C16
హ	0C39	గ	0C17

字符	代码	字符	代码
ా	0C3E	ఘ	0C18
ి	0C3F	ఙ	0C19
ీ	0C40	చ	0C1A
ు	0C41	ఛ	0C1B
ూ	0C42	జ	0C1C
ృ	0C43	ఝ	0C1D
ౄ	0C44	ఞ	0C1E
ె	0C47	ట	0C1F
ే	0C48	ఠ	0C20
ై	0C4A	డ	0C21
ొ	0C4B	ఢ	0C22
ో	0C4C	ణ	0C23
్	0C4D		

泰语字符集

对于泰语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
ก	0E01	ล	0E25
ข	0E02	ฦ	0E26
ฃ	0E03	ว	0E27
ค	0E04	ศ	0E28
ฅ	0E05	ษ	0E29
ฆ	0E06	ส	0E2A
ง	0E07	ห	0E2B
จ	0E08	ฬ	0E2C
ฉ	0E09	อ	0E2D
ช	0E0A	ฮ	0E2E
ซ	0E0B	ฯ	0E2F
ฌ	0E0C	ะ	0E30
ญ	0E0D	ั	0E31
ฎ	0E0E	า	0E32
ฏ	0E0F	ิ	0E34
ฐ	0E10	ี	0E35
ฑ	0E11	ึ	0E36
ฒ	0E12	ื	0E37
ณ	0E13	ุ	0E38
ด	0E14	ู	0E39

字符	代码	字符	代码
ต	0E15	ฺ	0E3A
ถ	0E16	เ	0E40
ท	0E17	แ	0E41
ธ	0E18	โ	0E42
น	0E19	ใ	0E43
บ	0E1A	ไ	0E44
ป	0E1B	ๅ	0E45
ผ	0E1C	ๆ	0E46
ฝ	0E1D	็	0E47
พ	0E1E	่	0E48
ฟ	0E1F	้	0E49
ภ	0E20	๊	0E4A
ม	0E21	๋	0E4B
ย	0E22	์	0E4C
ร	0E23	๎	0E4D
ฤ	0E24		

土耳其语字符集

对于土耳其语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- A - Z

- ' (撇号)
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
Ç	00C7	ö	00F6
Ö	00D6	û	00FB
Ü	00DC	ü	00FC
â	00E2	Ğ	011E
ä	00E4	ğ	011F
ç	00E7	İ	0130
è	00E8	ı	0131
é	00E9	Ş	015E
ê	00EA	ş	015F
í	00ED	š	0161
î	00EE	ž	017E
ó	00F3		

乌克兰语字符集

对于乌克兰语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
а	0430	р	0440
б	0431	с	0441
в	0432	т	0442
г	0433	у	0443
д	0434	ф	0444
е	0435	х	0445
ж	0436	ц	0446
з	0437	ч	0447
и	0438	ш	0448
й	0439	щ	0449
к	043A	ь	044C
л	043B	ю	044E
м	043C	я	044F
н	043D	є	0454
о	043E	і	0456
п	043F	ї	0457
ґ	0491		

维吾尔语字符集

对于维吾尔语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
#	0611	و	0648
#	0613	ى	0649
#	0614	ي	064A
ء	0621	ً	064B
آ	0622	ٌ	064C
أ	0623	ٍ	064D
ؤ	0624	َ	064E
إ	0625	ُ	064F
ئ	0626	ِ	0650
ا	0627	ّ	0651
ب	0628	ْ	0652
ة	0629	#	0653
ت	062A	#	0654
ث	062B	#	0657
ج	062C	ٰ	0670
ح	062D	ٹ	0679

字符	代码	字符	代码
خ	062E	ٺ	067A
د	062F	پ	067B
ذ	0630	ټ	067C
ر	0631	ٹ	067D
ز	0632	ٻ	067E
س	0633	ٽ	067F
ش	0634	ڀ	0680
ص	0635	ځ	0681
ض	0636	ڃ	0683
ط	0637	ڄ	0684
ظ	0638	څ	0685
ع	0639	چ	0686
غ	063A	ڇ	0687
ـ	0640	ڈ	0688
ف	0641	ډ	0689
ق	0642	ڊ	068A
ك	0643	ڌ	068C
ل	0644	ڍ	068D
م	0645	ڎ	068F
ن	0646	ڑ	0691

字符	代码	字符	代码
ۇ	0647	ۋ	0693
ۯ	0695		

乌兹别克语字符集

对于乌兹别克语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
т	0442	я	044F
б	0431	с	0441
о	043E	ҳ	04B3
п	043F	д	0434
ш	0448	р	0440
и	0438	ў	045E
ч	0447	г	0433
н	043D	ё	0451
қ	049B	й	0439
е	0435	в	0432
ю	044E	э	044D

字符	代码	字符	代码
з	0437	л	043B
х	0445	ф	0444
ц	0446	к	043A
м	043C	у	0443
ь	044C	ж	0436
ъ	044A	ғ	0493
а	0430		

越南语字符集

Amazon Transcribe 用数值表示越南语中的六种音调。下表显示了如何为“ma”字映射音调符号。

音调名称	音调符号	音调编号
ngang	ma	ma1
sắc	má	ma2
huyền	mà	ma3
hỏi	mả	ma4
ngã	mã	ma5
nặng	mạ	ma6

对于越南语自定义词汇表，您可以在 `Phrase` 字段中使用以下字符：

- a - z
- A - Z
- '（撇号）

- - (连字符)
- . (句点)
- & (与)
- ; (分号)
- _ (下划线)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
à	00E0	À	00C0
á	00E1	Á	00C1
â	00E2	Â	00C2
ã	00E3	Ã	00C3
è	00E8	È	00C8
é	00E9	É	00C9
ê	00EA	Ê	00CA
ì	00EC	Ì	00CC
í	00ED	Í	00CD
ò	00F2	Ò	00D2
ó	00F3	Ó	00D3
ô	00F4	Ô	00D4
õ	00F5	Õ	00D5
ù	00F9	Ù	00D9
ú	00FA	Ú	00DA

字符	代码	字符	代码
ý	00FD	Ý	00DD
ă	0103	Ă	0102
đ	0111	Đ	0110
ĩ	0129	Ĩ	0128
ũ	0169	Ũ	0168
ơ	01A1	Ơ	01A0
ư	01B0	Ư	01AF
ạ	1EA1	Ạ	1EA0
ả	1EA3	Ả	1EA2
ấ	1EA5	Ấ	1EA4
ầ	1EA7	Ầ	1EA6
ẩ	1EA9	Ẩ	1EA8
ẫ	1EAB	Ẫ	1EAA
ậ	1EAD	Ậ	1EAC
ắ	1EAF	Ắ	1EAE
ằ	1EB1	Ằ	1EB0
ẳ	1EB3	Ẳ	1EB2
ẵ	1EB5	Ẵ	1EB4
ặ	1EB7	Ặ	1EB6
ẹ	1EB9	Ẹ	1EB8

字符	代码	字符	代码
ẻ	1EBB	Ẻ	1EBA
ẽ	1EBD	Ẽ	1EBC
ế	1EBF	Ế	1EBE
ề	1EC1	Ề	1EC0
ể	1EC3	Ể	1EC2
ễ	1EC5	Ễ	1EC4
ệ	1EC7	Ệ	1EC6
ỉ	1EC9	Ỉ	1EC8
ị	1ECB	Ị	1ECA
ọ	1ECD	Ọ	1ECC
ỏ	1ECF	Ỏ	1ECE
ố	1ED1	Ố	1ED0
ồ	1ED3	Ồ	1ED2
ổ	1ED5	Ổ	1ED4
ỗ	1ED7	Ỗ	1ED6
ộ	1ED9	Ộ	1ED8
ớ	1EDB	Ớ	1EDA
ờ	1EDD	Ờ	1EDC
ở	1EDF	Ở	1EDE
ỡ	1EE1	Ỡ	1EE0

字符	代码	字符	代码
ợ	1EE3	Ợ	1EE2
ụ	1EE5	Ụ	1EE4
ủ	1EE7	Ủ	1EE6
ứ	1EE9	Ứ	1EE8
ừ	1EEB	Ừ	1EEA
ử	1EED	Ử	1EEC
ữ	1EEF	Ữ	1EEE
ự	1EF1	Ự	1EF0
ỳ	1EF3	Ỳ	1EF2
ỵ	1EF5	Ỵ	1EF4
ỷ	1EF7	Ỷ	1EF6
ỹ	1EF9	Ỹ	1EF8

威尔士语字符集

对于威尔士语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
à	00E0	ò	00F2

字符	代码	字符	代码
á	00E1	ó	00F3
â	00E2	ô	00F4
ä	00E4	ö	00F6
è	00E8	ù	00F9
é	00E9	ú	00FA
ê	00EA	û	00FB
ë	00EB	ü	00FC
ì	00EC	ý	00FD
í	00ED	Ÿ	00FF
î	00EE	ŵ	0175
ï	00EF	ŷ	0177
ỳ	1EF3		

沃洛夫语字符集

对于沃洛夫语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
à	00E0	ê	00EA

字符	代码	字符	代码
ã	00E3	ë	00EB
ç	00E7	ñ	00F1
è	00E8	ó	00F3
é	00E9	ô	00F4
ŋ	014B		

祖鲁语字符集

对于祖鲁语自定义词汇表，您可以在 Phrase 字段中使用以下字符：

- a - z
- - (连字符)
- . (句点)

您还可以在 Phrase 字段中使用以下 Unicode 字符：

字符	代码	字符	代码
s	0073	d	0064
t	0074	a	0061
a	0061	r	0072
n	006E	d	0064

Amazon Transcribe 的工作原理

Amazon Transcribe使用机器学习模型将语音转换为文本。

除转录的文本外，笔录还包含有关转录内容的数据，包括每个单词或标点符号的置信度分数和时间戳。要查看输出示例，请参阅数据输入和输出部分。有关可以应用于转录的功能的完整列表，请参阅功能摘要。

转录方法分为两大类：

- B@@ atch T ranscribe：转录已上传到Amazon S3存储桶的媒体文件。您可以使用AWS CLIAWS Management Console、和各种 AWSSDK 进行批量转录。
- 流媒体Tran scribe：实时转录媒体流。您可以使用AWS Management Console、HTTP/2 和各种 AWSSDK 进行流式转录。WebSockets

请注意，批量转录和流式转录的功能和语言支持有所不同。有关更多信息，请参阅Amazon Transcribe features和支持的语言。

主题

- 数据输入和输出
- 转录数字和标点符号

> API 操作帮助您开始操作
>
> Batch：`StartTranscriptionJob`
> 直播：`StartStreamTranscription`， StartStreamTranscriptionWebSocket

数据输入和输出

Amazon Transcribe将音频数据作为Amazon S3存储桶或媒体流中的媒体文件，并将其转换为文本数据。

如果您正在转录存储在存储Amazon S3段中的媒体文件，则是在执行批量转录。如果您正在转录媒体流，则是在执行流媒体转录。这两个流程有不同的规则和要求。

使用批量转录，任务队列列任务队列如果您不需要同时处理所有转录作业，则可以使用批量转录。这Amazon Transcribe允许跟踪您的转录作业，并在插槽可用时对其进行处理。

> **Note**
>
> Amazon Transcribe可能会暂时存储您的内容以不断提高其分析模型的质量。要了解更多信息，请参阅 Amazon Transcribe 常见问题。要请求删除可能由存储的内容Amazon Transcribe，请使用打开案例AWS Support。

主题

媒体格式

批量转录和流式转录支持的媒体类型有所不同，但建议两者都采用无损格式。有关详细信息，请参阅下表：

	Batch	流式处理
支持的格式	• AMR • FLAC • M4A • MP3 • MP4 • Ogg • WebM • WAV	• FLAC • Ogg Opus • PCM 编码
推荐格式	• FLAC • 采用 PCM 16 位编码的 WAV	• FLAC • PCM 签名的 16 位小端音频（请注意，这不包括 WAV）

为获得最佳效果，请使用无损格式，例如采用 PCM 16 位编码的 FLAC 或 WAV。

> **Note**
>
> 并非所有语言都支持流媒体转录。有关详细信息，请参阅支持的语言表中的 “数据输入” 列。

音频频道

Amazon Transcribe支持单通道和双通道媒体。目前不支持超过两个频道的媒体。

如果您的音频在一个频道上包含多个扬声器，并且您想在转录输出中对每个扬声器进行分区和标记，则可以使用扬声器分区（diariation）。

如果您的音频包含两个独立频道上的语音，则可以使用频道识别在脚本中分别转录每个频道。

这两个选项都会生成一个脚本文件。

> **Note**
>
> 如果您未启用发言者分区或频道识别，则您的笔录文本将作为一个连续部分提供。

采样率

对于批量转录作业，您可以选择提供采样率，但此参数是可选的。如果您将其包含在请求中，请确保您提供的值与音频中的实际采样率相匹配。如果您提供的采样率与您的音频不匹配，则您的工作可能会失败。

对于流媒体转录，您必须在请求中包括采样率。与批量转录作业一样，请确保您提供的值与音频中的实际采样率相匹配。

低保真音频（例如电话录音）的采样率通常使用 8,000 Hz。对于高保真音频，Amazon Transcribe支持介于 16,000 Hz 和 48,000 Hz 之间的值。

Output

转录输出采用 JSON 格式。笔录的第一部分包含段落形式的笔录本身，然后是每个单词和标点符号的其他数据。提供的数据取决于您在请求中包含的功能。您的成绩单至少包含每个单词的开始时间、结束时间和置信度分数。以下部分显示了基本转录请求的输出示例，该请求不包含任何其他选项或功能。

所有批处理记录都存储在存储Amazon S3桶中。您可以选择将脚本保存在自己的Amazon S3存储桶中，也可以Amazon Transcribe使用安全的默认存储桶。要了解有关创建和使用Amazon S3存储桶的更多信息，请参阅使用存储桶。

如果您希望将您的脚本存储在您拥有的Amazon S3存储段中，请在转录请求中指定该存储段的 URI。在开始批量转录任务之前，请务必授予此存储段的Amazon Transcribe写入权限。如果您指定了自己的存储桶，则您的脚本将保留在该存储桶中，直到您将其删除。

如果您未指定Amazon S3存储桶，则Amazon Transcribe使用安全的服务管理存储分区并为您提供可用于下载脚本的临时 URI。请注意，临时 URI 的有效期为 15 分钟。如果您在使用所提供的 URI 时`AccessDenied`遇到错误，请`GetTranscriptionJob`请求为您的脚本获取新的临时 URI。

如果您选择默认存储桶，则您的脚本将在任务到期（90 天）时删除。如果您想在此到期日期之后保留您的成绩单，则必须下载该笔录。

直播记录的返回方法与你在直播中使用的方法相同。

> **Tip**
>
> 如果您想将 JSON 输出转换为 Word 格式的 turn-by-turn 脚本，请参阅此GitHub 示例（适用于 Python3）。此脚本适用于通话后分析记录和启用了迪亚化功能的标准批处理记录。

输出示例

笔录以段落形式提供完整的转录，然后进行 word-for-word 细分，为每个单词和标点符号提供数据。这包括开始时间、结束时间、置信度分数和类型（`pronunciation`或`punctuation`）。

以下示例来自一个简单的批量转录作业，该作业不包含任何其他功能。当您对转录请求应用每一项附加功能时，您将在转录输出文件中获得更多数据。

基本批处理记录包含两个主要部分：

1. `transcripts`：将整个脚本包含在一个文本块中。
2. `items`：包含该`transcripts`部分中每个单词和标点符号的信息。

您在转录请求中包含的每项附加功能都会在您的成绩单中产生其他信息。

```
{
    "jobName": "my-first-transcription-job",
    "accountId": "111122223333",
```

```
    "results": {
        "transcripts": [
            {
                "transcript": "Welcome to Amazon Transcribe."
            }
        ],
        "items": [
            {
                "start_time": "0.64",
                "end_time": "1.09",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Welcome"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "1.09",
                "end_time": "1.21",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "to"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "1.21",
                "end_time": "1.74",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Amazon"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "1.74",
                "end_time": "2.56",
                "alternatives": [
```

```
                    {
                        "confidence": "1.0",
                        "content": "Transcribe"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ]
    },
    "status": "COMPLETED"
}
```

转录数字和标点符号

Amazon Transcribe自动为所有支持的语言添加标点符号，对于在书写系统中使用区分大小写的语言，相应地将单词大写。

对于大多数语言来说，数字都被转录成单词形式。但是，如果您的媒体使用英语或德语，则会根据使用数字的上下文对数字进行不同的Amazon Transcribe处理。

例如，如果说话者说 “”Meet me at eight-thirty AM on June first at one-hundred Main Street with three-dollars-and-fifty-cents and one-point-five chocolate bars，则将其转录为：

- 英语和德语方言：Meet me at 8:30 a.m. on June 1st at 100 Main Street with $3.50 and 1.5 chocolate bars
- 所有其他语言：Meet me at eight thirty a m on June first at one hundred Main Street with three dollars and fifty cents and one point five chocolate bars

要查看与英语和德语中的数字口语相关的所有规则，请参阅下表。

规则	英语方言 （输入音频 → 输出文本）	德语方言 （输入音频 → 输出文本）
将大于十的基数转换为数字。	• "Fifty five" → 55 • "a hundred" → 100 • "One thousand and thirty one" → 1031 • "One hundred twenty-three million four hundred fifty six thousand seven hundred eight nine" → 123,456,789	• "fünfundfünfzig" → 55 • "vier tausend sechs hundert einundachtzig" → 4681 • "eine Sache" → "eine Sache"
当“million”或“billion”后面没有数字时，将后跟“million”或“billion”的基数词转换为数字后跟一个单词。	• "one hundred million" → 100 million • "one billion" → 1 billion • "two point three million" → 2.3 million	• "zehn Millionen Menschen" → 10 Millionen Menschen • "zehn Millionen fünf hundert tausend" → 10.500.000
将大于十的序数转换为数字。	• "Forty third" → 43rd • "twenty sixth avenue" → 26th avenue	• "dreiundzwanzigste" → 23 • "vierzigster" → 40 • "ich war Erster" → "ich war Erster"
将小数转换成数字格式。	• "a quarter" → 1/4 • "three sixteenths" → 3/16 • "a half" → 1/2 • "a hundredth" → 1/100	分数不会转换为数字格式。 • "ein Drittel" → "ein Drittel"
如果连续有多个数字，则将小于十的数字转换为数字。	• "three four five" → 345 • "My phone number is four two five five five five one two one two" → My phone number is 4255551212	• "eins zwei drei" → 123 • "plus vier neun zwei vier eins" → +49241

规则	英语方言 （输入音频 → 输出文本）	德语方言 （输入音频 → 输出文本）
单词 “点” 或 “点” 显示为十进制。	• "three hundred and three dot five" → 303.5 • "three point twenty three" → 3.23 • "zero point four" → 0.4 • "point three" → 0.3	小数由 “,” 表示。 • "zweiundzwanzig komma drei" → 22,3
将单词“percent”转换成数字加百分比符号 (%)。	• "twenty three percent" → 23% • "twenty three point four five percent" → 23.45%	• "fünf Prozent Hürde" → 5% Hürde • "dreiundzwanzig komma vier Prozent" → 23,4%

规则	英语方言 （输入音频 → 输出文本）	德语方言 （输入音频 → 输出文本）
将货币词语转换为符号。	将数字后面的 “美元”、“美元”、“澳元”、“澳元” 或 “美元” 等词转换为数字前面的美元符号 ($)。 • "one dollar and fifteen cents" → $1.15 • "twenty three USD" → $23 • "twenty three Australian dollars" → $23 将数字后面的单词“pounds”、“British pounds”或“GDB“转换为英镑符号 (£) 加在数字之前。 • "twenty three pounds" → £23 • "I have two thousand pounds" → I have £2,000 • "five pounds thirty three pence" → £5.33 将数字后面的单词“rupees”、“Indian rupees”或“INR”转换卢比符号 (#) 加在数字之前。 • "twenty three rupees" → #23 • "fifty rupees thirty paise" → #50.30	将 “欧元” 一词转换为欧元符号。 • "ein euro" → 1 € • "ein Euro vierzig" → 1,40 € • "ein Euro vierzig Cent" → 1,40 €

规则	英语方言 （输入音频 → 输出文本）	德语方言 （输入音频 → 输出文本）
将时间转换成数字。	• "seven a m eastern standard time" → 7 a.m. eastern standard time • "twelve thirty p m" → 12:30 p.m.	• "vierzehn Uhr fünfzehn" → 14:15 Uhr
将日期转换成数字。	• "May fifth twenty twelve" → May 5th 2012 • "May five twenty twelve" → May 5 2012 • "five May twenty twelve" → 5 May 2012	• "dritter Dezember neunzehn hundert sechundfünfzig" → 3. Dezember 1956
用 “到” 一词分隔数字的跨度。	• "twenty three to thirty seven" → 23 to 37	不适用
年份以四位数表示；这仅对 20、21 和 22 世纪的年份有效。	• "nineteen sixty two" → 1962 • “the year is twenty twelve” → 年份是2012 • "twenty nineteen" → 2019 • "twenty one thirty" → 2130	不适用
显示斜线和短划线。	• "fifty-five dash thirteen" → 55-13 不显示斜杠。 • "fifty-five slash thirteen" → 55 slash 13	• "fünfundfünfzig Schrägstrich dreizehn" → 55/13 • "fünfundfünfzig Strich dreizehn" → 55-13

规则	英语方言 （输入音频 → 输出文本）	德语方言 （输入音频 → 输出文本）
显示带编号的段落。	未使用段落符号 (§) 显示已编号的段落。 • "paragraph seventeen" → paragraph 17	• "Paragraf siebzehn" → § 17

Amazon Transcribe 入门

在创建转录之前，您需要满足一些先决条件：

- 注册AWS 账户
- 安装AWS CLI和 SDK（如果您使用的是AWS Management Console进行转录，则可以跳过此步骤）
- 配置IAM凭证
- 设置存储Amazon S3桶
- 创建IAM策略

当您完成这些先决条件后，您可以进行转录。从以下列表中选择您的首选转录方法开始。

- AWS CLI
- AWS Management Console
- AWS 开发工具包
- HTTP
- WebSockets

> **Tip**
>
> 如果您不熟悉Amazon Transcribe或想探索我们的功能，我们建议您使用AWS Management Console。如果你想使用电脑麦克风开始直播，这也是最简单的选择。

由于使用 HTTP/2 进行流式传输 WebSockets 并且比其他转录方法更复杂，因此我们建议在开始使用这些方法之前先阅读设置流式转录本节。请注意，我们强烈建议使用 SDK 进行流媒体转录。

注册AWS 账户

您可以注册免费套餐账户或付费账户。这两个选项都允许您访问所有内容AWS 服务。免费套餐有试用期，在此期间，您可以浏览AWS 服务和估算使用量。试用期到期后，您可以迁移到付费账户。费用是pay-as-you-use 按一定比例累积的；有关详细信息，请参阅Amazon Transcribe定价。

> **Tip**
>
> 设置账户时，记下你的AWS 账户 ID，因为你需要它来创建IAM实体。

安装AWS CLI和 SDK

要使用Amazon Transcribe API，必须先安装AWS CLI。当前AWS CLI是版本 2。你可以在《AWS Command Line Interface用户指南》中找到 Linux、Mac、Windows 和 Docker 的安装说明。

AWS CLI安装完成后，必须将其配置为安全凭证和AWS 区域。

如果您想与 SDKAmazon Transcribe 一起使用，请选择首选语言以获取安装说明：

- .NET
- C++
- Go
- Java V2
- JavaScript
- PHP V3
- AWS SDK for Python (Boto3)（批量转录）
- Python（直播转录）
- Ruby V3
- Rust（批量转录）
- Rust（流媒体转录）

配置IAM凭证

当您创建时AWS 账户，最初使用的是一个对您账户中所有和资源拥有完全AWS访问权限的登录身份。此身份称为AWS 账户根用户，可使用您创建账户时所用的电子邮件地址和密码登录来访问。

强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。

作为最佳实践，要求用户（包括需要管理员访问权限的用户）结合使用联合身份验证和身份提供商的联合身份验证才能使用临时凭证访问AWS。

联合身份验证是使用通过身份源提供的凭证来访问AWS服务的任何用户。当联合身份访问 AWS 账户时，他们代入角色，而角色提供临时凭证。

要集中管理访问权限，我们建议您使用 AWS IAM Identity Center。您可以在中创建用户和组IAM Identity Center。您也可以连接并同步到您自己的身份源中的一组用户和组以跨所有AWS 账户和应用程序使用。有关更多信息，请参阅适用于 Amazon Transcribe 的 Identity and Access Management：

要了解有关IAM最佳实践的更多信息，请参阅中的安全最佳实践IAM。

创建Amazon S3存储桶

Amazon S3是一项安全的对象存储服务。 Amazon S3将您的文件（称为对象）存储在容器（称为存储桶）中。

要运行批量转录，您必须先将媒体文件上传到Amazon S3存储桶中。如果您没有为转录输出指定Amazon S3存储桶，请Amazon Transcribe将您的脚本放在临时AWS托管的Amazon S3存储桶中。90天后，AWS托管存储段中的转录输出将自动删除。

了解如何创建您的第一个 S3 存储桶并将对象上传到您的存储桶。

创建 IAM 策略

要管理访问权限AWS，您必须创建策略并将其附加到IAM身份（用户、组或角色）或AWS资源以创建策略定义了它所连接的实体的权限。例如，只有当您为该角色附加了授予其访问权限的策略时，该角色才能访问Amazon S3存储段中的媒体文件。如果您想进一步限制该角色，可以改为限制其对Amazon S3存储段内特定文件的访问权限。

要了解有关使用AWS策略的更多信息，请参阅：

- 中的策略和权限IAM
- 创建IAM策略
- Amazon Transcribe 如何与 IAM 协同工作

有关您可以使用的策略示例Amazon Transcribe，请参阅Amazon Transcribe 基于身份的策略示例。如果要生成自定义策略，请考虑使用AWS策略生成器。

您可以使用AWS Management Console、AWS CLI或AWS SDK 添加策略。有关说明，请参阅添加和删除IAM身份权限。

策略的格式为：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "my-policy-name",
            "Effect": "Allow",
            "Action": [
                "service:action"
            ],
            "Resource": [
                "amazon-resource-name"
            ]
        }
    ]
}
```

Amazon Resource Name（ARN）唯一标识所有AWSAmazon S3资源。您可以在策略中使用 ARN 为使用特定资源的特定操作授予权限。例如，如果您想授予对Amazon S3存储段及其子文件夹的读取权限，可以将以下代码添加到信任策略的Statement部分中：

```
{
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ]
}
```

以下是向Amazon S3存储段及其子文件夹授予Amazon Transcribe读取 (GetObject,ListBucketPutObject) 和 write () 权限的示例策略：DOC-EXAMPLE-BUCKET

```
{
  "Version": "2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
             "Effect": "Allow",
             "Action": [
                 "s3:PutObject"
             ],
             "Resource": [
                 "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                 "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
             ]
        }
   ]
}
```

使用进行转录AWS Management Console

您可以使用AWS控制台进行批处理和流式转录。如果您正在转录Amazon S3存储段中的媒体文件，则是在执行批量转录。如果您正在转录实时音频数据流，则是在执行流式转录。

在开始批量转录之前，您必须先将媒体文件上传到Amazon S3存储桶。要使用流式传输转录AWS Management Console，必须使用计算机麦克风。

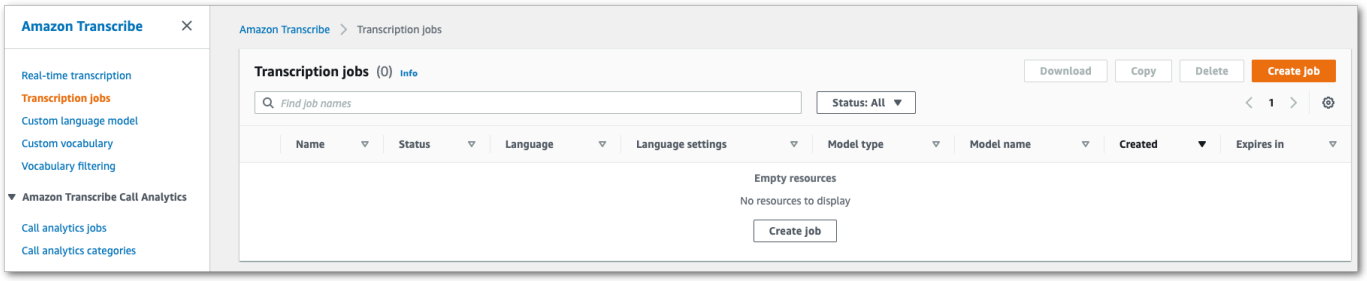
要查看支持的媒体格式和其他媒体要求和限制，请参阅数据输入和输出。

扩展以下部分，简要介绍每种转录方法。

Batch 转录

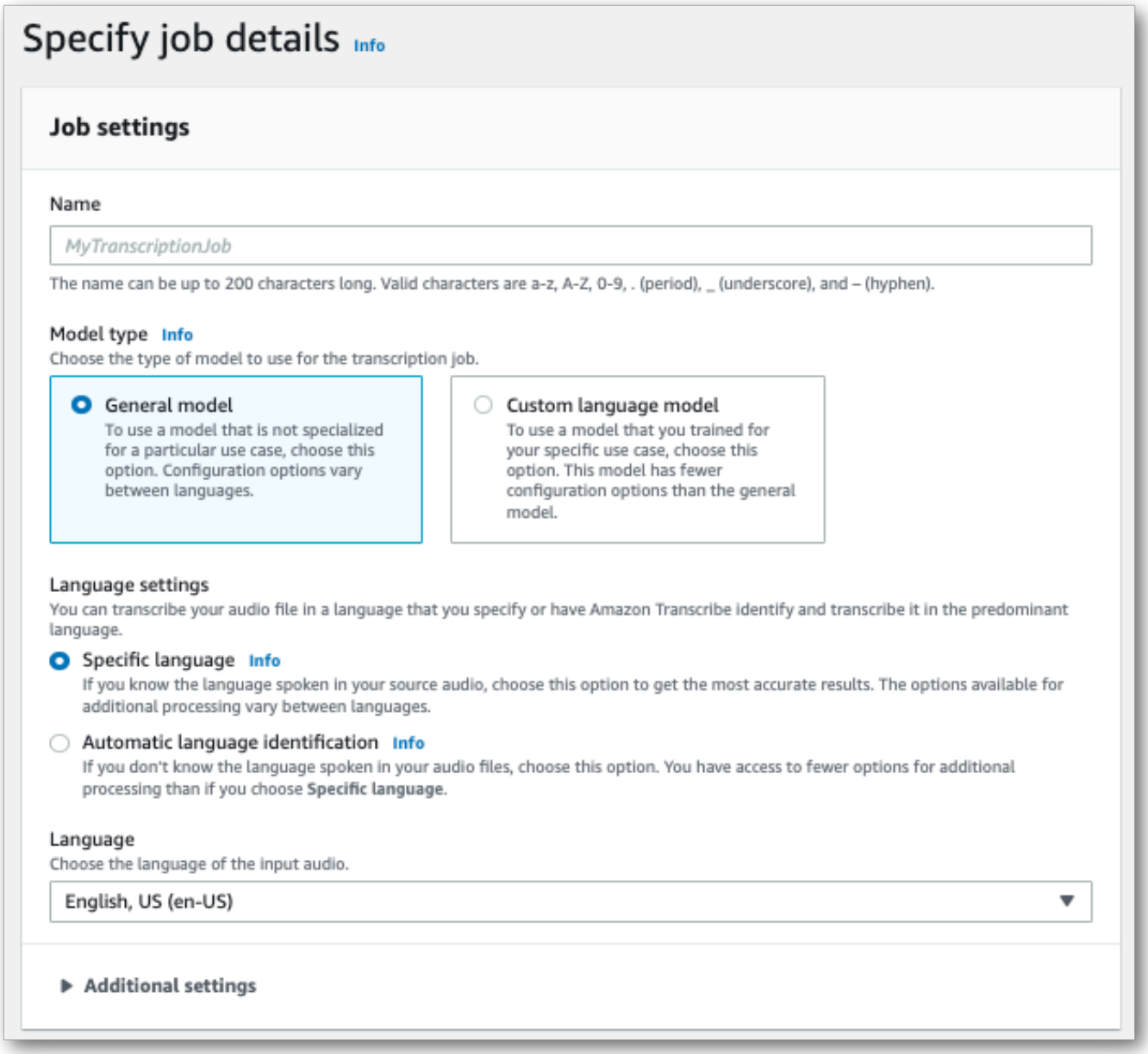
首先，请确保您已将要转录的媒体文件上传到Amazon S3存储桶中。如果您不确定如何执行此操作，请参阅Amazon S3用户指南：将对象上传到您的存储桶。

1. 从左AWS Management Console侧导航窗格中选择转录作业。此操作带您进入转录任务列表。



选择创建作业。

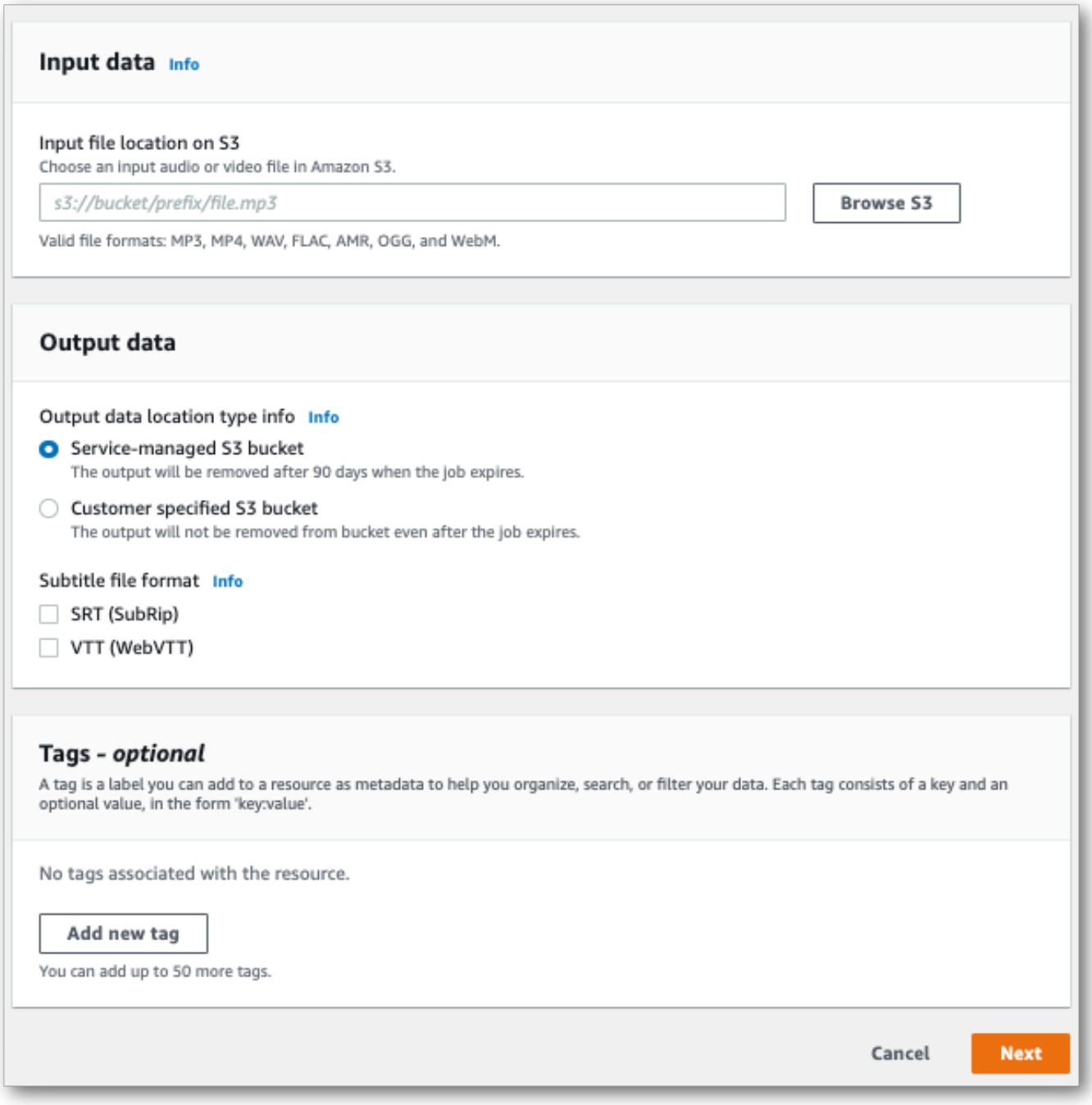
2. 填写 “指定作业详细信息” 页面上的字段。



输入位置必须是Amazon S3存储段内的对象。对于输出位置，您可以选择安全的Amazon S3服务管理存储分区，也可以指定自己的Amazon S3存储桶。

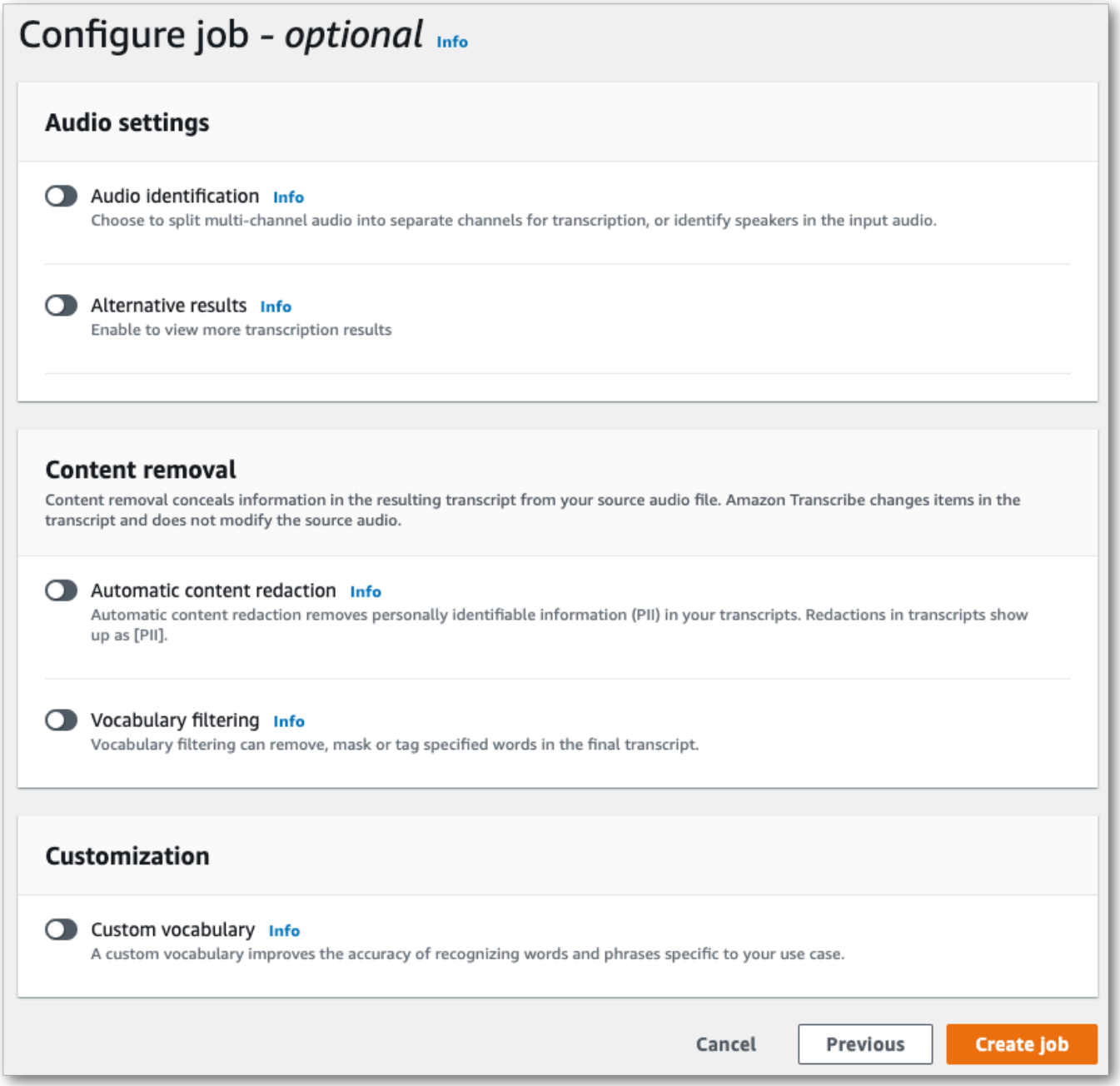
如果您选择服务管理的存储桶，则可以在中查看脚本预览AWS Management Console，也可以从任务详细信息页面下载脚本（见下文）。

如果您选择自己的Amazon S3存储桶，则无法在中看到预览AWS Management Console，必须前往Amazon S3存储桶下载您的脚本。



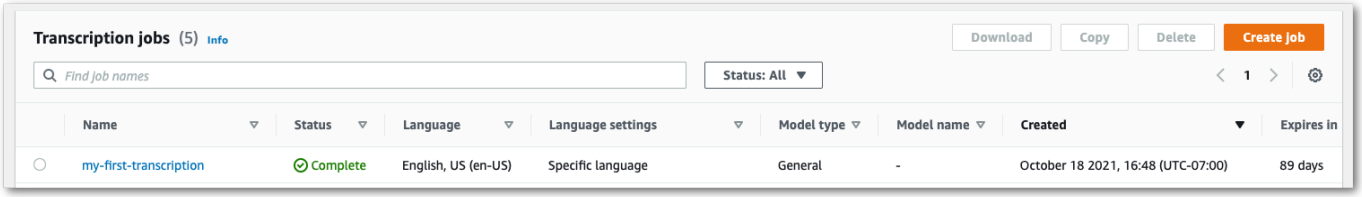
选择 Next（下一步）。

3. 在“配置作业”页面上选择任何所需的选项。如果您想使用自定义词汇或自定义语言模型与转录一起使用，则必须在开始转录工作之前创建它们。



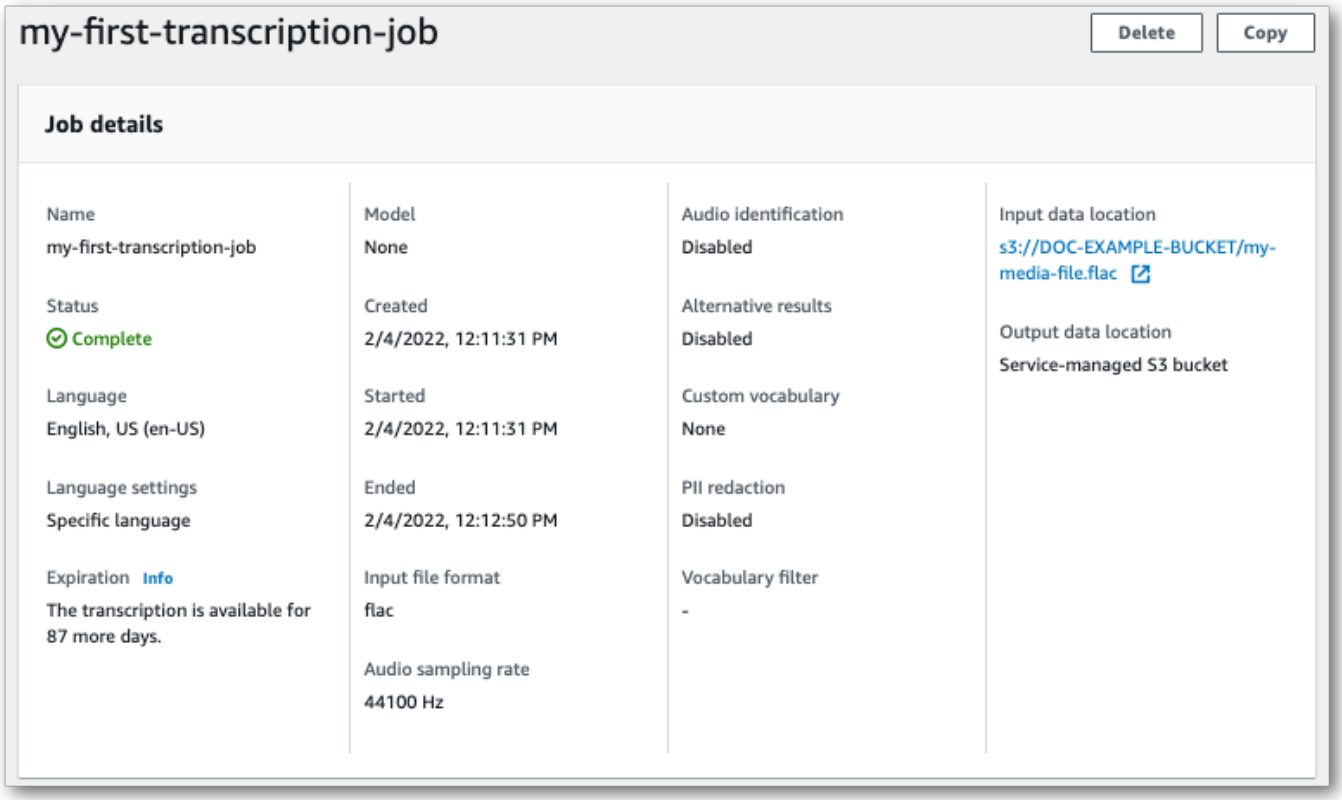
选择创建作业。

4. 您现在进入转录职位页面。此操作带您进入转录任务状态。完成后，选择您的转录。



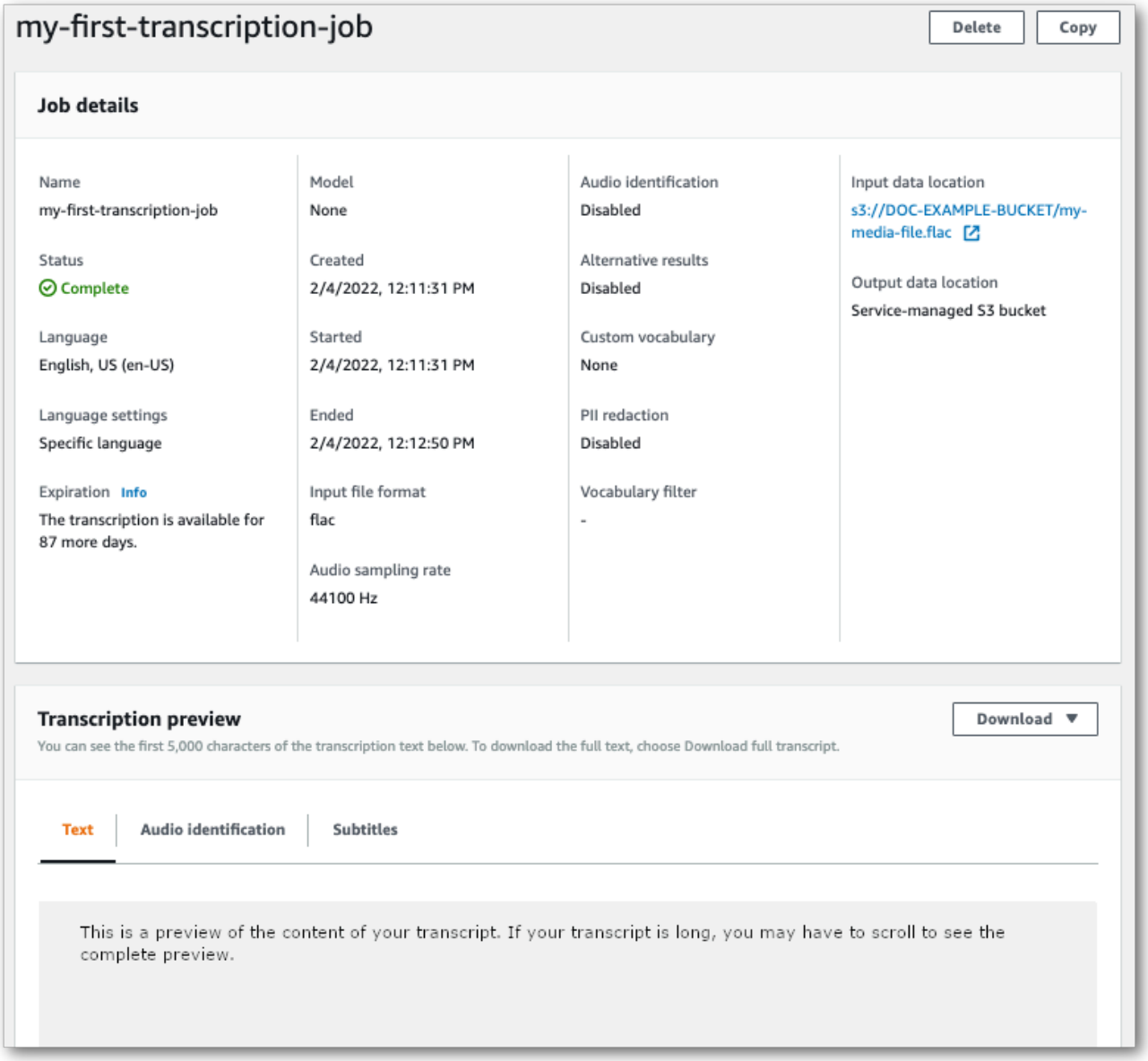
5. 您现在正在查看 Job 详情页面进行转录。在这里，您可以查看在设置转录任务时指定的所有选项。

   要查看您的脚本，请在输出数据位置下方的右列中选择链接的文件路径。这将带您进入指定的Amazon S3输出文件夹。选择您的输出文件，该文件现在的扩展名为.json。



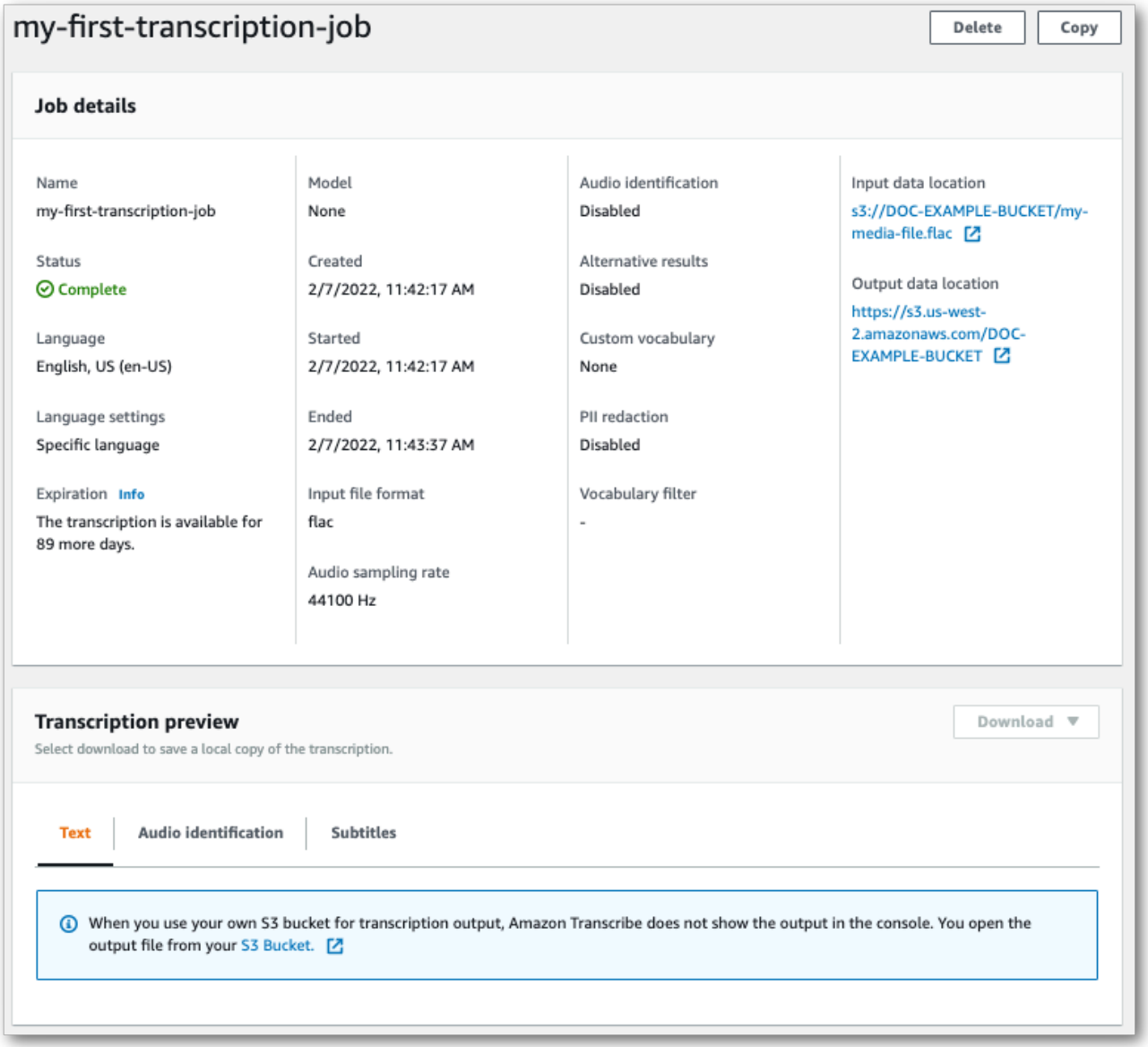
6. 下载脚本的方式取决于您选择的是服务管理Amazon S3存储分区还是自己的Amazon S3存储桶。

   a. 如果您选择服务管理的存储桶，则可以在转录作业的信息页面上看到转录预览窗格以及下载按钮。



选择 “下载”，然后选择 “下载脚本”。

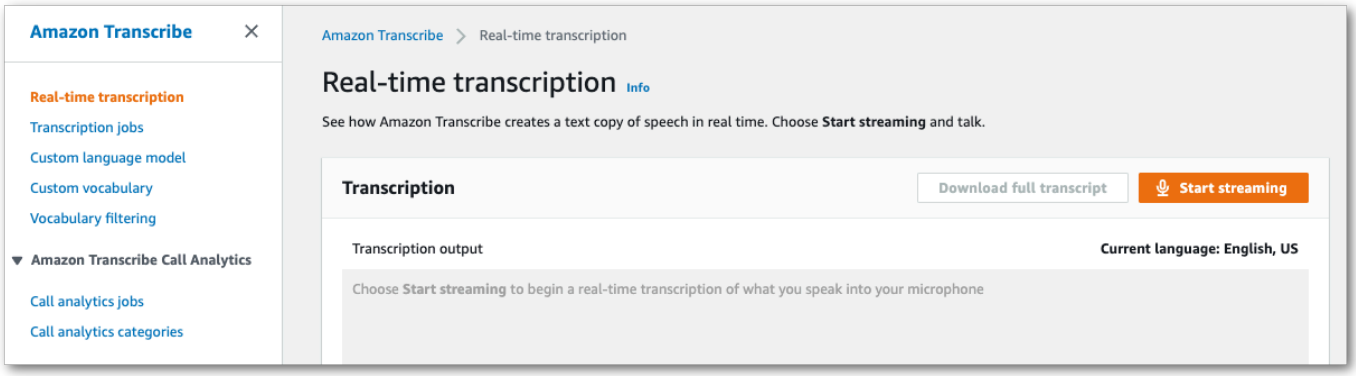
b. 如果您选择了自己的Amazon S3存储桶，则在转录作业信息页面的转录预览窗格中看不到任何文本。相反，你会看到一个蓝色的信息框，上面有指向你选择的Amazon S3存储桶的链接。



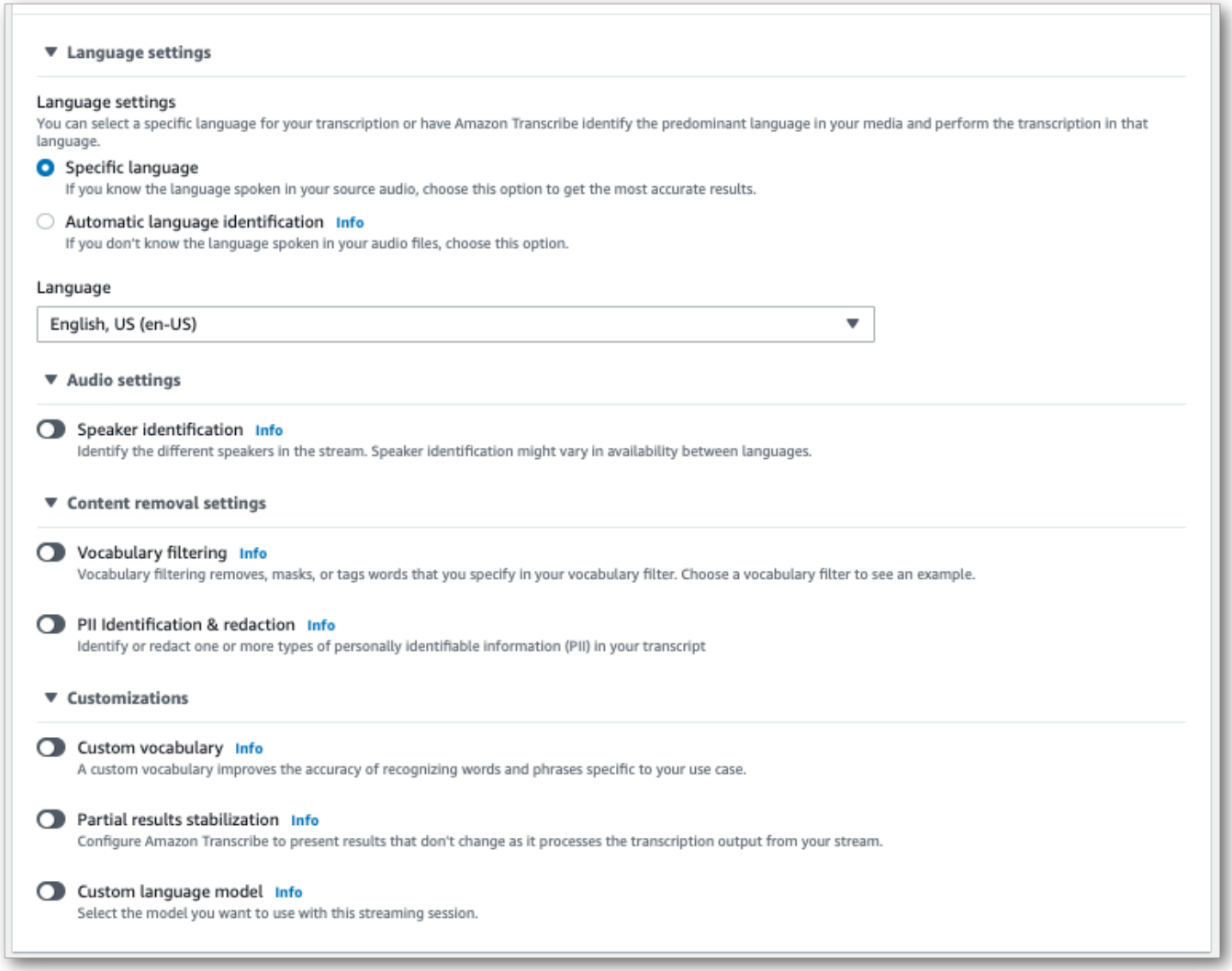
要访问您的脚本，请使用 Job 详细信息窗格中输出数据位置下的链接或转录预览窗格中蓝色信息框中的 S3 存储桶链接转到指定的Amazon S3存储桶。

直播转录

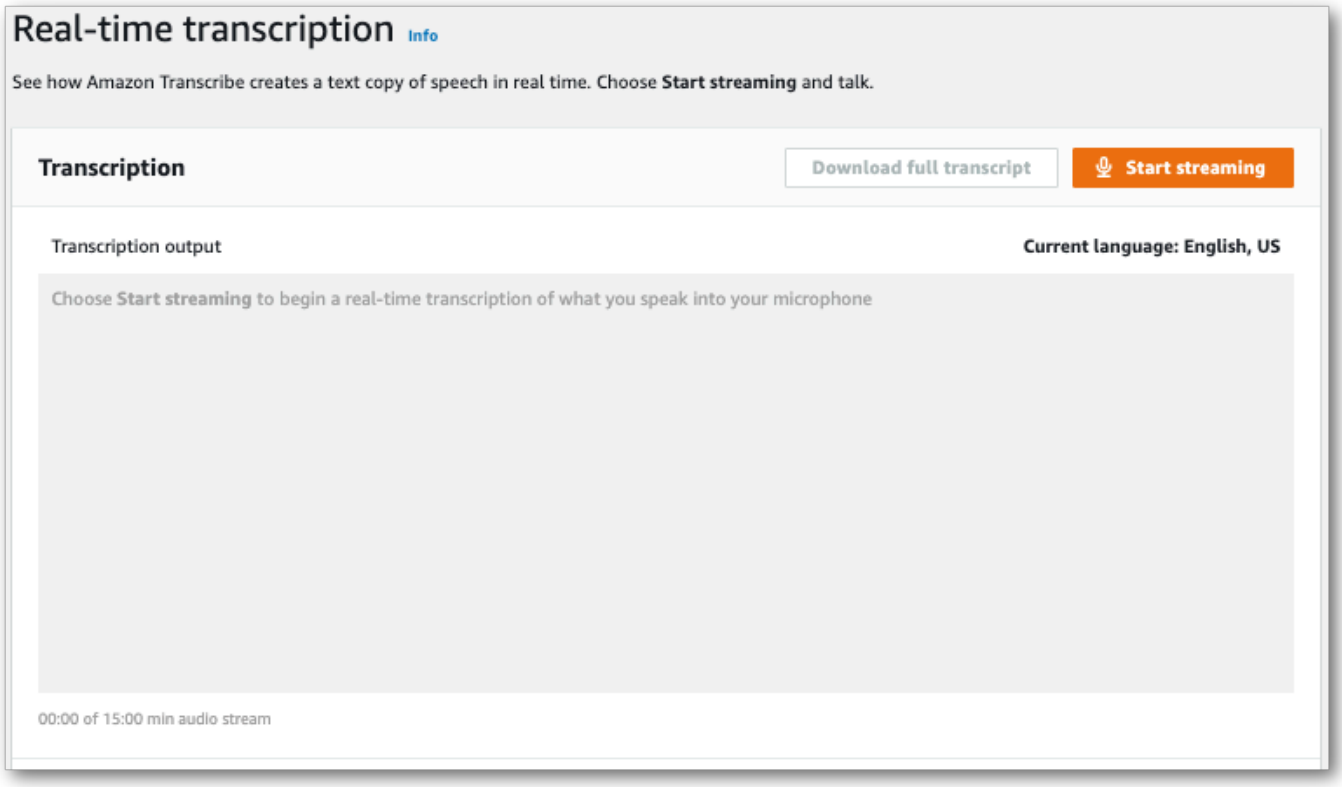
1. 从中AWS Management Console选择左侧导航窗格中的实时转录。随后您将转至主直播页面，您可以在其中选择选项，随后您将转至开始直播。



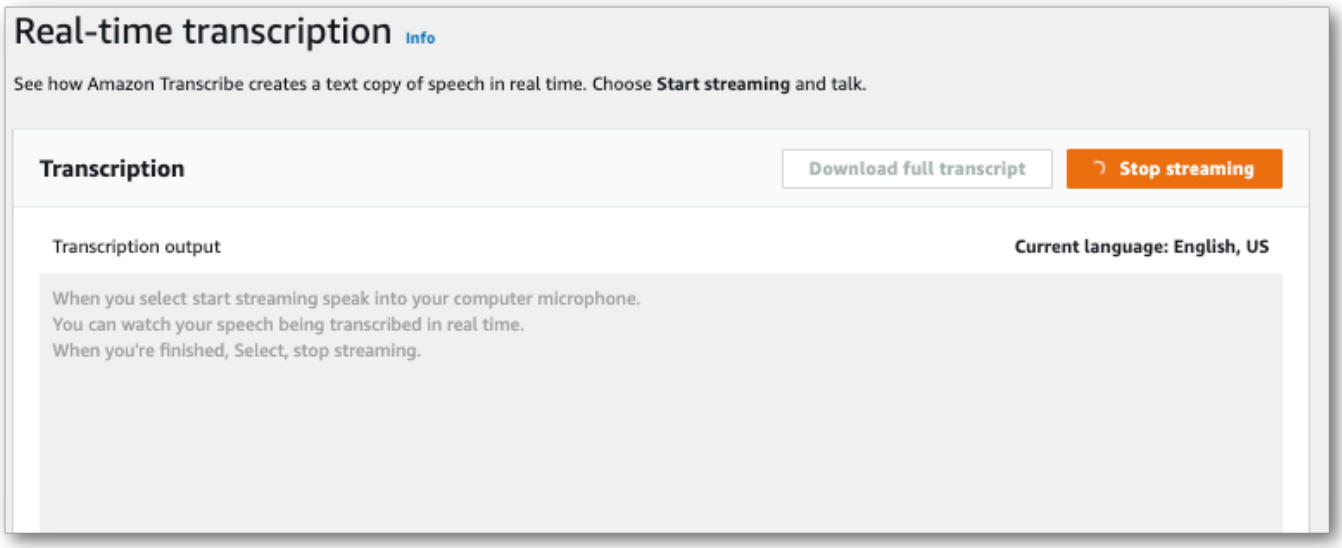
2. 在转录输出框下方，您可以选择各种语言和音频设置。



3. 选择适当的设置后，滚动到页面顶部并选择 “开始直播”，然后开始对着电脑麦克风讲话。您可以实时看到您的语音转录。



4. 完成后，选择 Stope (停止直播)。



现在，您可以通过选择 “下载完整脚本” 来下载您的成绩单。

使用AWS CLI

使用开始AWS CLI转录时，可以在 CLI 级别运行所有命令。或者，您可以运行要使用的命令，然后运行包含请求正文的 JSON 文件的AWS 区域和位置。本指南中的示例显示了这两种方法；但是，本节重点介绍前一种方法。

AWS CLI不支持流媒体转录。

在继续之前，请确保您已经：

- 将您的媒体文件上传到Amazon S3存储桶中。如果您不确定如何创建Amazon S3存储桶或上传文件，请参阅创建您的第一个Amazon S3存储桶和将对象上传到您的存储桶。
- 安装了AWS CLI。

您可以在 “AWS CLI命令参考”Amazon Transcribe 中找到的所有AWS CLI命令。

开始新任务

要开始新的转录，请使用`start-transcription-job`命令。

1. 在终端中，键入以下内容：

   ```
   aws transcribe start-transcription-job \
   ```

   “>” 出现在下一行，您现在可以继续添加必需的参数，如下一步所述。

   您也可以省略 '\' 并追加所有参数，用空格分隔每个参数。

2. 在`start-transcription-job`命令中，必须包括`region`、`transcription-job-name`、`media`、和`language-code`或`identify-language`。

   如果要指定输出位置，请在请求`output-bucket-name`中包括；如果要指定指定输出存储桶的子文件夹，也请包括`output-key`。

   ```
   aws transcribe start-transcription-job \
    --region us-west-2 \
    --transcription-job-name my-first-transcription-job \
    --media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
    --language-code en-US
   ```

如果附加所有参数，则此请求如下所示：

```
aws transcribe start-transcription-job --region us-west-2 --transcription-job-
name my-first-transcription-job --media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-
input-files/my-media-file.flac --language-code en-US
```

如果您选择不使用指定输出存储桶output-bucket-name，则将转录输出Amazon Transcribe放在服务管理的存储分区中。存储在服务管理存储段中的脚本将在 90 天后过期。

Amazon Transcribe回应为：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "my-first-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "en-US",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-
file.flac"
        },
        "StartTime": "2022-03-07T15:03:44.246000-08:00",
        "CreationTime": "2022-03-07T15:03:44.229000-08:00"
    }
}
```

如果从TranscriptionJobStatusIN_PROGRESS更改为，则转录作业成功COMPLETED。要查看更新的内容 TranscriptionJobStatus，请使用get-transcription-job或list-transcription-job命令，如下一节所示。

获取转录作业的状态

要获取有关您的转录作业的信息，请使用get-transcription-job命令。

此命令的唯一必需参数是作业所在的位置和作业的名称。AWS 区域

```
aws transcribe get-transcription-job \
 --region us-west-2 \
 --transcription-job-name my-first-transcription-job
```

Amazon Transcribe回应为：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "my-first-transcription-job",
        "TranscriptionJobStatus": "COMPLETED",
        "LanguageCode": "en-US",
        "MediaSampleRateHertz": 48000,
        "MediaFormat": "flac",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
        },
        "Transcript": {
            "TranscriptFileUri": "https://s3.the-URI-where-your-job-is-located.json"
        },
        "StartTime": "2022-03-07T15:03:44.246000-08:00",
        "CreationTime": "2022-03-07T15:03:44.229000-08:00",
        "CompletionTime": "2022-03-07T15:04:01.158000-08:00",
        "Settings": {
            "ChannelIdentification": false,
            "ShowAlternatives": false
        }
    }
}
```

如果您为转录输出选择了自己的Amazon S3存储桶，则此存储分区将随一起列出`TranscriptFileUri`。如果您选择了服务管理的存储桶，则会提供一个临时 URI；使用此 URI 下载您的脚本。

> **Note**
>
> 服务管理Amazon S3存储分区的临时 URI 仅在 15 分钟内有效。如果您在使用 URI 时`AccesDenied`遇到错误，`get-transcription-job`请再次运行请求以获取新的临时 URI。

列出你的转录工作

要列出给定内容中的所有转录任务AWS 区域，请使用`list-transcription-jobs`命令。

此命令的唯一必需参数是AWS 区域转录任务所在的参数。

```
aws transcribe list-transcription-jobs \
 --region us-west-2
```

Amazon Transcribe回应为：

```
{
    "NextToken": "A-very-long-string",
    "TranscriptionJobSummaries": [
        {
            "TranscriptionJobName": "my-first-transcription-job",
            "CreationTime": "2022-03-07T15:03:44.229000-08:00",
            "StartTime": "2022-03-07T15:03:44.246000-08:00",
            "CompletionTime": "2022-03-07T15:04:01.158000-08:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        }
    ]
}
```

删除您的转录作业

要删除您的转录作业，请使用`delete-transcription-job`命令。

此命令的唯一必需参数是作业所在的位置和作业的名称。AWS 区域

```
aws transcribe delete-transcription-job \
 --region us-west-2 \
 --transcription-job-name my-first-transcription-job
```

要确认您的删除请求成功，可以运行该`list-transcription-jobs`命令。您的任务应显示在列表中。

使用 AWS SDK 进行转录

不论是批量转录还是流式转录，您都可以使用 SDK。如果您要转录 Amazon S3 存储桶中的媒体文件，则将执行批量转录。如果您要转录音频数据的实时流，则将执行流式转录。

有关可与 Amazon Transcribe 配合使用的编程语言的列表，请参阅支持的编程语言。请注意，并非所有 AWS SDK 都支持流式转录。要查看支持的媒体格式以及其它媒体要求和限制，请参阅数据输入和输出。

有关所有可用的 AWS SDK 和构建器工具的更多信息，请参阅在 AWS 上构建所需的工具。

> **Tip**
>
> 有关使用 AWS SDK 的其它示例，包括特定特征、场景和跨服务示例，请参阅使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS一章。
> 另外，您还可以在以下 GitHub 存储库中找到 SDK 代码示例：
>
> - AWS 代码示例
> - Amazon Transcribe 示例

批量转录

您可以使用 Amazon S3 存储桶中的媒体文件的 URI 创建批量转录。如果您不确定如何创建 Amazon S3 存储桶或上传文件，请参阅创建您的第一个 S3 存储桶和将对象上传到您的存储桶。

Java

```
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.transcribe.TranscribeClient;
import software.amazon.awssdk.services.transcribe.model.*;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;

public class TranscribeDemoApp {
    private static final Region REGION = Region.US_WEST_2;
    private static TranscribeClient client;

    public static void main(String args[]) {

        client = TranscribeClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        String transcriptionJobName = "my-first-transcription-job";
        String mediaType = "flac"; // can be other types
        Media myMedia = Media.builder()
                .mediaFileUri("s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-
file.flac")
```

```
                .build();

        String outputS3BucketName = "s3://DOC-EXAMPLE-BUCKET";
        // Create the transcription job request
        StartTranscriptionJobRequest request =
 StartTranscriptionJobRequest.builder()
                .transcriptionJobName(transcriptionJobName)
                .languageCode(LanguageCode.EN_US.toString())
                .mediaSampleRateHertz(16000)
                .mediaFormat(mediaType)
                .media(myMedia)
                .outputBucketName(outputS3BucketName)
                .build();

        // send the request to start the transcription job
        StartTranscriptionJobResponse startJobResponse =
 client.startTranscriptionJob(request);

        System.out.println("Created the transcription job");
        System.out.println(startJobResponse.transcriptionJob());

        // Create the get job request
        GetTranscriptionJobRequest getJobRequest =
 GetTranscriptionJobRequest.builder()
                .transcriptionJobName(transcriptionJobName)
                .build();

        // send the request to get the transcription job including the job status
        GetTranscriptionJobResponse getJobResponse =
 client.getTranscriptionJob(getJobRequest);

        System.out.println("Get the transcription job request");
        System.out.println(getJobResponse.transcriptionJob());
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

}
```

JavaScript

```
const { TranscribeClient, StartTranscriptionJobCommand } = require("@aws-sdk/client-
transcribe"); // CommonJS import

const region = "us-west-2";
const credentials = {
  "accessKeyId": "",
  "secretAccessKey": "",
};

const input = {
  TranscriptionJobName: "my-first-transcription-job",
  LanguageCode: "en-US",
  Media: {
    MediaFileUri: "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  OutputBucketName: "DOC-EXAMPLE-BUCKET",
};

async function startTranscriptionRequest() {
  const transcribeConfig = {
    region,
    credentials
  };
  const transcribeClient = new TranscribeClient(transcribeConfig);
  const transcribeCommand = new StartTranscriptionJobCommand(input);
  try {
    const transcribeResponse = await transcribeClient.send(transcribeCommand);
    console.log("Transcription job created, the details:");
    console.log(transcribeResponse.TranscriptionJob);
  } catch(err) {
    console.log(err);
  }
}

startTranscriptionRequest();
```

Python

```
import time
import boto3
```

```
def transcribe_file(job_name, file_uri, transcribe_client):
    transcribe_client.start_transcription_job(
        TranscriptionJobName = job_name,
        Media = {
            'MediaFileUri': file_uri
        },
        MediaFormat = 'flac',
        LanguageCode = 'en-US'
    )

    max_tries = 60
    while max_tries > 0:
        max_tries -= 1
        job = transcribe_client.get_transcription_job(TranscriptionJobName =
 job_name)
        job_status = job['TranscriptionJob']['TranscriptionJobStatus']
        if job_status in ['COMPLETED', 'FAILED']:
            print(f"Job {job_name} is {job_status}.")
            if job_status == 'COMPLETED':
                print(
                    f"Download the transcript from\n"
                    f"\t{job['TranscriptionJob']['Transcript']
['TranscriptFileUri']}.")
            break
        else:
            print(f"Waiting for {job_name}. Current status is {job_status}.")
        time.sleep(10)


def main():
    transcribe_client = boto3.client('transcribe', region_name = 'us-west-2')
    file_uri = 's3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac'
    transcribe_file('Example-job', file_uri, transcribe_client)


if __name__ == '__main__':
    main()
```

流式转录

您可以使用流媒体文件或实时媒体流来创建流式转录。

请注意，Amazon Transcribe 流式转录不支持 AWS SDK for Python (Boto3) 标准。要使用 Python 启动流式转录，请使用此适用于 Amazon Transcribe 的异步 Python SDK。

Java

以下示例是一个转录流音频的 Java 程序。

要运行此示例，请注意以下条件：

- 您必须使用适用于 Java 2.x 的 AWS SDK。
- 客户端必须使用 Java 1.8 才能与适用于 Java 2.x 的 AWS SDK兼容。
- 您指定的采样率必须与音频流的实际采样率相匹配。

另请参阅：重试用于 Amazon Transcribe 流式转录的客户端 (Java SDK)。该代码管理与 Amazon Transcribe 的连接，并在连接出错时重试发送数据。例如，如果网络出现临时错误，则此客户端将重新发送失败的请求。

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_WEST_2;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws URISyntaxException,
 ExecutionException, InterruptedException, LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16,000 Hz, 16 bit sample size, mono
        int sampleRate = 16000;
```

```
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
```

```
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private InputStream getStreamFromFile(String myMediaFileName) {
        try {
            File inputFile = new
 File(getClass().getClassLoader().getResource(myMediaFileName).getFile());
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;


        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
```

```
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
```

```
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

JavaScript

```
const {
  TranscribeStreamingClient,
  StartStreamTranscriptionCommand,
} = require("@aws-sdk/client-transcribe-streaming");
const { createReadStream } = require("fs");
const { join } = require("path");

const audio = createReadStream(join(__dirname, "my-media-file.flac"),
 { highWaterMark: 1024 * 16});
```

```
const LanguageCode = "en-US";
const MediaEncoding = "pcm";
const MediaSampleRateHertz = "16000";
const credentials = {
  "accessKeyId": "",
  "secretAccessKey": "",
};
async function startRequest() {
  const client = new TranscribeStreamingClient({
    region: "us-west-2",
    credentials
  });

  const params = {
    LanguageCode,
    MediaEncoding,
    MediaSampleRateHertz,
    AudioStream: (async function* () {
      for await (const chunk of audio) {
        yield {AudioEvent: {AudioChunk: chunk}};
      }
    })(),
  };
  const command = new StartStreamTranscriptionCommand(params);
  // Send transcription request
  const response = await client.send(command);
  // Start to print response
  try {
    for await (const event of response.TranscriptResultStream) {
      console.log(JSON.stringify(event));
    }
  } catch(err) {
    console.log("error")
    console.log(err)
  }
}
startRequest();
```

Python

以下示例是一个转录流音频的 Python 程序。

要运行此示例，请注意以下条件：

- 您必须使用这个适用于 Python 的 SDK。
- 您指定的采样率必须与音频流的实际采样率相匹配。

```
import asyncio
# This example uses aiofile for asynchronous file reads.
# It's not a dependency of the project but can be installed
# with `pip install aiofile`.
import aiofile

from amazon_transcribe.client import TranscribeStreamingClient
from amazon_transcribe.handlers import TranscriptResultStreamHandler
from amazon_transcribe.model import TranscriptEvent

"""
Here's an example of a custom event handler you can extend to
process the returned transcription results as needed. This
handler will simply print the text out to your interpreter.
"""
class MyEventHandler(TranscriptResultStreamHandler):
    async def handle_transcript_event(self, transcript_event: TranscriptEvent):
        # This handler can be implemented to handle transcriptions as needed.
        # Here's an example to get started.
        results = transcript_event.transcript.results
        for result in results:
            for alt in result.alternatives:
                print(alt.transcript)


async def basic_transcribe():
    # Set up our client with your chosen Region
    client = TranscribeStreamingClient(region = "us-west-2")

    # Start transcription to generate async stream
    stream = await client.start_stream_transcription(
        language_code = "en-US",
        media_sample_rate_hz = 16000,
        media_encoding = "pcm",
    )

    async def write_chunks():
        # NOTE: For pre-recorded files longer than 5 minutes, the sent audio
        # chunks should be rate limited to match the real-time bitrate of the
```

```
        # audio stream to avoid signing issues.
        async with aiofile.AIOFile('filepath/my-media-file.flac', 'rb') as afp:
            reader = aiofile.Reader(afp, chunk_size = 1024 * 16)
            async for chunk in reader:
                await stream.input_stream.send_audio_event(audio_chunk = chunk)
        await stream.input_stream.end_stream()

    # Instantiate our handler and start processing events
    handler = MyEventHandler(stream.output_stream)
    await asyncio.gather(write_chunks(), handler.handle_events())

loop = asyncio.get_event_loop()
loop.run_until_complete(basic_transcribe())
loop.close()
```

C++

有关流式转录 C++ SDK 示例的信息，请参阅代码示例章节。

将此服务与 AWS SDK 结合使用

AWS 软件开发工具包 (SDK) 适用于许多常用编程语言。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。

软件开发工具包文档	代码示例
AWS SDK for C++	AWS SDK for C++ 代码示例
AWS SDK for Go	AWS SDK for Go 代码示例
AWS SDK for Java	AWS SDK for Java 代码示例
AWS SDK for JavaScript	AWS SDK for JavaScript 代码示例
AWS SDK for Kotlin	AWS SDK for Kotlin 代码示例
AWS SDK for .NET	AWS SDK for .NET 代码示例
AWS SDK for PHP	AWS SDK for PHP 代码示例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 代码示例

软件开发工具包文档	代码示例
AWS SDK for Ruby	AWS SDK for Ruby 代码示例
AWS SDK for Rust	AWS SDK for Rust 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
AWS SDK for Swift	AWS SDK for Swift 代码示例

有关特定于此服务的示例，请参阅使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS。

> **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。

使用 HTTP 进行转录或 WebSockets

Amazon Transcribe支持 HTTP 进行批处理 (HTTP/1.1) 和流式传输 (HTTP/2) 转录。 WebSockets 支持流式转录。

如果您正在转录Amazon S3存储段中的媒体文件，则是在执行批量转录。如果您正在转录实时音频数据流，则是在执行流式转录。

HTTP 和都 WebSockets 要求您使用AWS签名版本 4 标头对您的请求进行身份验证。有关更多信息，请参阅签名AWS API 请求。

Batch 转录

您可以使用以下标头进行批处理 HTTP 请求：

- host
- x-amz-target
- 内容类型
- x-amz-content-sha256
- x-amz-date
- 授权

示例如StartTranscriptionJob下：

```
POST /transcribe HTTP/1.1
host: transcribe.us-west-2.amazonaws.com
x-amz-target: com.amazonaws.transcribe.Transcribe.StartTranscriptionJob
content-type: application/x-amz-json-1.1
x-amz-content-sha256: string
x-amz-date: YYYYMMDDTHHMMSSZ
authorization: AWS4-HMAC-SHA256 Credential=access-key/YYYYMMSS/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string

{
    "TranscriptionJobName": "my-first-transcription-job",
    "LanguageCode": "en-US",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
    },
    "OutputBucketName": "DOC-EXAMPLE-BUCKET",
    "OutputKey": "my-output-files/"
}
```

API 参考中列出了其他操作和参数；所有AWS API 操作的通用参数列在通用参数部分中。其他签名元素详见AWS签名版本 4 请求的元素。

直播转录

使用 HTTP/2 进行流式传输转录 WebSockets ，并且比使用 SDK 更复杂。我们建议您在设置第一个直播之前查看该设置流式转录部分。

有关这些方法的更多信息，请参阅设置 HTTP/2 音频流或设置直 WebSocket 播。

> **Note**
>
> 我们强烈建议使用 SDK 进行流媒体转录。有关支持的 SDK 列表，请参阅支持的编程语言。

转录流式音频

使用 Amazon Transcribe 流媒体，您可以为媒体内容制作实时转录。与涉及上传媒体文件的批量转录不同，流媒体是实时传送到 Amazon Transcribe 的。 Amazon Transcribe 然后返回笔录，也是实时的。

流式媒体可以包括预先录制的媒体（电影、音乐和播客）和实时媒体（新闻直播）。常见的流媒体用例 Amazon Transcribe 包括体育赛事的直播隐藏式字幕和呼叫中心音频的实时监控。

流式媒体内容以一系列顺序数据包或“组块”的形式传送， Amazon Transcribe 即时转录这些内容。与批处理相比，使用流媒体的优势包括应用程序中的实时 speech-to-text 功能和更快的转录时间。但是，在某些情况下，这种速度的提高可能会影响准确性。

Amazon Transcribe 提供以下直播选项：

- SDK（首选）
- HTTP/2
- WebSockets
- AWS Management Console

要在中转录流式音频 AWS Management Console，请对着电脑麦克风说话。

> **Tip**
>
> 有关 SDK 代码示例，请参阅上的AWS 示例存储库 GitHub。

流式转录支持的音频格式有：

- FLAC
- Ogg 容器中的 Opus 编码音频
- PCM（仅带签名的 16 位小端音频格式，不包括 WAV）

建议使用无损格式（FLAC 或 PCM）。

> **Note**
>
> 并非所有语言都支持流式转录。有关详细信息，请参阅支持的语言表中的“数据输入”列。

要查看流媒体转录的 Amazon Transcribe 区域可用性，请参阅：Amazon Transcribe 终端节点和配额。

最佳实践

以下建议可提高流式转录的效率：

- 如果可能，请使用 PCM 编码的音频。
- 请确保您的音频流尽可能接近实时传输。
- 延迟取决于音频组块的大小。如果您能够使用音频类型（例如使用 PCM）指定组块大小，请将每个组块设置为 50 毫秒到 200 毫秒之间。您可以通过以下公式计算音频块大小：

```
chunk_size_in_bytes = chunk_duration_in_millisecond / 1000 * audio_sample_rate * 2
```

- 使用统一的组块大小。
- 确保正确指定了音频声道的数量。
- 对于单声道 PCM 音频，每个样本由两个字节组成，因此每个组块应由偶数字节组成。
- 对于双声道 PCM 音频，每个样本由四个字节组成，因此每个组块应是 4 字节的倍数。
- 当您的音频流不包含语音时，请编码并发送相同数量的无声音频。例如，PCM 的无声音频是一个零字节的音频流。
- 确保为音频指定正确的采样率。如果可能，请以 16000 Hz 的采样率进行录制；这在通过网络发送的质量和数据量之间做到了最佳折衷。请注意，大多数高端麦克风的录音频率为 44100 Hz 或 48000 Hz。

流式转录和部分结果

由于流媒体是实时运行的，因此成绩单是以部分结果生成的。 Amazon Transcribe 根据自然的语音片段（例如说话者的变化或音频的暂停）来分解传入的音频流。转录将以转录事件流的形式返回到您的应用程序，每个响应包含更多转录的语音，直到整个片段被转录。

以下代码块显示了其近似值。您可以通过登录 AWS Management Console、选择实时转录并对着麦克风说话来查看此过程的实际原理。边说话边观看转录输出窗格。

在此示例中，每行都是音频片段的部分结果。

```
The
The Amazon.
The Amazon is
The Amazon is the law.
The Amazon is the largest
The Amazon is the largest ray
The Amazon is the largest rain for
The Amazon is the largest rainforest.
The Amazon is the largest rainforest on the
The Amazon is the largest rainforest on the planet.
```

这些部分结果显示在 `Results` 对象内的转录输出中。此对象块中还有一个IsPartial字段。如果此字段为 true，则说明您的转录片段尚未完成。您可以在下面查看不完整片段和完整片段之间的区别：

```
"IsPartial": true (incomplete segment)

"Transcript": "The Amazon is the largest rainforest."

"EndTime": 4.545,
"IsPartial": true,
"ResultId": "12345a67-8bc9-0de1-2f34-a5b678c90d12",
"StartTime": 0.025


"IsPartial": false (complete segment)

"Transcript": "The Amazon is the largest rainforest on the planet."

"EndTime": 6.025,
"IsPartial": false,
"ResultId": "34567e89-0fa1-2bc3-4d56-78e90123456f",
"StartTime": 0.025
```

完整片段中的每个单词都有一个相关的置信度分数，该值介于 0 和 1 之间。值越大表示该单词被正确转录的可能性越大。

> **Tip**
>
> 音频片段的 `StartTime` 和 `EndTime` 可用于将转录输出与视频对话同步。

如果您运行的是需要低延迟的应用程序，则可能需要使用部分结果稳定功能。

部分结果稳定

Amazon Transcribe 一旦你开始直播音频，就会开始返回转录结果。它以增量方式返回这些部分结果，直到生成自然语音片段级别的最终结果。自然语音片段是连续的语音，其中包含暂停或说话者的变化。

Amazon Transcribe 继续输出部分结果，直到生成语音片段的最终转录结果。由于语音识别可能会随着上下文的增加而修改单词，因此每输出一个新的部分结果，流式转录都可能会略有变化。

此过程为每个语音片段提供了两个选项：

- 等待完成的片段
- 使用片段的部分结果

部分结果稳定 Amazon Transcribe 会改变为每个完整片段生成最终转录结果的方式。激活后，只有部分结果中的最后几个单词会改变。因此，转录准确性可能会受到影响。但是，与没有部分结果稳定的情况相比，返回转录的速度更快。在为视频添加字幕或为实时音频流生成字幕时，这种减少延迟可能会有所帮助。

以下示例演示在未激活部分结果稳定功能和激活部分结果稳定功能时如何处理相同的音频流。请注意，您可以将稳定性级别设置为低、中或高。低稳定性可提供最高的准确性。高稳定性转录速度更快，但准确性略低。

"Transcript":	"EndTime":	"IsPartial":
未启用部分结果稳定功能		

`The`
`The`
`The Amazon.`
`The Amazon is`
`The Amazon is the law.`
	`0.545`
	`1.045`
	`1.545`
	`2.045`
	`2.545`
	`3.045`
		`true`
		`true`
		`true`
		`true`
		`true`
		`true`

"Transcript":	"EndTime":	"IsPartial":
The Amazon is the largest The Amazon is the largest ray The Amazon is the largest rain for The Amazon is the largest rainforest. The Amazon is the largest rainforest on the The Amazon is the largest rainforest on the planet. The Amazon is the largest rainforest on the planet. The Amazon is the largest rainforest on the planet.	3.545 4.045 4.545 5.045 5.545 6.025 6.025	true true true true true true false

启用部分结果稳定功能（高稳定性）

"Transcript":	"EndTime":	"IsPartial":
The The The Amazon. The Amazon is The Amazon is the large The Amazon is the largest The Amazon is the largest rainfall. The Amazon is the largest rain forest. The Amazon is the largest rain forest on The Amazon is the largest rain forest on the planet. The Amazon is the largest rain forest on the planet. The Amazon is the largest rain forest on the planet. The Amazon is the largest rain forest on the planet. The Amazon is the largest rain forest on the planet.	0.515 1.015 1.515 2.015 2.515 3.015 3.515 4.015 4.515 5.015 5.515 6.015 6.335 6.335	true true true true true true true true true true true true true false

激活部分结果稳定功能时， Amazon Transcribe 使用Stable字段来指示项目是否稳定，其中 “项目” 是指转录的单词或标点符号。Stable 的值为 true 或 false。转录您的片段时，标记为 false（不稳定）的项目更有可能发生变化。相反，标记为 true（稳定）的项目则不会改变。

您可以选择呈现不稳定的单词，使字幕与语音保持一致。即使字幕随着上下文的增加而略有变化，用户体验也比定期文本突增更好，后者可能与语音一致，也可能不一致。

您也可以选择以不同的格式（例如斜体）显示不稳定的单词，以向查看者表明这些单词可能会发生变化。显示部分结果会限制在给定时间显示的文本数量。这在您应付空间限制时可能很重要，比如视频字幕。

> 通过 Machine Learn AWS ing 博客深入了解
>
> 要详细了解如何通过实时转录提高准确性，请参阅：
>
> - 通过稳定 Amazon Transcribe 部分结果来改善直播转录体验
> - “那是什么？” 使用 Amazon Transcribe 提高直播的字幕准确性

部分结果稳定示例输出

以下示例输出显示了未完成片段的 `Stable` 标志 (`"IsPartial": true`)。您可以看到“to”和“Amazon”这两个词并不稳定，因此在分段最终确定之前可能会发生变化。

```
"Transcript": {
    "Results": [
        {
            "Alternatives": [
                {
                    "Items": [
                        {
                            "Content": "Welcome",
                            "EndTime": 2.4225,
                            "Stable": true,
                            "StartTime": 1.65,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": "to",
                            "EndTime": 2.8325,
                            "Stable": false,
                            "StartTime": 2.4225,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": "Amazon",
                            "EndTime": 3.635,
```

```
                            "Stable": false,
                            "StartTime": 2.8325,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": ".",
                            "EndTime": 3.635,
                            "Stable": false,
                            "StartTime": 3.635,
                            "Type": "punctuation",
                            "VocabularyFilterMatch": false
                        }
                    ],
                    "Transcript": "Welcome to Amazon."
                }
            ],
            "EndTime": 4.165,
            "IsPartial": true,
            "ResultId": "12345a67-8bc9-0de1-2f34-a5b678c90d12",
            "StartTime": 1.65
        }
    ]
}
```

设置流式转录

本节展开讲了主要的流式转录部分。它旨在为想要使用 HTTP/2 或 WebSockets 直接设置直播而不是使用 SDK 设置直播的 AWS 用户提供信息。本节中的信息也可用于构建您自己的 SDK。

> **Important**
>
> 我们强烈建议使用软件开发工具包，而不是直接使用 HTTP/2。 WebSockets SDK 是转录数据流最简单、最可靠的方法。要开始使用 AWS SDK 进行流式传输，请参阅使用 AWS SDK 进行转录 。

设置 HTTP/2 音频流

用于流式传输转录请求的 HTTP/2 协议的关键组件是： Amazon Transcribe

- 标头帧。它包含您的请求的 HTTP/2 标头，以及授权标头中的签名，该签名 Amazon Transcribe 用作对数据帧进行签名的种子签名。
- 一个或多个消息帧采用事件流编码，其中包含元数据和原始音频字节。
- 结束帧。这是事件流编码中正文为空的已签名消息。

> **Note**
>
> Amazon Transcribe 每个 HTTP/2 会话仅支持一个数据流。如果您尝试使用多个音频流，则转录请求将失败。

1. 将以下策略附加到发出请求的 IAM 角色。有关更多信息，请参阅添加 IAM 策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "my-transcribe-http2-policy",
            "Effect": "Allow",
            "Action": "transcribe:StartStreamTranscription",
            "Resource": "*"
        }
    ]
}
```

2. 要启动会话，请向 Amazon Transcribe发送 HTTP/2 请求。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: YYYYMMDDTHHMMSSZ
Authorization: AWS4-HMAC-SHA256 Credential=access-key/YYYYMMDD/us-west-2/
transcribe/aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-
amz-date;x-amz-target;x-amz-security-token, Signature=string
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
transfer-encoding: chunked
```

API 参考中列出了其它操作和参数；常用参数部分中列出了所有 AWS API 操作通用的参数。

Amazon Transcribe 发送以下响应：

```
HTTP/2.0 200
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-request-id: 8a08df7d-5998-48bf-a303-484355b4ab4e
x-amzn-transcribe-session-id: b4526fcf-5eee-4361-8192-d1cb9e9d6887
content-type: application/json
```

3. 创建包含您音频数据的音频事件。将下表中描述的标头与音频字节组块一起组合到事件编码消息中。要创建事件消息负载，请按照原始字节格式使用缓冲区。

标头名称字节长度	标头名称（字符串）	标头值类型	值字符串字节长度	值字符串 (UTF-8)
13	:content-type	7	24	application/ octet-stream
11	:event-type	7	10	AudioEvent
13	:message-type	7	5	事件

此示例请求中的二进制数据是经过 base64 编码的。在实际请求中，数据为原始字节。

```
:content-type: "application/vnd.amazon.eventstream"
:event-type: "AudioEvent"
:message-type: "event"
UklGRjzxPQBXQVZFZm10IBAAAAABAAEAgD4AAAB9AAACABAAZGF0YVTwPQAAAAAAAAAAAAAAAAD//wIA/
f8EAA==
```

4. 创建一个音频消息，其中包含音频数据。

   a. 音频消息数据帧包含事件编码标头，这些标头包含音频组块和音频事件的当前日期和签名。

标头名称字节长度	标头名称（字符串）	标头值类型	值字符串字节长度	值
16	:chunk-signature	6	变化	生成的签名
5	:date	8	8	时间戳

此请求中的二进制数据是经过 base64 编码的。在实际请求中，数据为原始字节。

```
:date: 2019-01-29T01:56:17.291Z
:chunk-signature: signature

AAAA0gAAAIKVoRFcTTcjb250ZW50LXR5cGUHABhhcHBsaWNhdGlvbi9vY3RldC1zdHJlYW0LOmV2ZW50LXR5
cGUHAApBdWRpb0V2ZW50DTptZXNzYWdlLXR5cGUHAAVldmVudAxDb256ZW50LVR5cGUHABphcHBsaWNhdGlv
bi94LWFtei1qc29uLTEuMVJJRkY88T0AV0FWRWZtdCAQAAAAAQABAIA
+AAAAfQAAAgAQAGRhdGFU8D0AAAAA
AAAAAAAAAAA//8CAP3/BAC7QLFf
```

b. 构造待签字符串，如为签名版本 4 创建待签字符串。您的字符串遵循以下格式：

```
String stringToSign =
"AWS4-HMAC-SHA256" +
"\n" +
DateTime +
"\n" +
Keypath +
"\n" +
Hex(priorSignature) +
"\n" +
HexHash(nonSignatureHeaders) +
"\n" +
HexHash(payload);
```

- DateTime：签名的创建日期和时间。格式为 YYYYMMDDTHHMMSSZ，其中 YYYY=year、MM=month、DD=day、HH=hour、MM=minute、SS=seconds，T 和 Z 是固定字符。有关更多信息，请参阅处理签名版本 4 中的日期。

- Keypath：采用 `date/region/service/aws4_request` 格式的签名范围。例如，`20220127/us-west-2/transcribe/aws4_request`。
- Hex：一个将输入编码为十六进制表示形式的函数。
- priorSignature：上一帧的签名。对于第一个数据帧，请使用标头帧的签名。
- HexHash: 一个函数，它首先创建其输入的 SHA-256 哈希值，然后使用十六进制函数对哈希进行编码。
- nonSignatureHeaders: 编码为字符串的 DateTime 标头。
- payload：包含音频事件数据的字节缓冲区。

c. 从您的 AWS 私有访问密钥中派生签名密钥，然后用它来签名`stringToSign`。为了实现更高程度的保护，派生密钥特定于日期、服务和 AWS 区域。有关更多信息，请参阅为 AWS签名版本 4 计算签名。

请务必实施用于派生签名密钥的 `GetSignatureKey` 函数。如果您尚未派生签名密钥，请参阅关于如何为签名版本 4 派生签名密钥的示例。

```
String signature = HMACSHA256(derivedSigningKey, stringToSign);
```

- HMACSHA256：一个使用 SHA-256 哈希函数创建签名的函数。
- derivedSigningKey：签名版本 4 签名密钥。
- stringToSign：您为数据框计算的字符串。

在计算数据帧的签名后，构造一个包含日期、签名和音频事件负载的字节缓冲区。将字节数组发送到 Amazon Transcribe 以进行转录。

5. 要向指示音频流已完成，请发送仅包含日期和签名的结束帧（空数据帧）。构造此结束帧的方式与构造数据帧的方式相同。

Amazon Transcribe 使用发送到您的应用程序的一系列转录事件进行响应。此响应是经过事件流编码的。它包含标准前导信息和以下标头。

标头名称字节长度	标头名称（字符串）	标头值类型	值字符串字节长度	值字符串 (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	15	TranscriptEvent

标头名称字节长度	标头名称（字符串）	标头值类型	值字符串字节长度	值字符串 (UTF-8)
13	:message-type	7	5	事件

这些事件采用原始字节格式发送。在该示例中，字节是经过 base64 编码的。

```
AAAAUwAAAEP1RHpYBTpkYXRlCAAAAWiXUkMLEDpjaHVuay1zaWduYXR1cmUGACCt6Zy+uymwEK2SrLp/
zVBI
5eGn83jdBwCaRUBJA+eaDafqjqI=
```

要查看转录结果，请使用事件流编码对原始字节进行解码。

```
:content-type: "application/vnd.amazon.eventstream"
:event-type: "TranscriptEvent"
:message-type: "event"

{
    "Transcript":
        {
            "Results":
                [
                    results
                ]
        }
}
```

6. 要结束音频流，请向 Amazon Transcribe发送一个空的音频事件。完全像创建任何其他音频事件一样，创建负载为空的音频事件。为事件签名并将此签名包括在 `:chunk-signature` 标头中，如下所示：

```
:date: 2019-01-29T01:56:17.291Z
:chunk-signature: signature
```

处理 HTTP/2 流式转录错误

如果在处理您的媒体流时出现错误，则 Amazon Transcribe 会发送异常响应。响应是经过事件流编码的。

此响应包含标准前导信息和以下标头：

标头名称字节长度	标头名称（字符串）	标头值类型	值字符串字节长度	值字符串 (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	19	BadRequestException
13	:message-type	7	9	exception

异常响应解码后将包含以下信息：

```
:content-type: "application/vnd.amazon.eventstream"
:event-type: "BadRequestException"
:message-type: "exception"

Exception message
```

设置直 WebSocket 播

用于流式传输转录请求的WebSocket协议的关键组成部分 Amazon Transcribe 是：

- 升级请求。它包含您的请求的查询参数，以及 Amazon Transcribe 用作对数据帧进行签名的种子签名的签名。
- 一个或多个消息帧采用事件流编码，其中包含元数据和原始音频字节。
- 结束帧。这是事件流编码中正文为空的已签名消息。

> **Note**
>
> Amazon Transcribe 每个 WebSocket 会话仅支持一个直播。如果您尝试使用多个音频流，则转录请求将失败。

1. 将以下策略附加到发出请求的 IAM 角色。有关更多信息，请参阅添加 IAM 策略。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "my-transcribe-websocket-policy",
            "Effect": "Allow",
            "Action": "transcribe:StartStreamTranscriptionWebSocket",
            "Resource": "*"
        }
    ]
}
```

2. 要启动会话，请使用以下格式创建预签名 URL。为了便于阅读，已增加了换行符。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=access-key%2FYYYYMMDD%2Fus-west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=YYYYMMDDTHHMMSSZ
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
```

> **Note**
>
> X-Amz-Expires 的最大值为 300（5 分钟）。

API 参考中列出了其它操作和参数；常用参数部分中列出了所有 AWS API 操作通用的参数。

构造请求的 URL 并创建签名版本 4 签名，请参阅以下步骤。这些示例是伪代码。

a. 创建规范请求。一个规范请求，其中包含来自标准格式的请求的信息。这样可以确保在 AWS 收到请求时，它可以计算出您为网址创建的相同签名。有关更多信息，请参阅为签名版本 4 创建规范请求。

```
# HTTP verb
method = "GET"
```

```
# Service name
service = "transcribe"
# Region
region = "us-west-2"
# Amazon Transcribe streaming endpoint
endpoint = "wss://transcribestreaming.us-west-2.amazonaws.com:8443"
# Host
host = "transcribestreaming.us-west-2.amazonaws.com:8443"
# Date and time of request
amz-date = YYYYMMDDTHHMMSSZ
# Date without time for credential scope
datestamp = YYYYMMDD
```

b. 创建规范 URI，这是域与查询字符串之间的 URI 部分。

```
canonical_uri = "/stream-transcription-websocket"
```

c. 创建规范标头和签名标头。请注意规范标头尾部的 \n。

- 追加小写标头名称，后跟冒号 (:)。
- 追加该标头的值的逗号分隔列表。请勿对有多个值的标头进行值排序。
- 追加一个新行(\n)。

```
canonical_headers = "host:" + host + "\n"
signed_headers = "host"
```

d. 将算法与哈希算法匹配。使用 SHA-256。

```
algorithm = "AWS4-HMAC-SHA256"
```

e. 创建凭证范围，该范围将派生密钥范围限定为日期、 AWS 区域和服务。例如，*20220127*/*us-west-2*/transcribe/aws4_request。

```
credential_scope = datestamp + "/" + region + "/" + service + "/" +
 "aws4_request"
```

f. 创建规范查询字符串。查询字符串值必须是 URL 编码，并且按名称排序。

- 按字符代码点以升序顺序对参数名称进行排序。具有重复名称的参数应按值进行排序。例如，以大写字母 F 开头的参数名称排在以小写字母 b 开头的参数名称之前。

- 请勿对 RFC 3986 定义的任何非预留字符进行 URI 编码，这些字符包括：A-Z、a-z、0-9、连字符 (-)、下划线 (_)、句点 (.) 和波形符 (~)。
- 使用 %XY 对所有其他字符进行百分比编码，其中“X”和“Y”为十六进制字符（0-9 和大写字母 A-F）。例如，空格字符必须编码为 %20（不像某些编码方案那样包含“+”），扩展 UTF-8 字符必须采用格式 %XY%ZA%BC。
- 对参数值中的任何等于 (=) 字符进行双重编码。

```
canonical_querystring  = "X-Amz-Algorithm=" + algorithm
canonical_querystring += "&X-Amz-Credential="+ URI-encode(access key + "/" +
 credential_scope)
canonical_querystring += "&X-Amz-Date=" + amz_date
canonical_querystring += "&X-Amz-Expires=300"
canonical_querystring += "&X-Amz-Security-Token=" + token
canonical_querystring += "&X-Amz-SignedHeaders=" + signed_headers
canonical_querystring += "&language-code=en-US&media-encoding=flac&sample-
rate=16000"
```

g. 创建负载的哈希。对于 GET 请求，负载为空字符串。

```
payload_hash = HashSHA256(("").Encode("utf-8")).HexDigest()
```

h. 组合以下元素以创建规范请求。

```
canonical_request = method + '\n'
   + canonical_uri + '\n'
   + canonical_querystring + '\n'
   + canonical_headers + '\n'
   + signed_headers + '\n'
   + payload_hash
```

3. 创建待签字符串，其中包含您的请求的元信息。在计算请求签名时，您可以使用该字符串登录下一步。有关更多信息，请参阅为签名版本 4 创建待签字符串。

```
string_to_sign=algorithm + "\n"
   + amz_date + "\n"
   + credential_scope + "\n"
   + HashSHA256(canonical_request.Encode("utf-8")).HexDigest()
```

4. 计算签名。为此，请从您的 AWS 私有访问密钥中获取签名密钥。为了实现更高程度的保护，派生密钥特定于日期、服务和 AWS 区域。使用此派生密钥签名请求。有关更多信息，请参阅计算签名版本 4 的 AWS 签名。

   请务必实施用于派生签名密钥的 GetSignatureKey 函数。如果您尚未派生签名密钥，请参阅关于如何为签名版本 4 派生签名密钥的示例。

   ```
   #Create the signing key
   signing_key = GetSignatureKey(secret_key, datestamp, region, service)

   # Sign the string_to_sign using the signing key
   signature = HMAC.new(signing_key, (string_to_sign).Encode("utf-8"),
    Sha256()).HexDigest
   ```

   函数 HMAC(key, data) 表示以二进制格式返回结果的 HMAC-SHA256 函数。

5. 将签名信息添加到请求中并创建请求 URL。

   在计算签名之后，将它添加到查询字符串。有关更多信息，请参阅将签名添加到请求中。

   首先，将身份验证信息添加到查询字符串。

   ```
   canonical_querystring += "&X-Amz-Signature=" + signature
   ```

   其次，创建请求的 URL。

   ```
   request_url = endpoint + canonical_uri + "?" + canonical_querystring
   ```

   使用您的 WebSocket 图书馆中的请求 URL 向发出请求 Amazon Transcribe。

6. 对的请求 Amazon Transcribe 必须包含以下标头。通常，这些标头由您的 WebSocket 客户端库管理。

   ```
   Host: transcribestreaming.us-west-2.amazonaws.com:8443
   Connection: Upgrade
   Upgrade: websocket
   Origin: URI-of-WebSocket-client
   Sec-WebSocket-Version: 13
   Sec-WebSocket-Key: randomly-generated-string
   ```

7. Amazon Transcribe 收到您的 WebSocket 请求后，它会以 WebSocket 升级响应进行响应。通常，您的 WebSocket 库会管理此响应并设置用于与之通信的套接字 Amazon Transcribe。

   以下是来自的回复 Amazon Transcribe。为了便于阅读，已增加了换行符。

```
HTTP/1.1 101 WebSocket Protocol Handshake

Connection: upgrade
Upgrade: websocket
websocket-origin: wss://transcribestreaming.us-west-2.amazonaws.com:8443
websocket-location: transcribestreaming.us-west-2.amazonaws.com:8443/stream-
transcription-websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-west-2%2Ftranscribe
%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&language-code=en-US
&session-id=String
&media-encoding=flac
&sample-rate=16000
x-amzn-RequestId: RequestId
Strict-Transport-Security: max-age=31536000
sec-websocket-accept: hash-of-the-Sec-WebSocket-Key-header
```

8. 提出 WebSocket 直播请求。

   建立 WebSocket 连接后，客户端可以开始发送一系列音频帧，每个帧都使用事件流编码进行编码。

   每个数据帧包含三个标头以及原始音频字节组块，后跟描述这些标头的表。

标头名称字节长度	标头名称（字符串）	标头值类型	值字符串字节长度	值字符串 (UTF-8)
13	:content-type	7	24	application/ octet-stream
11	:event-type	7	10	AudioEvent

标头名称字节长度	标头名称（字符串）	标头值类型	值字符串字节长度	值字符串 (UTF-8)
13	:message-type	7	5	事件

9. 要结束数据流，请在事件流编码消息中发送空音频组块。

   响应包含负载中的事件流编码的原始字节。它包含标准前导信息和以下标头。

标头名称字节长度	标头名称（字符串）	标头值类型	值字符串字节长度	值字符串 (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	15	TranscriptEvent
13	:message-type	7	5	事件

   解码二进制响应时，最终会得到包含转录结果的 JSON 结构。

处理 WebSocket 直播错误

如果在处理您的请求时出现异常，则使用包含事件流编码 Amazon Transcribe 响应的终端 WebSocket 帧进行响应。该响应包含下表中描述的标头，以及包含描述性错误消息的响应的正文。发送异常响应后， Amazon Transcribe 发送一个闭合帧。

标头名称字节长度	标头名称（字符串）	标头值类型	值字符串字节长度	值字符串 (UTF-8)
13	:content-type	7	16	application/json
15	:exception-type	7	变化	各种编号，请参阅下文
13	:message-type	7	9	exception

`exception-type` 标头包含以下值之一：

- **BadRequestException**：创建流时发生客户端错误，或者流式处理数据中发生错误。确保您的客户端已准备好接受数据，然后重试请求。
- **InternalFailureException**: Amazon Transcribe 在与客户握手时遇到了问题。请再次尝试您的请求。
- **LimitExceededException**：客户端超出了并发流限制。有关更多信息，请参阅Amazon Transcribe 限制。减少要转录的流的数量。
- **UnrecognizedClientException**：使用错误的访问密钥或私有密钥签署了 WebSocket 升级请求。请确保您正确创建了访问密钥，然后重试请求。

Amazon Transcribe 也可以返回任何常见的服务错误。有关错误列表，请参阅常见错误。

事件流编码

Amazon Transcribe 使用一种名为事件流编码的格式来传输转录。

事件流编码在客户端和服务器之间提供双向通信。发送到 Amazon Transcribe 流媒体服务的数据帧以这种格式编码。来自的响应 Amazon Transcribe 也使用这种编码。

每个消息都包含两部分：前导信息和数据。前导信息包括：

1. 消息的总字节长度
2. 所有标头的组合字节长度

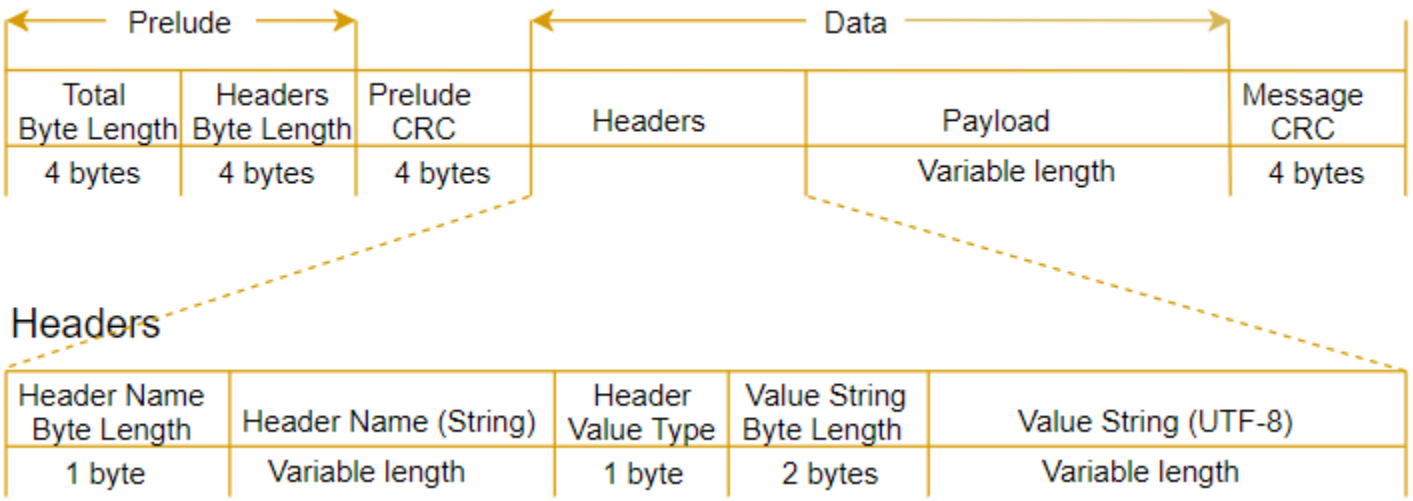
数据部分包含：

1. 标头
2. 有效负载

每个部分以 4 字节 big-endian 整数循环冗余检查 (CRC) 校验和结尾。消息 CRC 校验和同时适用于前导信息部分和数据部分。 Amazon Transcribe 使用 CRC32（通常称为 GZIP CRC32）来计算这两个 CRC。有关 CRC32 的更多信息，请参阅 GZIP 文件格式规范版本 4.3。

总消息开销（包括前导信息和两个校验和）为 16 个字节。

下图显示了构成消息和标头的组件。每个消息有多个标头。



每个消息都包含以下组件：

- 前导消息：包含两个 4 字节的字段，固定总长度为 8 个字节。
  - 第一个 4 字节：这是整个消息的 big-endian 整数字节长度，包括这个 4 字节长度字段本身。
  - 第二个 4 字节：这是消息的“标头”部分的 big-endian 整数字节长度，不包括“标头”长度字段本身。
- 前导信息 CRC：消息的前导信息部分的 4 字节 CRC 校验和，不包括 CRC 本身。前导消息与消息 CRC 有不同的 CRC。这样 Amazon Transcribe 可以确保它可以立即检测到损坏的字节长度信息，而不会导致错误，例如缓冲区溢出。
- 标头：用于批注消息元数据；例如，消息类型和内容类型。消息有多个标头，它们是键值对，其中键是 UTF-8 字符串。标头可按任何顺序出现在消息的“标头”部分中，并且每个标头只能出现一次。
- 负载：要转录的音频内容。
- 消息 CRC：从消息开头到校验和开头的 4 字节 CRC 校验和。也就是说，消息中除 CRC 本身之外的所有内容。

标题帧是直播转录的授权框架。 Amazon Transcribe 使用授权标头的值作为种子，为请求中的数据帧生成授权标头链。

每个标头包含以下组成部分；每帧有多个标头。

- 标头名称字节长度：标头名称的字节长度。
- 标头名称：指示标头类型的标头名称。有关有效值，请参阅下面的帧描述。
- 标头值类型：指示标头值的数字。以下列表显示了标头的可能值及其表示的内容。
  - 0 – TRUE
  - 1 – FALSE

- 2 – BYTE
- 3 – SHORT
- 4 – INTEGER
- 5 – LONG
- 6 – BYTE ARRAY
- 7 – STRING
- 8 – TIMESTAMP
- 9 – UUID

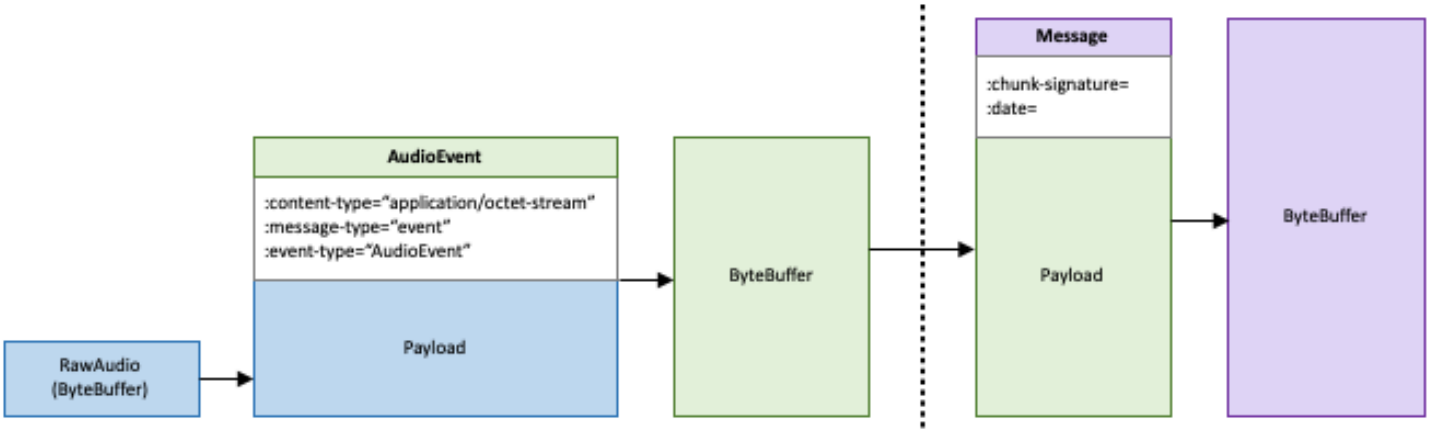
- 值字符串字节长度：标头值字符串的字节长度。
- 标头值：标头字符串的值。此字段的有效值取决于标头的类型。有关更多信息，请参阅设置 HTTP/2 音频流 或设置直 WebSocket 播。

数据帧

每个流式请求都包含一个或多个数据帧。创建数据帧有两个步骤：

1. 将原始音频数据与元数据组合以创建请求负载。
2. 将负载与签名组合以构成将发送到 Amazon Transcribe的事件消息。

下图演示了工作原理。



任务队列列任务队列

使用作业队列，您提交的转录任务请求可以多于可同时处理的转录任务请求。如果没有任务排队，一旦达到允许的并发请求配额，就必须等到一个或多个请求完成后才能提交新请求。

对于转录任务请求，Job 排队是可选的。通话后分析请求会自动启用任务队列。

如果您启用作业队列，则Amazon Transcribe会创建一个包含所有超出限制的请求的队列。请求完成后，系统会立即从您的队列中提取新请求并进行处理。排队的请求按照 FIFO（先进先出）顺序处理。

您最多可以请求我们放放放放放平任务队列列我们放放平任务队列列 如果您超过此限制，则会出现`LimitExceededConcurrentJobException`错误。为了保持最佳性能，最多Amazon Transcribe只使用配额的 90%（带宽比率为 0.9）来处理排队作业。请注意，这些是客户可以请求我们放放放放放放放放放放放放放放放放放放放放放放这些

> **Tip**
>
> 您可以在《AWS一般参考》中找到Amazon Transcribe资源的默认限制和配额列表。客户可以请求我们放放放放其其中一些默认值。

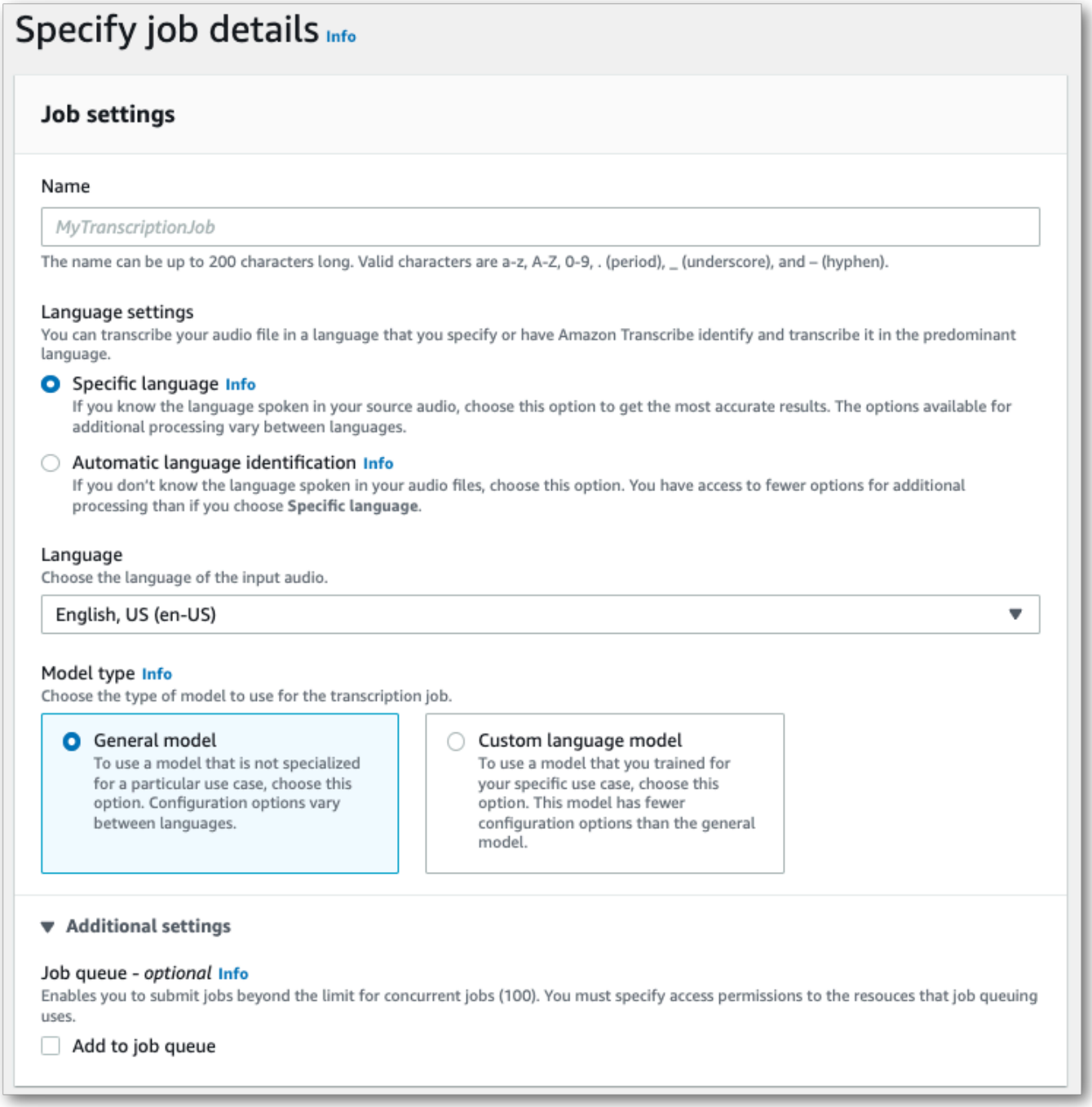
如果您启用任务排队但不超过并发请求的配额，则所有请求都将同时处理。

启用任务队列列列任务队列

您可以使用AWS Management ConsoleAWS CLI、或 AWSSDK 启用任务队列；有关示例，请参阅以下示例：

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。
3. 在 “Job 设置” 框中，有一个 “其他设置” 面板。如果展开此面板，则可以选择 “添加到作业队列” 框以启用作业队列。



4. 在“指定作业详细信息”页面上填写要包含的任何其他字段，然后选择下一步。这将带您进入配置作业-可选页面。
5. 选择“创建作业”以运行转录作业。

AWS CLI

此示例使用带有AllowDeferredExecution子job-execution-settings参数的start-transcription-job命令和参数。请注意，当您在请求AllowDeferredExecution中包含时，还必须包括DataAccessRoleArn。

有关更多信息，请参阅 StartTranscriptionJob 和 JobExecutionSettings。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
--job-execution-settings
 AllowDeferredExecution=true,DataAccessRoleArn=arn:aws:iam::111122223333:role/
ExampleRole
```

这是使用start-transcription-job命令的另一个示例，以及一个支持排队的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://my-first-queueing-request.json
```

my-first-queueing-request.json 文件包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "JobExecutionSettings": {
        "AllowDeferredExecution": true,
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole"
  }
}
```

AWS SDK for Python (Boto3)

此示例使用 start_transcription_job 方法的AllowDeferredExecution参数启用作业队列。AWS SDK for Python (Boto3)请注意，当您在请求AllowDeferredExecution中包含时，还必须包括DataAccessRoleArn。有关更多信息，请参阅 StartTranscriptionJob 和 JobExecutionSettings。

有关使用AWS软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS章。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-queueing-request"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    JobExecutionSettings = {
        'AllowDeferredExecution': True,
        'DataAccessRoleArn': 'arn:aws:iam::111122223333:role/ExampleRole'
   }
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

您可以通过AWS Management Console或提交GetTranscriptionJob请求来查看排队作业的进度。当作业排队时，Status是QUEUED。任务开始处理IN_PROGRESS后，状态将更改为，然后在处理完成FAILED时更改为COMPLETED或完成。

为资源添加标签

标签是一种自定义元数据标签，您可以将其添加到资源中，以便在搜索中更容易识别、组织和查找。标签由两个独立的部分组成：标签键和标签值。这被称为键:值对。

标签键通常代表较大的类别，而标签值代表该类别的子集。例如，你可以使用标签 key=Color 和标签 value=BL UE，这将生成 key: value 对`Color:Blue`。请注意，您可以将标签的值设为空的字符串，但是不能将其设为空值。省略标签值与使用空字符串相同。

> **Tip**
>
> AWS Billing and Cost Management可以使用标签将账单分成动态类别。例如，如果您添加了代表公司内部不同部门的标签，例如`Department:Sales`或`Department:Legal`，则AWS可以为您提供每个部门的成本分配。

在 Amazon Transcribe 中，您可以为以下资源添加标签：

- 转录职位
- 医学转录职位
- 通话分析通话后转录工作
- 自定义词汇
- 自定义医学词汇
- 自定义词汇过滤器
- 自定义语言模型

标签键的长度最大可以为 128 个字符，标签值的长度最大可以为 126 字符；两者都区分大小写。Amazon Transcribe每个资源支持最多 50 个标签。对于给定的资源，每个标签键必须是唯一的，只有一个值。请注意，您的标签不能以开头，`aws:`因为会为系统生成的标签AWS保留此前缀。您无法添加、修改或删除`aws:*`标签，它们不计入您的 tags-per-resource 限制。

> **特定于资源标记的 API 操作**
>
> `ListTagsForResource`, `TagResource`, `UntagResource`
> 要使用标记 API，您必须在请求中包含Amazon 资源名称 (ARN)。ARN 有这样的格式`arn:partition:service:region:account-id:resource-type/resource-`

id。例如，与转录任务关联的 ARN 可能如下所示：arn:*aws*:transcribe:*us-west-2*:*111122223333*:transcription-job/*my-transcription-job-name*。

要了解有关标记的更多信息，包括最佳实践，请参阅标记AWS资源。

基于标签的访问控制

您可以使用标签来控制访问AWS 账户。对于基于标签的访问控制，您需要在IAM策略的条件元素中提供标签信息。然后，您可以使用标签及其关联的标签条件键控制对：

- 资源：根据分配给Amazon Transcribe资源的标签控制对资源的访问。
  - 使用aws:ResourceTag/*key-name*条件键指定必须将哪个标签键:值对附加到资源。
- 请求：控制可以在请求中传递哪些标签。
  - 使用aws:RequestTag/*key-name*条件键指定可在IAM用户或角色中添加、修改或删除的标签。
- 授权流程：控制授权过程任何部分的基于标签的访问权限。
  - 使用aws:TagKeys/条件键控制是否可以对资源、请求或委托人使用特定的标签键。在这种情况下，键值无关紧要。

有关基于标签的访问策略示例，请参阅基于标签查看转录任务。

有关基于标签的访问控制的更多详细信息，请参阅使用标签控制对AWS资源的访问。

向Amazon Transcribe资源添加标签

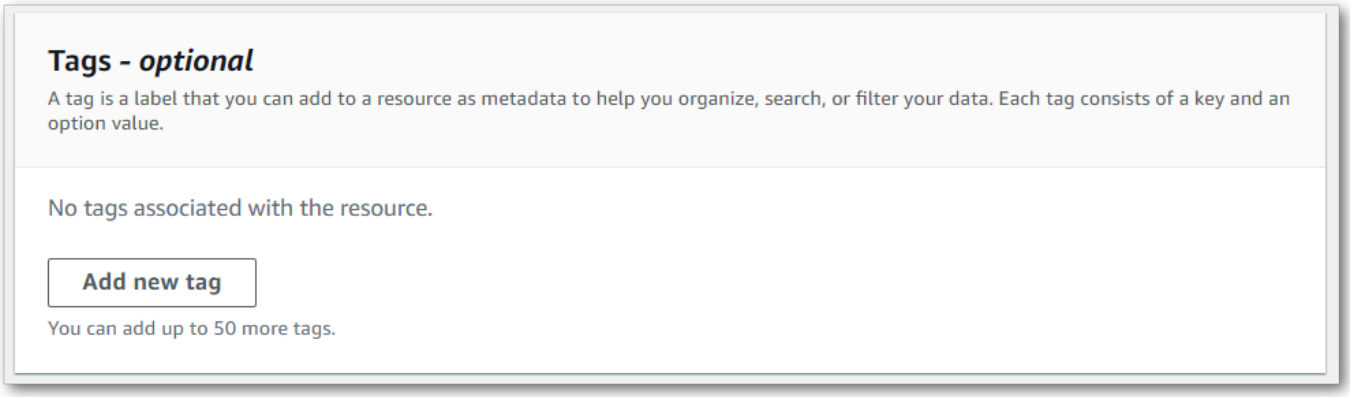
您可以在运行Amazon Transcribe作业之前或之后添加标签。使用现有的 Create* 和 Start* API，您可以在转录请求中添加标签。

您可以使用、或 AWSSDK 添加AWS Management ConsoleAWS CLI、修改或删除标签；有关示例，请参阅以下内容：

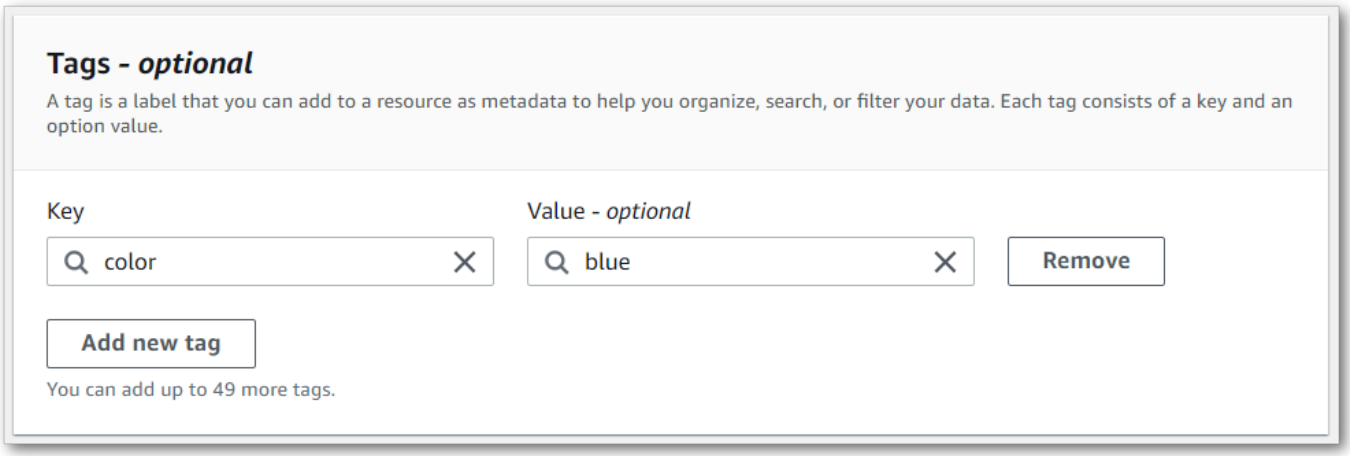
AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。

3. 滚动到“指定作业详细信息”页面底部，找到“标签-可选”框并选择“添加新标签”。



4. 为“密钥”字段和（可选）“值”字段输入信息。



5. 在“指定作业详细信息”页面上填写要包含的任何其他字段，然后选择下一步。这将带您进入配置作业-可选页面。

   选择“创建作业”以运行转录作业。

6. 通过导航到转录作业页面，选择转录作业，然后滚动到该作业信息页面的底部，您可以查看与转录作业相关的标签。如果要编辑标签，可以选择管理标签来执行此操作。

AWS CLI

此示例使用start-transcription-job命令和Tags参数。有关更多信息，请参阅 StartTranscriptionJob 和 Tag。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
--tags Key=color,Value=blue Key=shape,Value=square
```

这是使用start-transcription-job命令的另一个示例，以及为该任务添加标签的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://filepath/my-first-tagging-job.json
```

my-first-tagging-job.json 文件包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "Tags": [
        {
           "Key": "color",
           "Value": "blue"
        },
        {
           "Key": "shape",
           "Value": "square"
        }
  ]
}
```

AWS SDK for Python (Boto3)

以下示例使用使用 start_transcription_job 方法的Tags参数来添加标签。AWS SDK for Python (Boto3) 有关更多信息，请参阅 StartTranscriptionJob 和 Tag。

有关使用AWS软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS章。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Tags = [
        {
            'Key':'color',
            'Value':'blue'
        }
    ]
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

对扬声器进行分区（分区）

使用扬声器计时功能，您可以在转录输出中区分不同的说话人。 Amazon Transcribe可以区分最多 10 个唯一的说话人，并使用唯一值（spk_0通过spk_9）标记每个唯一说话人的文本。

除了标准笔录部分（transcripts和items）外，启用扬声器分区的请求还包括一个speaker_labels部分。本部分按发言者分组，包含每句话的信息，包括演讲者标签和时间戳。

```
"speaker_labels": {
    "channel_label": "ch_0",
    "speakers": 2,
    "segments": [
         {
            "start_time": "4.87",
            "speaker_label": "spk_0",
            "end_time": "6.88",
            "items": [
                {
                    "start_time": "4.87",
                    "speaker_label": "spk_0",
                    "end_time": "5.02"
                },
        ...
        {
            "start_time": "8.49",
            "speaker_label": "spk_1",
            "end_time": "9.24",
            "items": [
                {
                    "start_time": "8.49",
                    "speaker_label": "spk_1",
                    "end_time": "8.88"
                },
```

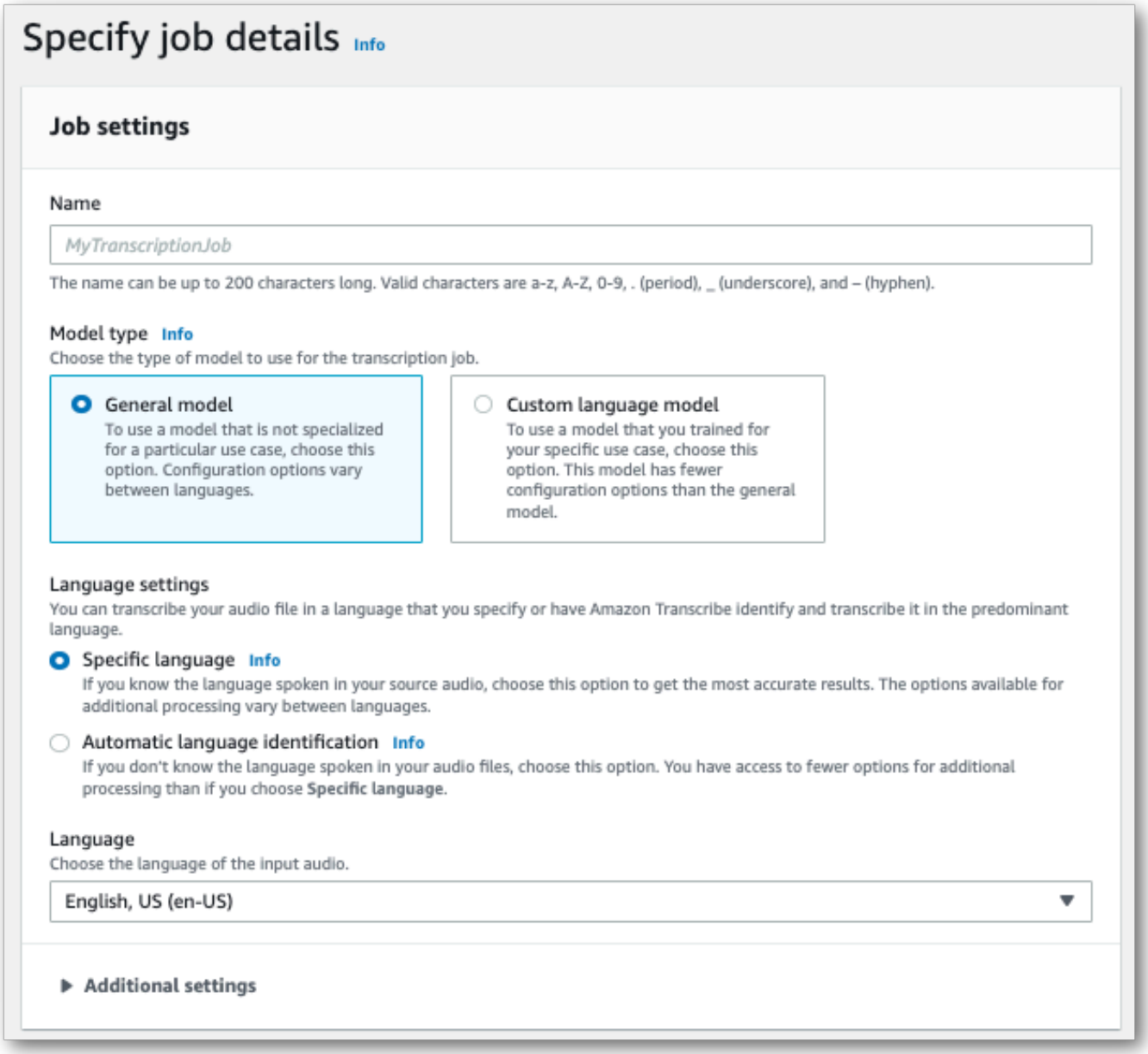
要查看包含扬声器分区的完整示例记录（适用于两个发言者），请参阅对比输出示例（批处理）。

在批量转录中对扬声器进行分区

要对批量转录中的发言者进行分区，请参阅以下示例：

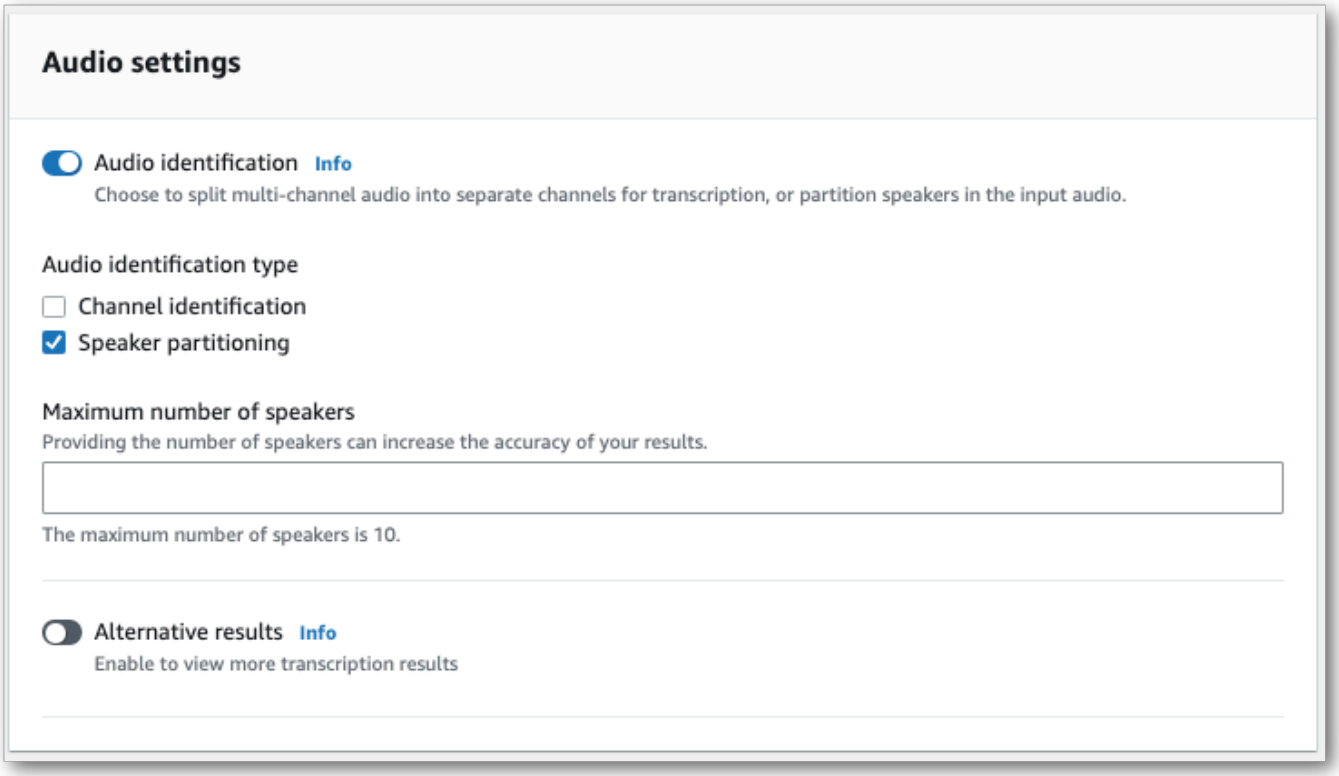
AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。



3. 在 “指定作业详细信息” 页面上填写要包含的所有字段，然后选择下一步。这将带您进入配置作业-可选页面。

   在音频设置面板中，选择扬声器分区（在 “音频识别类型” 标题下）。您可以指定要分区的扬声器数量，单位为秒。



4. 选择 “创建作业” 以运行转录作业。

AWS CLI

此示例使用 start-transcription-job。有关更多信息，请参阅StartTranscriptionJob：

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
--show-speaker-labels TRUE \
--max-speaker-labels 3
```

这是使用start-transcription-job命令的另一个示例，以及一个允许扬声器与该任务进行分区的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
```

```
--cli-input-json file://my-first-transcription-job.json
```

my-first-transcription-job.json 文件包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "ShowSpeakerLabels": 'TRUE',
  "MaxSpeakerLabels": 3
 }
```

AWS SDK for Python (Boto3)

此示例使用 start_transcription_job 方法识别频道。AWS SDK for Python (Boto3)有关更多信息，请参阅StartTranscriptionJob。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    ShowSpeakerLabels: 'TRUE',
    MaxSpeakerLabels: 3
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
```

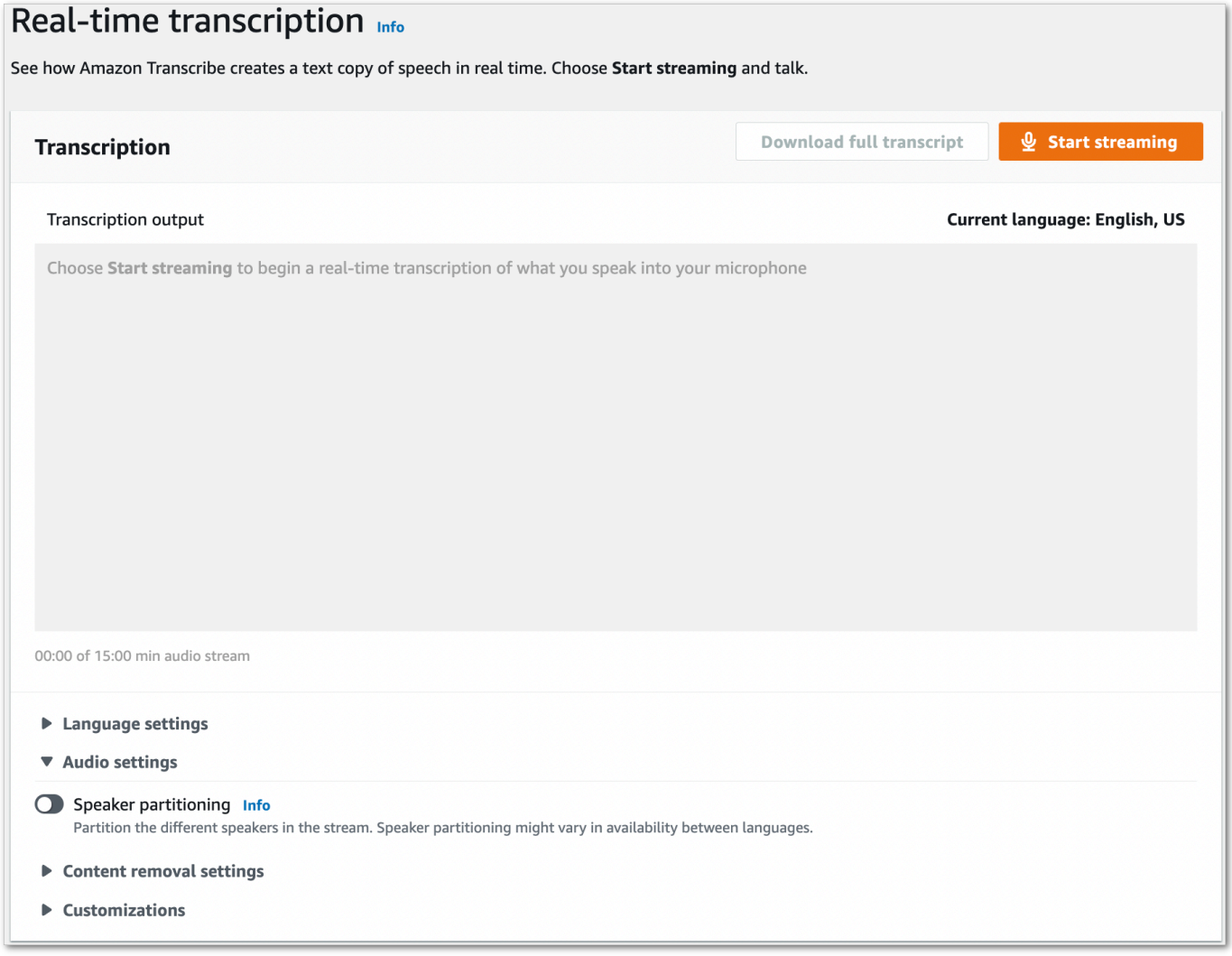
```
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

在流媒体转录中对扬声器进行分区

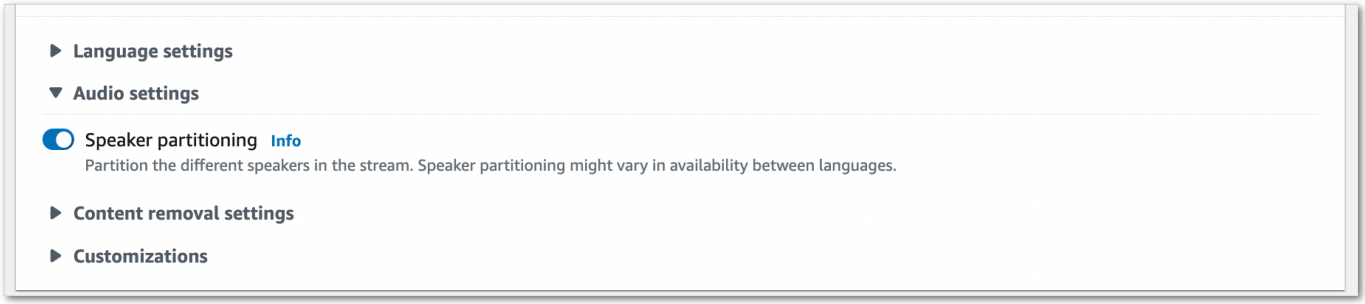
要对流转录中的发言者进行分区，请参阅以下示例：

直播转录

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择 Real-time transcription (实时转录)。向下滚动到音频设置，如果该字段最小化，则将其展开。



3. 开启扬声器分区。



4. 您现在已准备好（单位为秒）。选择 “开始直播” 并开始讲话。要结束听写，请选择 “停止直播”。

HTTP/2 流直播

此示例创建了一个 HTTP/2 请求，用于对转录输出中的扬声器进行分区。有关使用 HTTP/2 流式传输的更多信息Amazon Transcribe，请参阅设置 HTTP/2 音频流。有关特定参数和标题的更多详细信息Amazon Transcribe，请参阅StartStreamTranscription。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: 20220208T235959Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key/20220208/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-transcribe-show-speaker-label: true
transfer-encoding: chunked
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

WebSocket 流

此示例创建了一个预签名 URL，用于在转录输出中将发言者分开。为了便于阅读，已增加了换行符。有关将 WebSocket 直播与一起使用的更多信息Amazon Transcribe，请参阅设置直 WebSocket 播。有关参数的更多详细信息，请参阅StartStreamTranscription。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
```

```
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&language-code=en-US
&specialty=PRIMARYCARE
&type=DICTATION
&media-encoding=flac
&sample-rate=16000
&show-speaker-label=true
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

对比输出示例（批处理）

以下是启用了迪亚化功能的批量转录的输出示例。

```
{
    "jobName": "my-first-transcription-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "I've been on hold for an hour. Sorry about that."
            }
        ],
        "speaker_labels": {
            "channel_label": "ch_0",
            "speakers": 2,
            "segments": [
                {
                    "start_time": "4.87",
                    "speaker_label": "spk_0",
                    "end_time": "6.88",
                    "items": [
                        {
                            "start_time": "4.87",
                            "speaker_label": "spk_0",
                            "end_time": "5.02"
```

```
                    },
                    {
                        "start_time": "5.02",
                        "speaker_label": "spk_0",
                        "end_time": "5.17"
                    },
                    {
                        "start_time": "5.17",
                        "speaker_label": "spk_0",
                        "end_time": "5.29"
                    },
                    {
                        "start_time": "5.29",
                        "speaker_label": "spk_0",
                        "end_time": "5.64"
                    },
                    {
                        "start_time": "5.64",
                        "speaker_label": "spk_0",
                        "end_time": "5.84"
                    },
                    {
                        "start_time": "6.11",
                        "speaker_label": "spk_0",
                        "end_time": "6.26"
                    },
                    {
                        "start_time": "6.26",
                        "speaker_label": "spk_0",
                        "end_time": "6.88"
                    }
                ]
            },
            {
                "start_time": "8.49",
                "speaker_label": "spk_1",
                "end_time": "9.24",
                "items": [
                    {
                        "start_time": "8.49",
                        "speaker_label": "spk_1",
                        "end_time": "8.88"
                    },
                    {
```

```
                        "start_time": "8.88",
                        "speaker_label": "spk_1",
                        "end_time": "9.05"
                    },
                    {
                        "start_time": "9.05",
                        "speaker_label": "spk_1",
                        "end_time": "9.24"
                    }
                ]
            }
        ]
    },
    "items": [
        {
            "start_time": "4.87",
            "speaker_label": "spk_0",
            "end_time": "5.02",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "I've"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "5.02",
            "speaker_label": "spk_0",
            "end_time": "5.17",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "been"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "5.17",
            "speaker_label": "spk_0",
            "end_time": "5.29",
            "alternatives": [
                {
```

```
                    "confidence": "1.0",
                    "content": "on"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "5.29",
            "speaker_label": "spk_0",
            "end_time": "5.64",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "hold"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "5.64",
            "speaker_label": "spk_0",
            "end_time": "5.84",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "for"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "6.11",
            "speaker_label": "spk_0",
            "end_time": "6.26",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "an"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "6.26",
```

```
                "speaker_label": "spk_0",
                "end_time": "6.88",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hour"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "speaker_label": "spk_0",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            },
            {
                "start_time": "8.49",
                "speaker_label": "spk_1",
                "end_time": "8.88",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Sorry"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "8.88",
                "speaker_label": "spk_1",
                "end_time": "9.05",
                "alternatives": [
                    {
                        "confidence": "0.902",
                        "content": "about"
                    }
                ],
                "type": "pronunciation"
            },
```

```
            {
                "start_time": "9.05",
                "speaker_label": "spk_1",
                "end_time": "9.24",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "that"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "speaker_label": "spk_1",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ]
    },
    "status": "COMPLETED"
}
```

转录多声道音频

如果您的音频有两个通道，则可以使用频道识别来分别转录每个频道的语音。 Amazon Transcribe目前不支持超过两个声道的音频。

在您的脚本中，会为频道分配标签ch_0和ch_1。

除了标准脚本部分（transcripts和items）外，启用频道识别的请求还包括一个channel_labels部分。本部分包含按频道分组的每句话语或标点符号及其相关的频道标签、时间戳和置信度分数。

```
"channel_labels": {
    "channels": [
        {
            "channel_label": "ch_0",
            "items": [
                {
                    "channel_label": "ch_0",
                    "start_time": "4.86",
                    "end_time": "5.01",
                    "alternatives": [
                        {
                            "confidence": "1.0",
                            "content": "I've"
                        }
                    ],
                    "type": "pronunciation"
                },
                ...
            "channel_label": "ch_1",
            "items": [
                {
                    "channel_label": "ch_1",
                    "start_time": "8.5",
                    "end_time": "8.89",
                    "alternatives": [
                        {
                            "confidence": "1.0",
                            "content": "Sorry"
                        }
                    ],
                    "type": "pronunciation"
```

```
            },
            ...
        "number_of_channels": 2
    },
```

请注意，如果一个频道上的某个人与另一个频道上的某个人同时说话，则当这些人互相发言时，每个频道的时间戳会重叠。

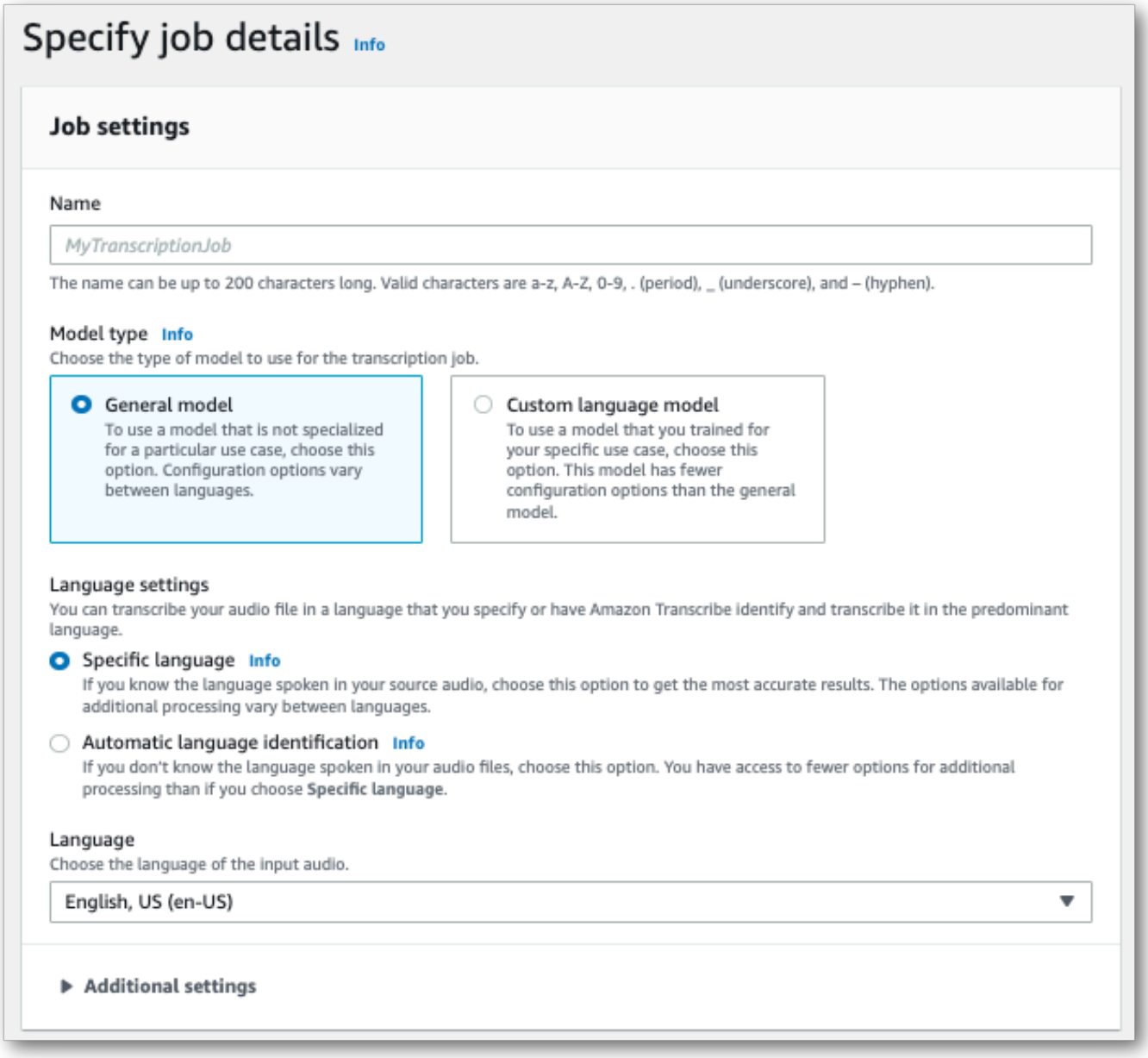
要查看带有频道标识的完整示例脚本，请参阅信道识别输出示例（批处理）。

在批量转录中使用信道识别

要识别批量转录中的频道，可以使用AWS Management ConsoleAWS CLI、或 AWSSDK；有关示例，请参阅以下内容：

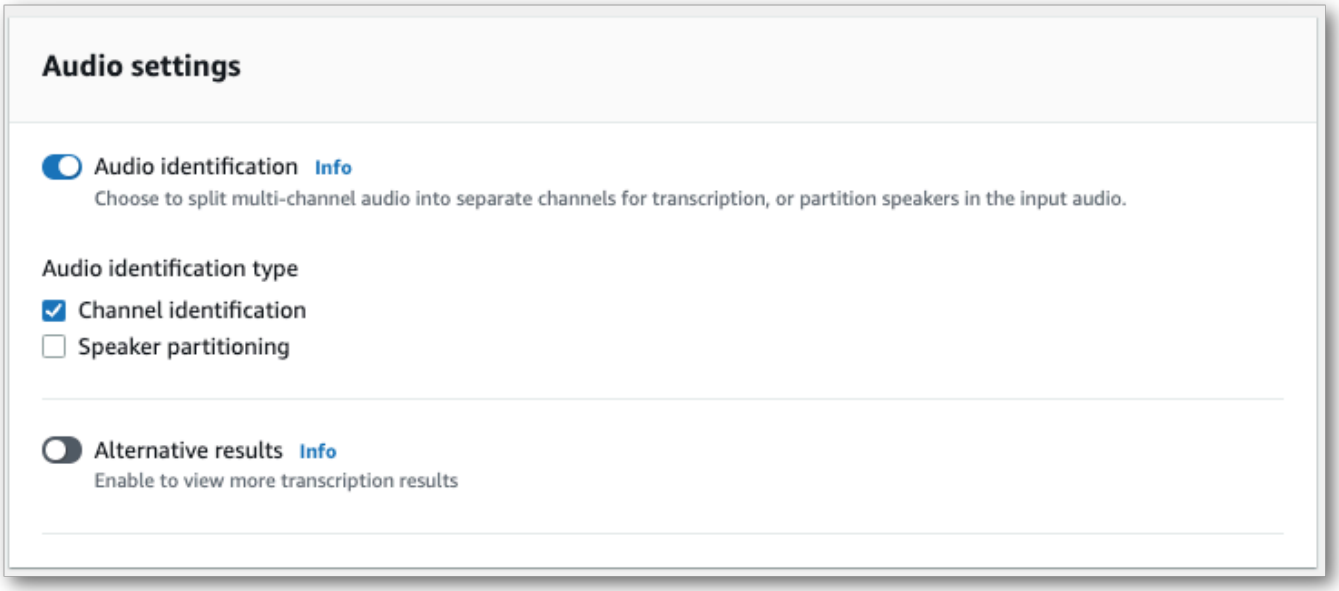
AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。



3. 在 “指定作业详细信息” 页面上填写要包含的所有字段，然后选择下一步。这将带您进入配置作业-可选页面。

   在音频设置面板中，选择频道识别（在 “音频识别类型” 标题下）。



4. 选择 “创建作业” 以运行转录作业。

AWS CLI

此示例使用 start-transcription-job。有关更多信息，请参阅StartTranscriptionJob：

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
--settings ChannelIdentification=true
```

这是使用start-transcription-job命令的另一个示例，以及一个支持通过该任务进行信道识别的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://my-first-transcription-job.json
```

my-first-transcription-job.json 文件包含以下请求正文。

```
{
```

```
    "TranscriptionJobName": "my-first-transcription-job",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
    "OutputBucketName": "DOC-EXAMPLE-BUCKET",
    "OutputKey": "my-output-files/",
    "LanguageCode": "en-US",
    "Settings": {
        "ChannelIdentification": true
    }
}
```

AWS SDK for Python (Boto3)

此示例使用 start_transcription_job 方法识别频道。AWS SDK for Python (Boto3)有关更多信息，请参阅StartTranscriptionJob。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Settings = {
        'ChannelIdentification':True
    }
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

在流媒体转录中使用频道识别

要识别流媒体转录中的频道，可以使用 HTTP/2 或 WebSockets；有关示例，请参见以下内容：

HTTTP/2 直播

此示例创建了一个 HTTP/2 请求，用于在转录输出中分隔频道。有关使用 HTTP/2 流式传输的更多信息Amazon Transcribe，请参阅设置 HTTP/2 音频流。有关特定参数和标题的更多详细信息Amazon Transcribe，请参阅StartStreamTranscription。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: 20220208T235959Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key/20220208/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-channel-identification: TRUE
transfer-encoding: chunked
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

WebSocket 流

此示例创建了一个预签名 URL，用于分隔转录输出中的频道。为了便于阅读，已增加了换行符。有关将 WebSocket 直播与一起使用的更多信息Amazon Transcribe，请参阅设置直 WebSocket 播。有关参数的更多详细信息，请参阅StartStreamTranscription。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
```

```
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&language-code=en-US
&specialty=PRIMARYCARE
&type=DICTATION
&media-encoding=flac
&sample-rate=16000
&channel-identification=TRUE
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

信道识别输出示例（批处理）

以下是启用信道识别的批量转录的输出示例。

```
{
    "jobName": "my-first-transcription-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "I've been on hold for an hour. Sorry about that."
            }
        ],
        "channel_labels": {
            "channels": [
                {
                    "channel_label": "ch_0",
                    "items": [
                        {
                            "channel_label": "ch_0",
                            "start_time": "4.86",
                            "end_time": "5.01",
                            "alternatives": [
                                {
                                    "confidence": "1.0",
                                    "content": "I've"
                                }
                            ],
                            "type": "pronunciation"
                        },
                        {
                            "channel_label": "ch_0",
                            "start_time": "5.01",
```

```
                "end_time": "5.16",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "been"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.16",
                "end_time": "5.28",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "on"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.28",
                "end_time": "5.62",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hold"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.62",
                "end_time": "5.83",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "for"
                    }
                ],
                "type": "pronunciation"
```

```
                    },
                    {
                        "channel_label": "ch_0",
                        "start_time": "6.1",
                        "end_time": "6.25",
                        "alternatives": [
                            {
                                "confidence": "1.0",
                                "content": "an"
                            }
                        ],
                        "type": "pronunciation"
                    },
                    {
                        "channel_label": "ch_0",
                        "start_time": "6.25",
                        "end_time": "6.87",
                        "alternatives": [
                            {
                                "confidence": "1.0",
                                "content": "hour"
                            }
                        ],
                        "type": "pronunciation"
                    },
                    {
                        "channel_label": "ch_0",
                        "language_code": "en-US",
                        "alternatives": [
                            {
                                "confidence": "0.0",
                                "content": "."
                            }
                        ],
                        "type": "punctuation"
                    }
                ]
            },
            {
            "channel_label": "ch_1",
                "items": [
                    {
                        "channel_label": "ch_1",
                        "start_time": "8.5",
```

```
                "end_time": "8.89",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Sorry"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_1",
                "start_time": "8.89",
                "end_time": "9.06",
                "alternatives": [
                    {
                        "confidence": "0.9176",
                        "content": "about"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_1",
                "start_time": "9.06",
                "end_time": "9.25",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "that"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_1",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ]
```

```
                }
            ],
            "number_of_channels": 2
        },
        "items": [
            {
                "channel_label": "ch_0",
                "start_time": "4.86",
                "end_time": "5.01",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "I've"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.01",
                "end_time": "5.16",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "been"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.16",
                "end_time": "5.28",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "on"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.28",
```

```
                "end_time": "5.62",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hold"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.62",
                "end_time": "5.83",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "for"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "6.1",
                "end_time": "6.25",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "an"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "6.25",
                "end_time": "6.87",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hour"
                    }
                ],
                "type": "pronunciation"
```

```
            },
            {
                "channel_label": "ch_0",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            },
            {
                "channel_label": "ch_1",
                "start_time": "8.5",
                "end_time": "8.89",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Sorry"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_1",
                "start_time": "8.89",
                "end_time": "9.06",
                "alternatives": [
                    {
                        "confidence": "0.9176",
                        "content": "about"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_1",
                "start_time": "9.06",
                "end_time": "9.25",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "that"
                    }
```

```
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_1",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ]
    },
    "status": "COMPLETED"
}
```

识别媒体中的主要语言

Amazon Transcribe 无需指定语言代码即可自动识别媒体中使用的语言。

批量转录语言识别可以识别媒体文件中使用的主要语言，或者，如果您的媒体包含多种语言，则可以识别所有使用的语言。为了提高语言识别的准确性，您可以选择提供一份您认为媒体中可能存在的两种或多种语言的列表。

流语言识别可以在每个声道中识别一种语言（最多支持两个声道），或者，如果流包含多种语言，它可以识别所说的所有语言。流式转录请求中必须至少包含两个额外的语言选项。提供语言选项可以更快地识别语言。Amazon Transcribe 识别语言的速度越快，流式转录前几秒数据丢失的变化就越小。

> **Important**
>
> 批量转录和流式转录支持的语言有所不同。有关详细信息，请参阅支持的语言表中的数据输入列。请注意，语言识别目前不支持瑞典语和越南语。

要了解有关使用语言识别的监控和事件的信息，请参阅语言识别事件。

批量转录作业的语言识别

使用批量语言识别来自动识别媒体文件中的一种或多种语言。

如果您的媒体仅包含一种语言，则可以启用单语言识别，该功能可以识别媒体文件中使用的主导语言，并仅使用该语言创建转录。

如果您的媒体包含多种语言，则可以启用多语言识别，该功能可以识别媒体文件中使用的所有语言，并使用识别出的每种语言创建转录。请注意，这将会生成多语言转录。您可以使用其他服务，例如 Amazon Translate，来翻译您的成绩单。

有关支持的语言和相关语言代码的完整列表，请参阅支持的语言表。

为获得最佳效果，请确保您的媒体文件包含至少 30 秒的语音。

有关 AWS Management Console、 AWS CLI和 AWS Python 开发工具包的用法示例，请参阅在批量转录中使用语言识别。

识别多语言音频中的语言

多语言识别专为多语言媒体文件而设计，可为您提供反映媒体中使用的所有支持的语言的转录。这就表示，如果发言者在对话中变换语言，或者如果每个参与者说的是不同的语言，则您的转录输出会正确检测并转录每种语言。例如，如果您的媒体包含双语发言者，其交替使用美国英语 (en-US) 和印地语 (hi-IN)，则多语言识别可以识别说出的美国英语并转录为 en-US，然后将说出的印地语转录为 hi-IN。

这不同于单语言识别，后者只使用一种主导语言来创建转录。在这种情况下，主要语言以外的任何语言都会被错误地转录。

> **Note**
>
> 多语言识别目前不支持编辑和自定义语言模型。

> **Note**
>
> 目前支持以下语言进行多语言识别：en-ab、en-au、en-GB、en-ie、en-in、en-nz、en-US、en-za、es-es、es-us、fr-ca、fr-fr、zh-tw、pt-br、pt-pt、de-pt、de-ch、de-de、a-za、ar-ae、da-dk、he-il、hi-in、id-id，fa-ir、it-it、ja-jp、ko-kr、ms-my、nl-nl、ru-ru、ta-in、te-in、te-in、th-th、tr-tr

多语言转录提供检测到的语言的摘要以及每种语言在媒体中使用的总时间。示例如下：

```
"results": {
        "transcripts": [
            {
                "transcript": "welcome to Amazon transcribe. ## ## ###### ### #### ####
 ## #### ### ####################"
            }
        ],

    ...

        "language_codes": [
            {
                "language_code": "en-US",
```

```
                "duration_in_seconds": 2.45
            },
            {
                "language_code": "hi-IN",
                "duration_in_seconds": 5.325
            },
            {
                "language_code": "ja-JP",
                "duration_in_seconds": 4.15
            }
        ]
}
```

提高语言识别的准确性

通过语言识别，您可以选择包含您认为媒体中可能存在的语言列表。包括语言选项 (LanguageOptions) 限制仅使用您在 Amazon Transcribe 将音频与正确语言匹配时指定的语言，这可以加快语言识别速度并提高与分配正确语言方言相关的准确性。

如果选择包含语言代码，则必须包含至少两个语言代码。您可以包含的语言代码数量没有限制，但为了获得最佳效率和准确性，我们建议使用两到五个语言代码。

> **Note**
>
> 如果您在请求中包含语言代码，而您提供的语言代码均不与您的音频中标识的一个或多个语言相匹配，则请从您指定的语言代码 Amazon Transcribe 中选择最接近的语言匹配项。然后，它会生成该语言的转录。例如，如果您的媒体 Amazon Transcribe 使用的是美国英语 (en-US)，并且您提供的语言代码zh-CNfr-FRde-DE、和， Amazon Transcribe 很可能会将您的媒体与德语 (de-DE) 匹配并生成德语转录。语言代码和说出的语言不匹配可能会导致转录不准确，因此我们建议在添加语言代码时要小心谨慎。

将语言识别与其它 Amazon Transcribe 特征结合使用

您可以将批量语言识别与任何其它 Amazon Transcribe 特征结合使用。如果将语言识别与其它特征结合使用，则只能使用这些特征支持的语言。例如，如果在内容编辑中使用语言识别，则只能使用美国英语 (en-US)，因为这是编辑支持的唯一语言。有关更多信息，请参阅支持的语言和特定语言的特征。

> **Important**
>
> 如果您在启用了内容编辑功能的情况下使用自动语言识别，并且您的音频包含美国英语 (en-US) 以外的语言，则您的转录中只有美国英语内容会被编辑。其它语言则无法编辑，也不会出现警告或作业失败。

自定义语言模型、自定义词汇表和自定义词汇表过滤器

如果要在语言识别请求中添加一个或多个自定义语言模型、自定义词汇表或自定义词汇表过滤器，则必须包含 `LanguageIdSettings` 参数。然后，您可以使用相应的自定义语言模型、自定义词汇表和自定义词汇表过滤器来指定语言代码。请注意，多语言识别不支持自定义语言模型。

为了确保识别出正确的语言方言，建议您在使用 `LanguageIdSettings` 时包含 `LanguageOptions`。例如，如果您指定了`en-US`自定义词汇表，但 Amazon Transcribe 确定媒体中使用的语言是`en-AU`，则您的自定义词汇不会应用于您的转录。如果您包含 `LanguageOptions` 并指定 `en-US` 为唯一的英语方言，则您的自定义词汇表会应用于您的转录。

有关请求中的 `LanguageIdSettings` 的示例，请参阅在批量转录中使用语言识别部分 AWS CLI 和 AWS SDK 下拉面板中的选项 2。

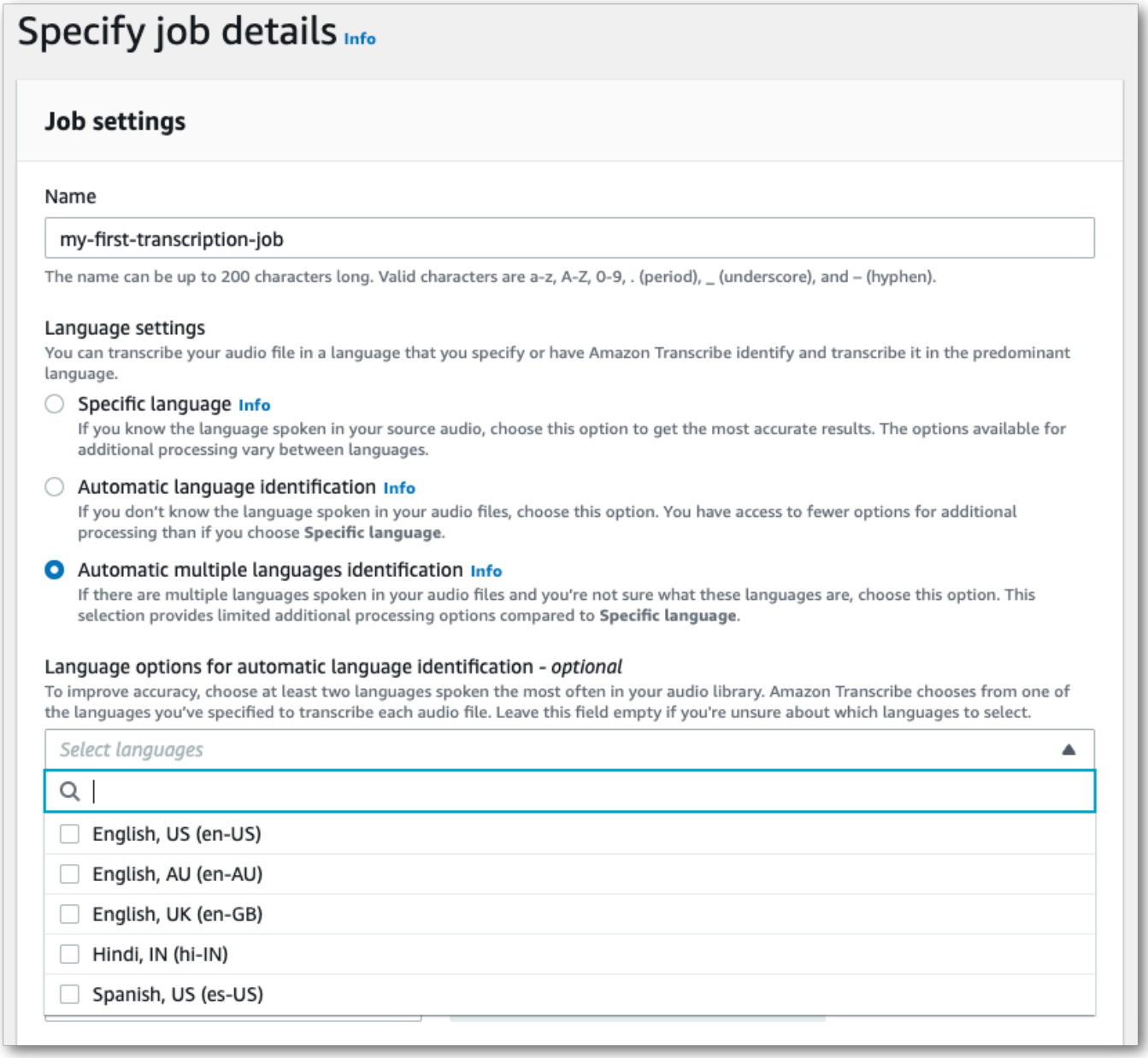
在批量转录中使用语言识别

您可以通过 AWS Management Console、AWS CLI 或 AWS SDK 在批量转录作业中使用自动语言识别；有关示例，请参阅以下内容：

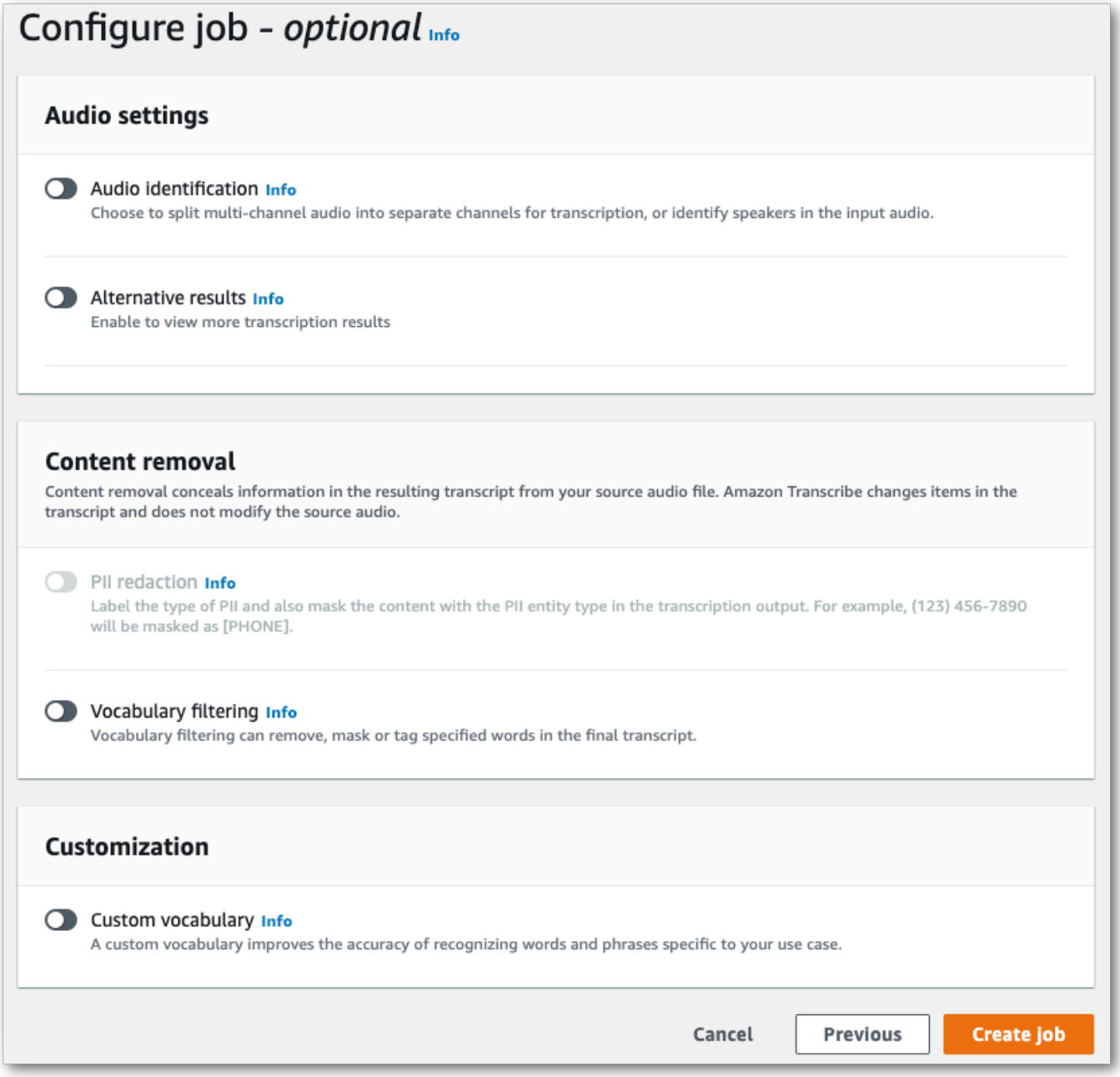
AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开指定作业详细信息页面。
3. 在作业设置面板中，找到语言设置部分，然后选择自动语言识别或自动多语言识别。

   如果您知道音频文件中存在哪些语言，则可以选择多种语言选项（从选择语言下拉框中）。提供语言选项可以提高准确性，但不会要求这么做。



4. 在指定作业详细信息页面上填写要包含的任何其它字段，然后选择下一步。此时您将会看到配置作业 - 可选页面。



5. 选择创建作业以运行您的转录作业。

AWS CLI

此示例使用start-transcription-job命令和IdentifyLanguage参数。有关更多信息，请参阅 StartTranscriptionJob 和 LanguageIdSettings。

选项 1：不使用 language-id-settings 参数。如果您未在请求中包含自定义语言模型、自定义词汇表或自定义词汇表过滤器，请使用此选项。虽然 language-options 为可选项，我们还是建议使用该选项。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--identify-language \  (or --identify-multiple-languages) \
--language-options "en-US" "hi-IN"
```

选项 2：使用 language-id-settings 参数。如果您在请求中包含了自定义语言模型、自定义词汇表或自定义词汇表过滤器，请使用此选项。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--identify-language \  (or --identify-multiple-languages)
--language-options "en-US" "hi-IN" \
--language-id-settings en-US=VocabularyName=my-en-US-vocabulary,en-
US=VocabularyFilterName=my-en-US-vocabulary-filter,en-US=LanguageModelName=my-en-US-
language-model,hi-IN=VocabularyName=my-hi-IN-vocabulary,hi-IN=VocabularyFilterName=my-
hi-IN-vocabulary-filter
```

以下是另一个使用start-transcription-job命令的示例，以及标识语言的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://filepath/my-first-language-id-job.json
```

my-first-language-id-job.json 文件包含以下请求正文。

选项 1：不使用 LanguageIdSettings 参数。如果您未在请求中包含自定义语言模型、自定义词汇表或自定义词汇表过滤器，请使用此选项。虽然 LanguageOptions 为可选项，我们还是建议使用该选项。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
   },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "IdentifyLanguage": true,  (or "IdentifyMultipleLanguages": true),
  "LanguageOptions": [
        "en-US", "hi-IN"
  ]
}
```

选项 2：使用 LanguageIdSettings 参数。如果您在请求中包含了自定义语言模型、自定义词汇表或自定义词汇表过滤器，请使用此选项。

```
{
   "TranscriptionJobName": "my-first-transcription-job",
   "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
   },
   "OutputBucketName": "DOC-EXAMPLE-BUCKET",
   "OutputKey": "my-output-files/",
   "IdentifyLanguage": true,  (or "IdentifyMultipleLanguages": true)
   "LanguageOptions": [
        "en-US", "hi-IN"
   ],
   "LanguageIdSettings": {
         "en-US" : {
            "LanguageModelName": "my-en-US-language-model",
            "VocabularyFilterName": "my-en-US-vocabulary-filter",
            "VocabularyName": "my-en-US-vocabulary"
         },
         "hi-IN": {
             "VocabularyName": "my-hi-IN-vocabulary",
             "VocabularyFilterName": "my-hi-IN-vocabulary-filter"
         }
    }
}
```

AWS SDK for Python (Boto3)

此示例使用 start_transcription_ job 方法的IdentifyLanguage参数来标识文件的语言。 AWS SDK for Python (Boto3) 有关更多信息，请参阅 StartTranscriptionJob 和 LanguageIdSettings。

有关使用 AWS 软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本章。使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS

选项 1：不使用 LanguageIdSettings 参数。如果您未在请求中包含自定义语言模型、自定义词汇表或自定义词汇表过滤器，请使用此选项。虽然 LanguageOptions 为可选项，我们还是建议使用该选项。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    MediaFormat = 'flac',
    IdentifyLanguage = True,  (or IdentifyMultipleLanguages = True),
    LanguageOptions = [
        'en-US', 'hi-IN'
    ]
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

选项 2：使用 LanguageIdSettings 参数。如果您在请求中包含了自定义语言模型、自定义词汇表或自定义词汇表过滤器，请使用此选项。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    MediaFormat='flac',
    IdentifyLanguage=True,  (or IdentifyMultipleLanguages=True)
    LanguageOptions = [
        'en-US', 'hi-IN'
    ],
    LanguageIdSettings={
        'en-US': {
            'VocabularyName': 'my-en-US-vocabulary',
            'VocabularyFilterName': 'my-en-US-vocabulary-filter',
            'LanguageModelName': 'my-en-US-language-model'
        },
        'hi-IN': {
            'VocabularyName': 'my-hi-IN-vocabulary',
            'VocabularyFilterName': 'my-hi-IN-vocabulary-filter'
        }
    }
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

流式转录中的语言识别

流式转录语言识别可以识别您的媒体流中使用的主要语言。Amazon Transcribe 需要至少三秒钟的语音才能识别语言。

如果流仅包含一种语言，您可以启用单语言识别，这会识别媒体文件中使用的主要语言并仅使用该语言创建转录。

如果流包含多种语言，您可以启用多语言识别，这会识别流中使用的所有语言，并使用每种识别的语言创建转录。请注意，这将会生成多语言转录。您可以使用其它服务（例如 Amazon Transcribe）来翻译转录。

要使用流式转录语言识别，您必须提供至少两个语言代码，并且对于每个音频流，每种语言只能选择一种语言方言。这表示，您不能为同一个转录选择 en-US 和 en-AU 作为语言选项。

您还可以选择从您提供的一组语言代码中选择一种首选语言。添加首选语言可以加快语言识别过程，这对于简短的音频片段很有帮助。

> **Important**
>
> 如果您提供的语言代码与在您的音频中识别的一个或多个语言均不匹配，则 Amazon Transcribe 会从您指定的语言代码中选择最接近的语言匹配项。然后，它会生成该语言的转录。例如，如果您的媒体使用的是美国英语 (en-US)，而您为 Amazon Transcribe 提供了语言代码 zh-CN、fr-FR 和 de-DE，Amazon Transcribe 很可能会将您的媒体与德语 (de-DE) 匹配并生成德语转录。语言代码和说出的语言不匹配可能会导致转录不准确，因此我们建议在添加语言代码时要小心谨慎。

如果您的媒体包含两个声道，则 Amazon Transcribe 可以识别每个声道中使用的主要语言。在这种情况下，将 ChannelIdentification 参数设置为 true，每个声道将分别转录。注意，此参数的默认值为 false。如果您不对其进行更改，则只会转录第一个声道，并且只识别一种语言。

流式转录语言识别不能与自定义语言模型或编辑结合使用。如果将语言识别与其它特征结合使用，则只能使用这些特征支持的语言以及流式转录支持的语言。请参阅支持的语言。

> **Note**
>
> PCM 和 FLAC 是唯一支持的用于流式转录语言识别的音频格式。

识别多语言音频中的语言

多语言识别适用于多语言流，并为您提供反映流中使用的所有支持的语言的转录。这就表示，如果发言者在对话中变换语言，或者如果每个参与者说的是不同的语言，则您的转录输出会正确检测并转录每种语言。

例如，如果流包含交替使用美国英语 (en-US) 和印地语 (hi-IN) 的双语发言者，则多语言识别可以识别所说的美国英语并转录为 en-US，并将所说的印地语转录为 hi-IN。这不同于单语言识别，后者只使用一种主要语言来创建转录。在这种情况下，主要语言以外的任何语言都会被错误地转录。

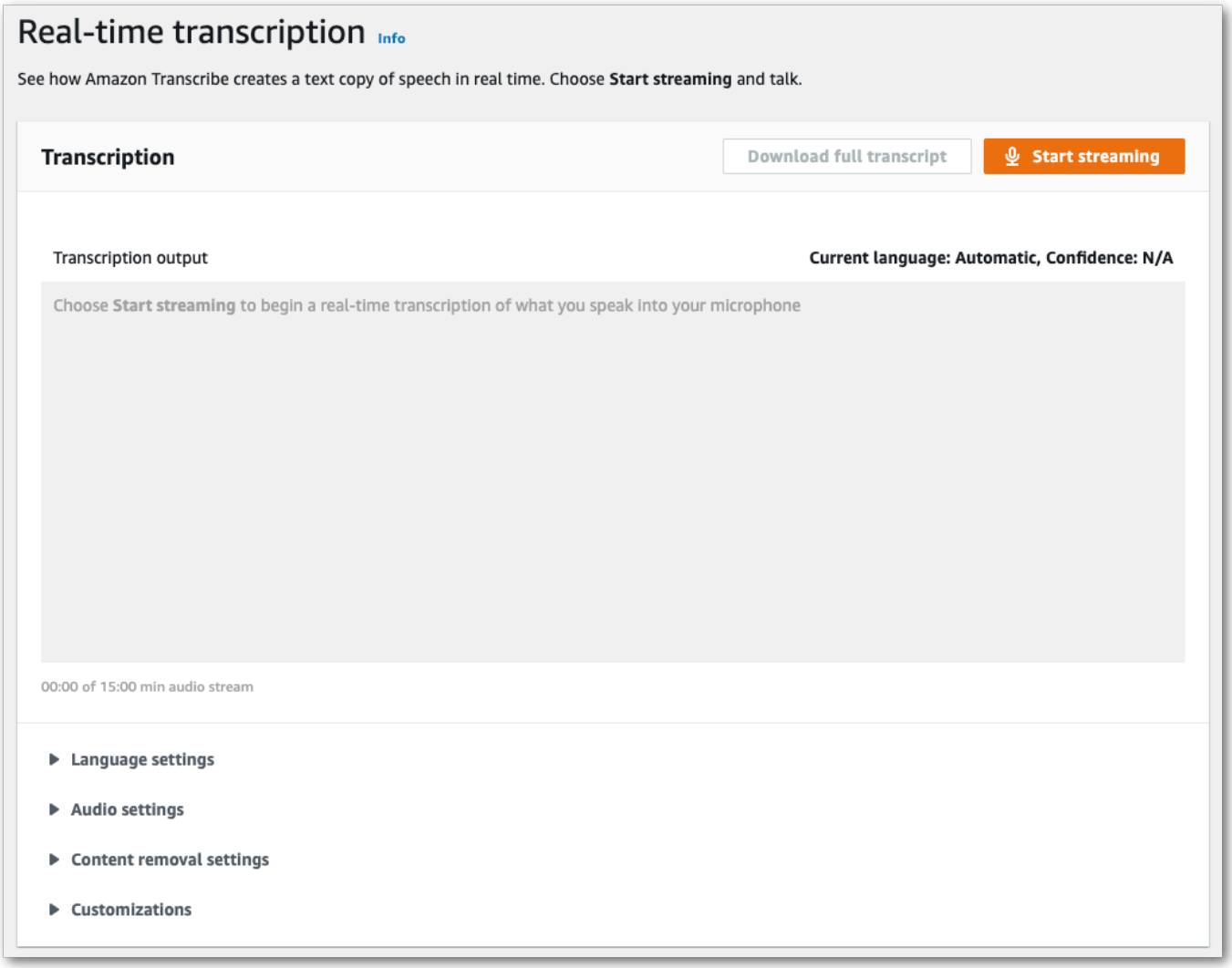
> **Note**
>
> 多语言识别目前不支持编辑和自定义语言模型。

在流式转录媒体中使用语言识别

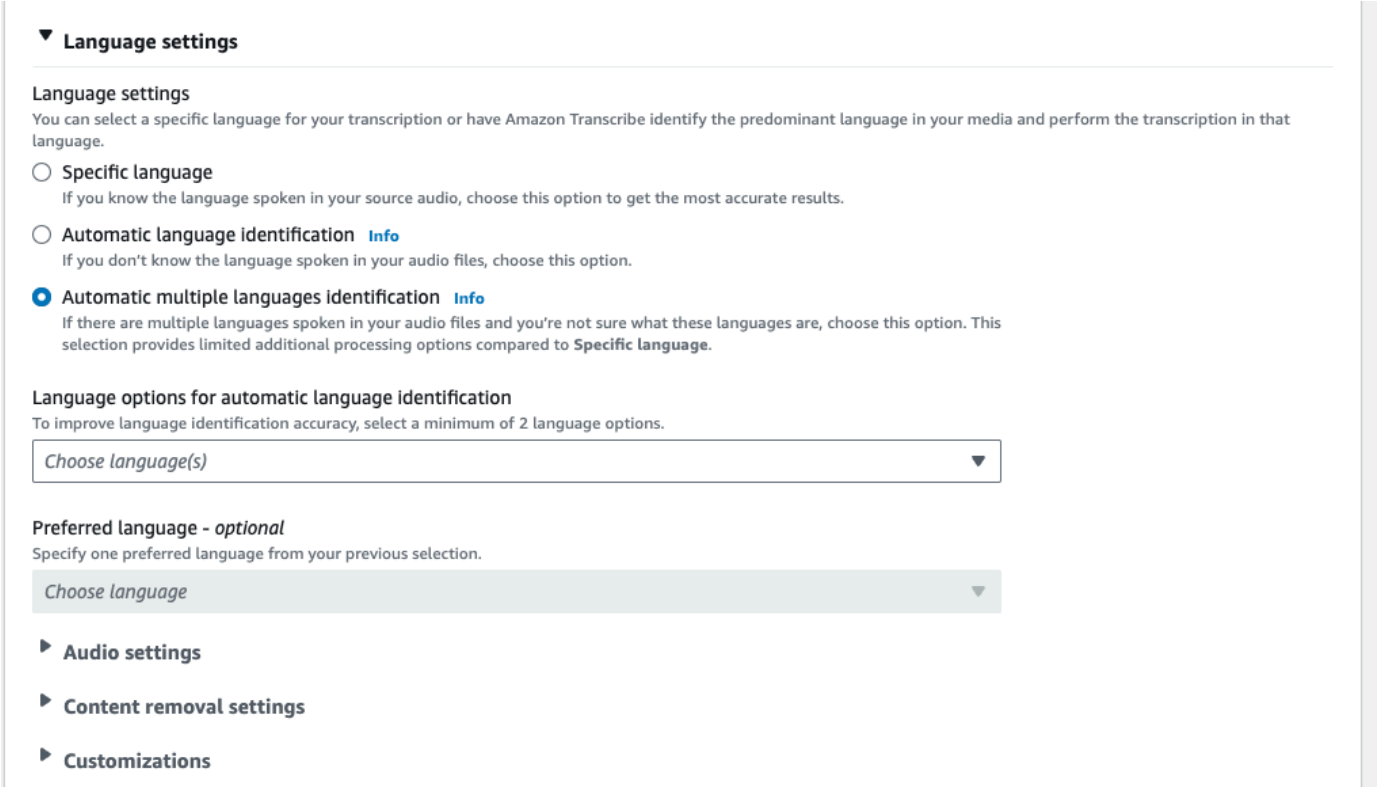
您可以通过 AWS Management Console、HTTP/2 或 WebSocket 在批量转录作业中使用自动语言识别；有关示例，请参阅以下内容：

AWS Management Console

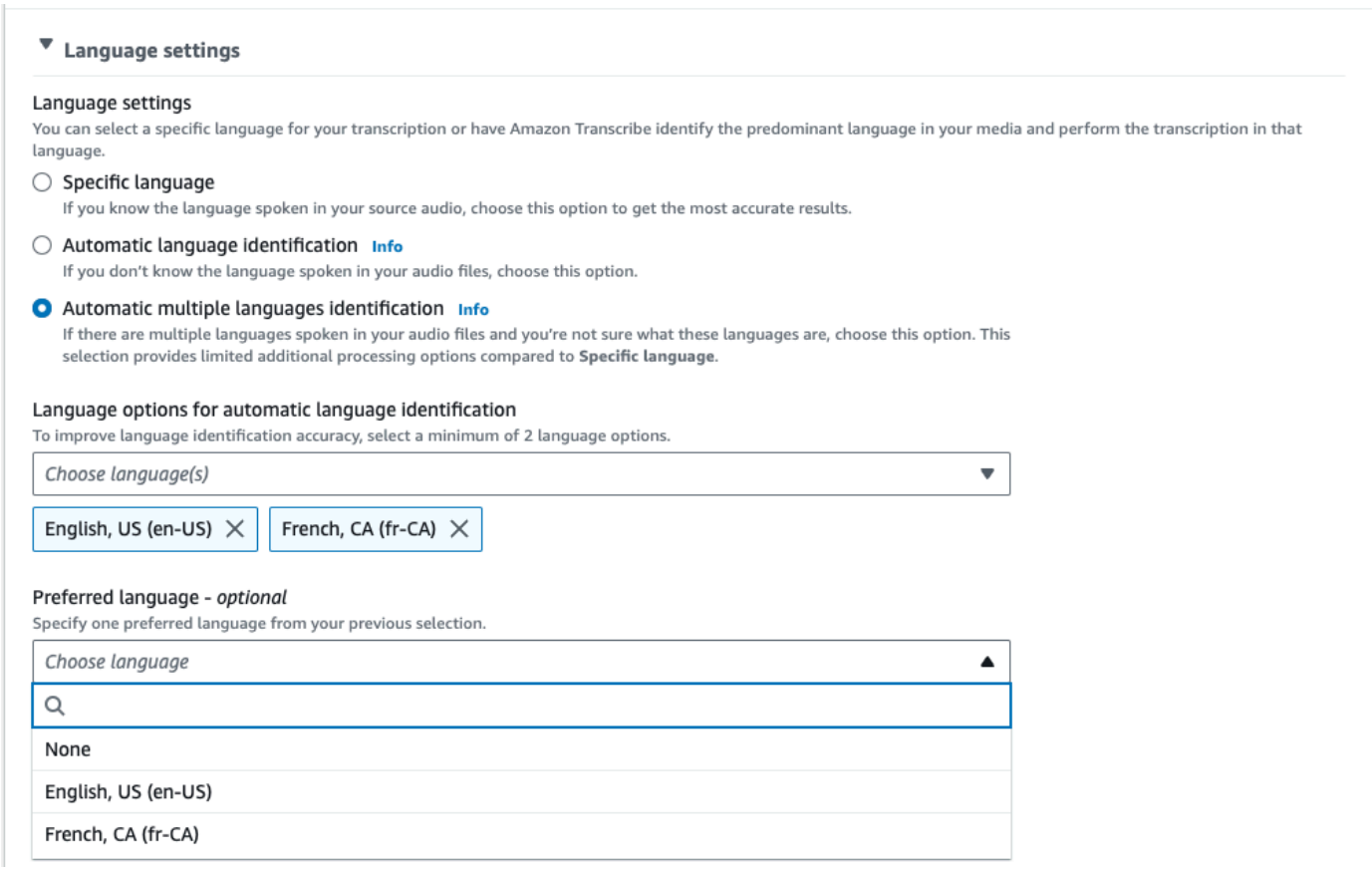
1. 登录到 AWS Management Console。
2. 在导航窗格中，选择 Real-time transcription (实时转录)。向下滚动到语言设置，如果该字段已最小化，则将其展开。



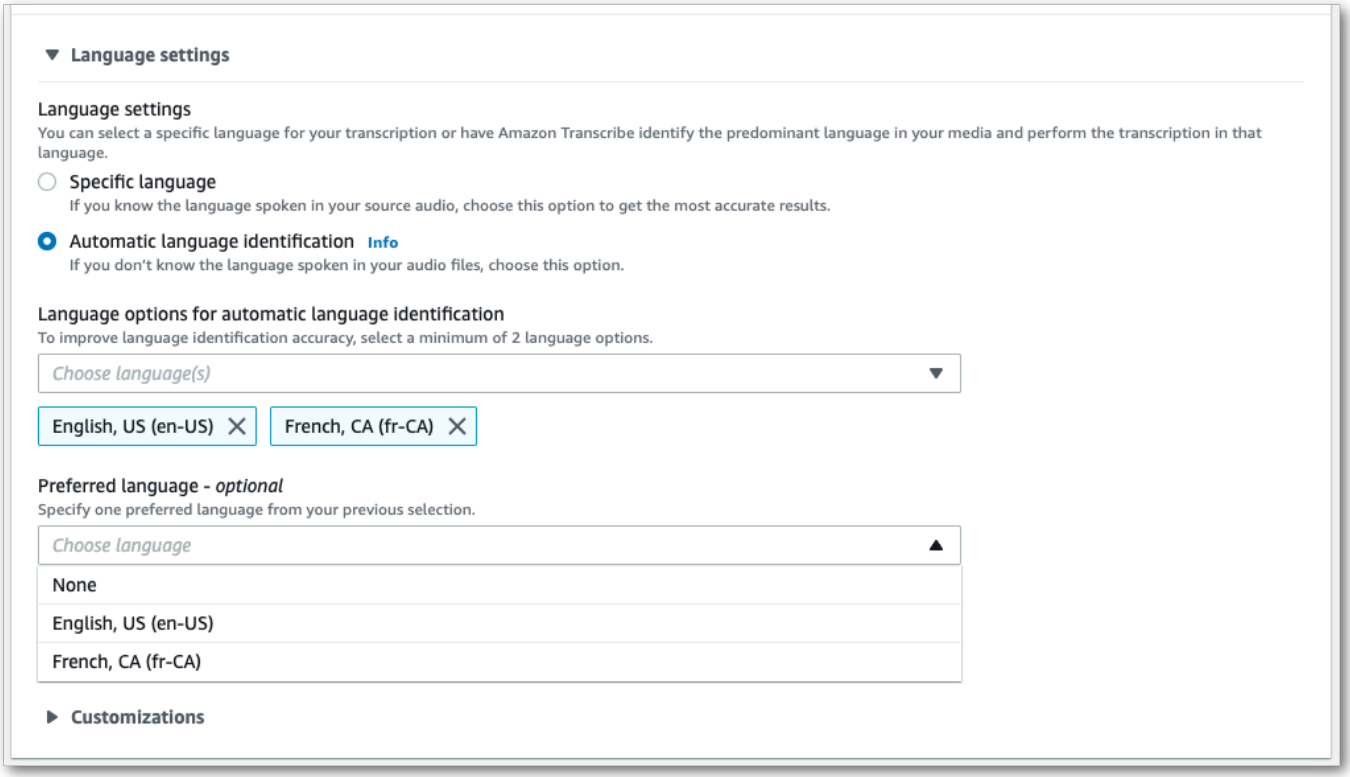
3. 选择自动语言识别或自动多语言识别。



4. 为您的转录提供至少两个语言代码。请注意，每种语言只能提供一种方言。例如，对于同一个转录，您不能同时选择 en-US 和 fr-CA 作为语言选项。



5. （可选）从您在上一步中选择的语言子集中，您可以为转录选择首选语言。



6. 您现在已准备就绪，可以转录音频流了。选择开始流式转录并开始讲话。要结束口述，请选择停止流式转录。

HTTP/2 音频流

该示例创建了一个启用语言识别的 HTTP/2 请求。有关结合 Amazon Transcribe 使用 HTTP/2 流式转录的更多信息，请参阅设置 HTTP/2 音频流。有关特定于 Amazon Transcribe 的参数和标题的更多详细信息，请参阅 StartStreamTranscription。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: 20220208T235959Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key/20220208/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-transcribe-identify-language: true
x-amzn-transcribe-language-options: en-US,de-DE
x-amzn-transcribe-preferred-language: en-US
transfer-encoding: chunked
```

该示例创建一个启用了多语言识别的 HTTP/2 请求。有关结合 Amazon Transcribe 使用 HTTP/2 流式转录的更多信息，请参阅设置 HTTP/2 音频流。有关特定于 Amazon Transcribe 的参数和标题的更多详细信息，请参阅 StartStreamTranscription。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: 20220208T235959Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key/20220208/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-transcribe-identify-multiple-languages: true
```

```
x-amzn-transcribe-language-options: en-US,de-DE
x-amzn-transcribe-preferred-language: en-US
transfer-encoding: chunked
```

如果在请求中使用 identify-language 或 identify-multiple-languages，您还必须包含 language-options。不能在同一个请求中同时使用 language-code 和 identify-language。

参数定义可在 API 参考中找到；所有 AWS API 操作的通用参数列在常见参数部分中。

WebSocket 音频流

该示例创建了一个在 WebSocket 音频流中使用语言识别的预签名 URL。为了便于阅读，已增加了换行符。有关结合 Amazon Transcribe 使用 WebSocket 音频流的更多信息，请参阅设置直 WebSocket 播。有关参数的更多详细信息，请参阅 StartStreamTranscription。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&media-encoding=flac
&sample-rate=16000
&identify-language=true
&language-options=en-US,de-DE
&preferred-language=en-US
```

该示例创建一个在 WebSocket 流中使用多语言识别的预签名 URL。为了便于阅读，已增加了换行符。有关结合 Amazon Transcribe 使用 WebSocket 音频流的更多信息，请参阅设置直 WebSocket 播。有关参数的更多详细信息，请参阅 StartStreamTranscription。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
```

```
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&media-encoding=flac
&sample-rate=16000
&identify-multiple-languages=true
&language-options=en-US,de-DE
&preferred-language=en-US
```

如果在请求中使用 `identify-language` 或 `identify-multiple-languages`，您还必须包含 `language-options`。不能在同一个请求中同时使用 `language-code` 和 `identify-language`。

参数定义可在 API 参考中找到；所有 AWS API 操作的通用参数列在常见参数部分中。

替代转录

Amazon Transcribe转录音频时，它会创建同一脚本的不同版本，并为每个版本分配置信度分数。在典型的转录中，你只能得到置信度分数最高的版本。

如果您开启备用转录，则Amazon Transcribe返回可信度较低的其他成绩单版本。您最多可以选择返回 10 个备用转录。如果您指定的备选方案数量多于所Amazon Transcribe识别的替代项数，则仅返回替代项的实际数量。

所有备选方案都位于同一个转录输出文件中，并在片段级别上呈现。片段是语音中的自然暂停，例如更换扬声器或音频暂停。

替代转录仅适用于批量转录。

您的转录输出的结构如下。省略号 (`...`) 为简洁起见，在代码示例中指明了内容在哪里被删除。

1. 给定片段的完整最终转录。

```
"results": {
    "language_code": "en-US",
    "transcripts": [
        {
            "transcript": "The amazon is the largest rainforest on the planet."
        }
    ],
```

2. 上一`transcript`节中每个单词的置信度分数。

```
"items": [
    {
        "start_time": "1.15",
        "end_time": "1.35",
        "alternatives": [
            {
                "confidence": "1.0",
                "content": "The"
            }
        ],
        "type": "pronunciation"
    },
    {
```

```
        "start_time": "1.35",
        "end_time": "2.05",
        "alternatives": [
            {
                "confidence": "1.0",
                "content": "amazon"
            }
        ],
        "type": "pronunciation"
    },
```

3. 您的替代转录位于转录输出的segments部分。每个区段的备选方案按置信度分数降序排序。

```
"segments": [
            {
                "start_time": "1.04",
                "end_time": "5.065",
                "alternatives": [
                    {
                ...
                        "transcript": "The amazon is the largest rain forest on the
 planet.",
                        "items": [
                            {
                             "start_time": "1.15",
                                "confidence": "1.0",
                                "end_time": "1.35",
                                "type": "pronunciation",
                                "content": "The"
                            },
                            ...
                            {
                                "start_time": "3.06",
                                "confidence": "0.0037",
                                "end_time": "3.38",
                                "type": "pronunciation",
                                "content": "rain"
                            },
                            {
                                "start_time": "3.38",
                                "confidence": "0.0037",
                                "end_time": "3.96",
                                "type": "pronunciation",
                                "content": "forest"
```

```
                                },
```

4. 转录输出末尾的状态。

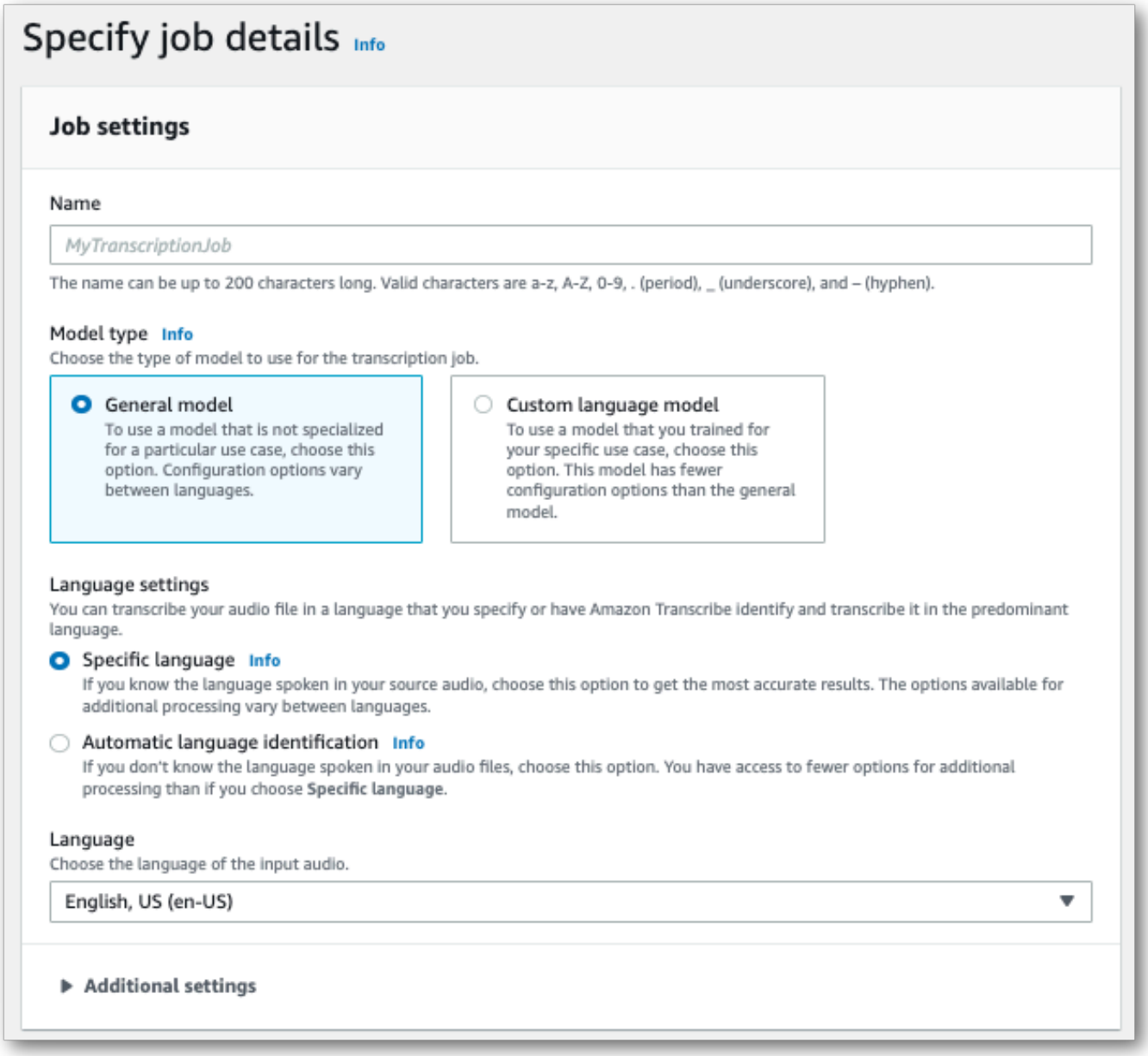
```
"status": "COMPLETED"
}
```

请求替代转录

您可以使用AWS Management Console、AWS CLI或 AWSSDK 请求备用转录；有关示例，请参阅以下内容：

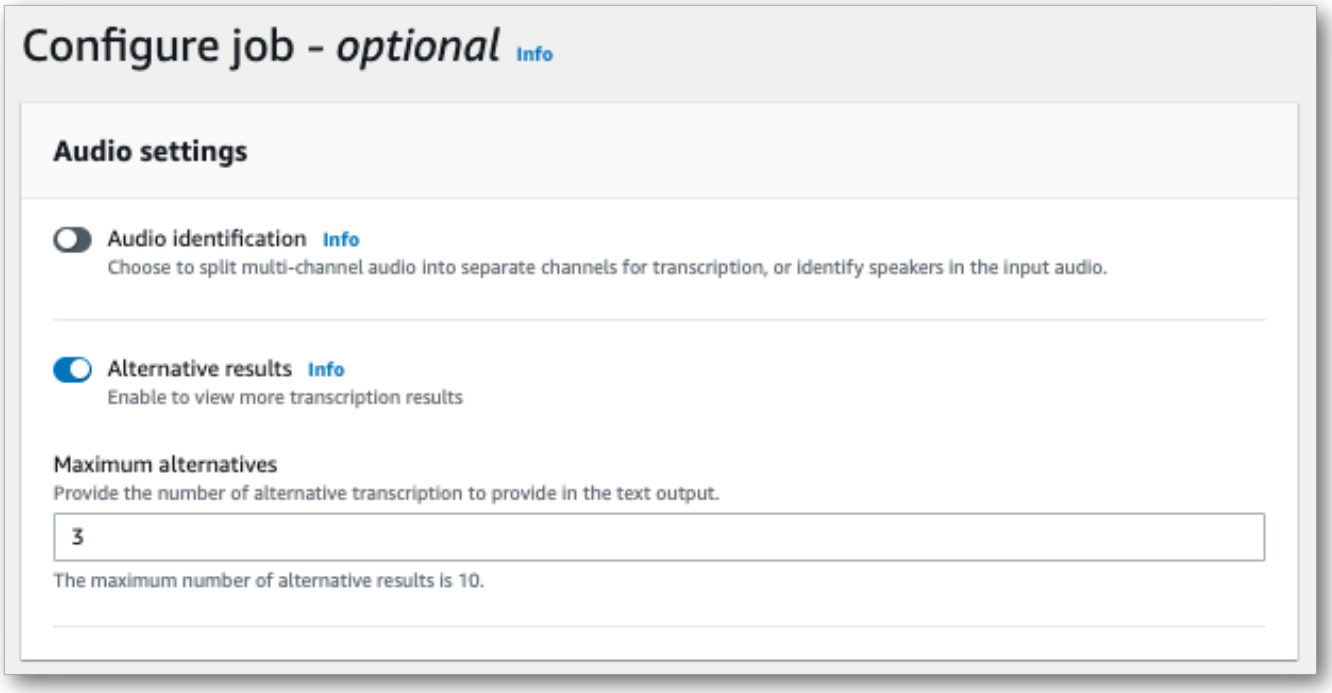
AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。



3. 在“指定作业详细信息”页面上填写要包含的所有字段，然后选择下一步。这将带您进入配置作业-可选页面。

   选择替代结果并指定您想要在成绩单中显示的最大替代转录结果数。



4. 选择 “创建作业” 以运行转录作业。

AWS CLI

此示例使用start-transcription-job命令和ShowAlternatives参数。有关更多信息，请参阅 StartTranscriptionJob 和 ShowAlternatives。

请注意，如果您在请求ShowAlternatives=true中包括，则还必须包括MaxAlternatives。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
--settings ShowAlternatives=true,MaxAlternatives=4
```

这是使用start-transcription-job命令的另一个示例，以及包含替代转录的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
```

```
--cli-input-json file://filepath/my-first-alt-transcription-job.json
```

文件 my-first-alt-transcription-job.json 包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
   },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "Settings": {
        "ShowAlternatives": true,
        "MaxAlternatives": 4
   }
}
```

AWS SDK for Python (Boto3)

以下示例使用 start_transcription_job 方法的ShowAlternatives参数来请求替代转录。AWS SDK for Python (Boto3)有关更多信息，请参阅 StartTranscriptionJob 和 ShowAlternatives。

有关使用AWS软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS章。

请注意，如果您在请求'ShowAlternatives':True中包括，则还必须包括MaxAlternatives。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
```

```
    Settings = {
        'ShowAlternatives':True,
        'MaxAlternatives':4
    }
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

使用自定义词汇和自定义语言模型提高转录准确性

如果您的媒体包含特定领域或非标准术语，例如品牌名称、首字母缩略词、技术词汇和行话，则Amazon Transcribe可能无法在转录输出中正确捕获这些术语。

要更正转录错误并针对特定用例自定义输出，您可以创建自定义词汇和自定义语言模型。

- 自定义词汇旨在调整和提高所有上下文中特定单词的识别和格式。这包括提供单词Amazon Transcribe以及（可选）发音和显示表单。

  如果Amazon Transcribe无法正确呈现成绩单中的特定术语，您可以创建一个自定义词汇表文件，告诉您希望Amazon Transcribe如何显示这些术语。这种针对特定单词的方法最适合更正品牌名称和首字母缩略词等术语。
- 自定义语言模型旨在捕捉与术语相关的上下文。这包括Amazon Transcribe提供大量特定领域的文本数据。

  如果Amazon Transcribe无法正确呈现技术术语或在成绩单中使用了不正确的同音异音，则可以创建自定义语言模型Amazon Transcribe来教授您的领域特定语言。例如，自定义语言模型可以学习何时使用 “floe”（浮冰）与 “流动”（线性流）。

  这种情境感知方法最适合转录大量特定领域的语音。与单独的自定义词汇相比，自定义语言模型可以显著提高准确性。使用批量转录时，您可以在请求中包含自定义语言模型和自定义词汇表。

> **Tip**
>
> 要实现最高的转录准确性，请将自定义词汇与您的自定义语言模型结合使用。

有关如何使用创建自定义词汇表的视频演示AWS Management Console，请参阅使用自定义词汇表。

有关如何创建和使用自定义语言模型的视频演示，请参阅使用自定义语言模型 (CLM) 提高转录准确性。

> **通过Machine LearnAWS ing 博客深入了解**
>
> Custom 词汇：
>
> - 使用 F1 比赛的实时转录Amazon Transcribe

自定义语言模型：

- 构建自定义语言模型以增强 speech-to-text 性能Amazon Transcribe
- 使用以下自定义语言模型提高课堂讲座的转录准确性Amazon Transcribe

自定义词汇

使用自定义词汇来提高一个或多个特定单词的转录准确性。这些通常是特定领域的术语，例如品牌名称和首字母缩略词、专有名词以及无法正确呈现的单词。Amazon Transcribe

自定义词汇表可用于所有支持的语言。请注意，在自定义词汇表中只能使用您语言的字符集中列出的字符。

Important

使用时，您对自己的数据的完整性负责Amazon Transcribe。不要在自定义词汇表中输入机密信息、个人信息 (PII) 或受保护的健康信息 (PHI)。

创建自定义词汇表时的注意事项：

- 您在每个词汇文件 AWS 账户
- 每个自定义词汇文件的大小限制为 50 Kb
- 如果使用 API 创建自定义词汇表，则您的词汇文件必须采用文本 (*.txt) 格式。如果使用AWS Management Console，则您的词汇文件可以是文本 (*.txt) 格式或逗号分隔值 (*.csv) 格式。
- 自定义词汇表中的每个条目不能超过 256 个字符
- 必须事先已在转录中创建 Custom 词汇。AWS 区域

Tip

您可以使用来测试您的自定义词汇表AWS Management Console。当您的自定义词汇准备就绪可供使用后，登录，选择实时转录，滚动到自定义，打开自定义词汇表，然后从下拉列表中选择您的自定义词汇表。AWS Management Console然后选择 “开始直播”。对着麦克风说出自定义词汇中的一些单词，看看它们是否能正确呈现。

自定义词汇表与列表

> **Important**
>
> 列表格式的自定义词汇已过时。如果您要创建新的自定义词汇表，请使用表格格式。

表格为自定义词汇表中单词的输入和输出提供了更多选项，并且可以更好地控制单词的输入和输出。对于表格，必须指定多个类别(Phrase, IPA, SoundsLike, and DisplayAs)，以便对输出进行微调。

列表没有其他选项，因此您只能键入您希望它们出现在脚本中的条目，将所有空格替换为连字符。

AWS Management ConsoleAWS CLI、和 AWS SDK 都以相同的方式使用自定义词汇表；每种方法的列表使用方式不同，因此可能需要额外的格式才能在方法之间成功使用。

有关更多信息，请参阅 使用表格创建自定义词汇表 和 使用列表创建自定义词汇。

要更深入地了解如何使用带有自定义词汇表的 Amazon Aummented AI，请参阅开始构建Augmented AI 点评以及 Amazon Transcribe

> **特定于自定义词汇表的 API 操作**
>
> `CreateVocabulary`, `DeleteVocabulary`, `GetVocabulary`, `ListVocabularies`, `UpdateVocabulary`

使用表格创建自定义词汇表

使用表格格式是创建自定义词汇表的首选方式。词汇表表格必须由四列 (Phrase, SoundsLike, IPA, and DisplayAs) 组成，可以按任意顺序排列：

Phrase	SoundsLike	IPA	DisplayAs
必需。表格中的每一行都必须在此列中包含一个条目。 请勿在此列中使用空格。	可选。该列中的行应保留空白，因为 Amazon Transcribe 不再需要使用该信息准确转录相应的短语。将来不再支持该列。	可选。该列中的行应保留空白，因为 Amazon Transcribe 不再需要使用该信息准确转录相应的短语。将来不再支持该列。	可选。此列中的行可以留空。 您可以在此列中使用空格。

Phrase	SoundsLike	IPA	DisplayAs
如果您的条目包含多个单词，请用连字符 (-) 分隔每个单词。例如，**Andorra-la-Vella** 或 **Los-Angeles** 。 对于首字母缩略词，任何发音的字母都必须用句点分隔。尾部句点也需要发音。如果您的首字母缩略词是复数，则必须在首字母缩略词和“s”之间使用连字符。例如，“CLI”是 **C.L.I.**（不是 **C.L.I**），“ABC”是 **A.B.C.-s**（不是 **A.B.C-s**）。 如果您的短语由单词和首字母缩略词组成，则这两个部分必须用连字符分隔。例如，“DynamoDB”是 **Dynamo-D.B.** 。 请勿在此列中包含数字；数值必须拼写出来。例如，“VX02Q”是 **V.X.-zero-two-Q.**。	如果您决定指定值，请不要在该列中使用空格。您不能在同一行中同时输入 SoundsLike 和 IPA 的条目。	如果您决定指定值，您必须在每个 IPA 字符（单字节）或有效的 IPA 字符对（双字节）之间添加一个空格。您不能在同一行中同时输入 IPA 和 SoundsLike 的条目。	定义您希望条目在转录输出中显示的方式。例如，Phrase 列中的 **Andorra-la-Vella** 在 DisplayAs 列中是 **Andorra la Vella**。 如果此列中的某行为空，则 Amazon Transcribe 使用 Phrase 列的内容来确定输出。 您可以在此列中包含数字 (0-9)。

创建表格时需要注意的事项：

- 您的表必须包含所有四列 (Phrase, SoundsLike, IPA, and DisplayAs)，但 `Phrase` 列是唯一必须在每行包含一个条目的列。所有其它列都可以留空。
- 每列必须以 TAB 或逗号 (,) 分隔；这适用于自定义词汇表文件中的每一行。如果一行包含空列，则仍必须为每列包含一个分隔符（TAB 或逗号）。
- 只允许在 `IPA` 和 `DisplayAs` 列中使用空格。不要使用空格分隔列。
- 不再需要 `IPA` 和 `SoundsLike` 条目，但现在仍然需要这些列标题。Amazon Transcribe 不再需要使用该信息准确转录相应的短语，并且将来不再支持这些列。如果决定添加条目，您不能在给定行中同时包含 `IPA` 和 `SoundsLike` 字段的条目。请选择其中的一个字段。
- `DisplayAs` 列支持符号和特殊字符（例如，C++）。所有其它列都支持您的语言的字符集页面上所列的字符。
- 如果要在 `Phrase` 列中包含数值，则必须将其拼写出来。仅 `DisplayAs` 列支持数字 (`0-9`)。
- 您必须将表格另存为 `LF` 格式的纯文本 (*.txt) 文件。如果您使用任何其它格式，例如 `CRLF`，则无法处理您的自定义词汇表。
- 您必须将自定义词汇表文件上传到 Amazon S3 存储桶中并使用 `CreateVocabulary` 对其进行处理，然后才能将其包含在转录请求中。有关说明，请参阅 创建自定义词汇表表格。

> **Note**
>
> 输入首字母缩略词，或者其字母应以用点 (**`A.B.C.`**) 隔开的单个字母的形式单独发音的其它单词。要输入首字母缩略词的复数形式（如“ABCs”），请用连字符 (**`A.B.C.-s`**) 将“s”与首字母缩略词隔开。您可以使用大写或小写字母定义首字母缩略词。并非所有语言都支持首字母缩略词；请参阅支持的语言和特定语言的特征。

以下是自定义词汇表表格示例（其中 **`[TAB]`** 表示制表符）：

```
Phrase[TAB]SoundsLike[TAB]IPA[TAB]DisplayAs
Los-Angeles[TAB][TAB][TAB]Los Angeles
Eva-Maria[TAB][TAB][TAB]
A.B.C.-s[TAB][TAB][TAB]ABCs
Amazon-dot-com[TAB][TAB][TAB]Amazon.com
C.L.I.[TAB][TAB][TAB]CLI
Andorra-la-Vella[TAB][TAB][TAB]Andorra la Vella
Dynamo-D.B.[TAB][TAB][TAB]DynamoDB
V.X.-zero-two[TAB][TAB][TAB]VX02
V.X.-zero-two-Q.[TAB][TAB][TAB]VX02Q
```

为了清晰起见，以下是列对齐的同一张表格。请勿在自定义词汇表表格中的列之间添加空格；您的表格应该像前面的示例一样看上去未对齐。

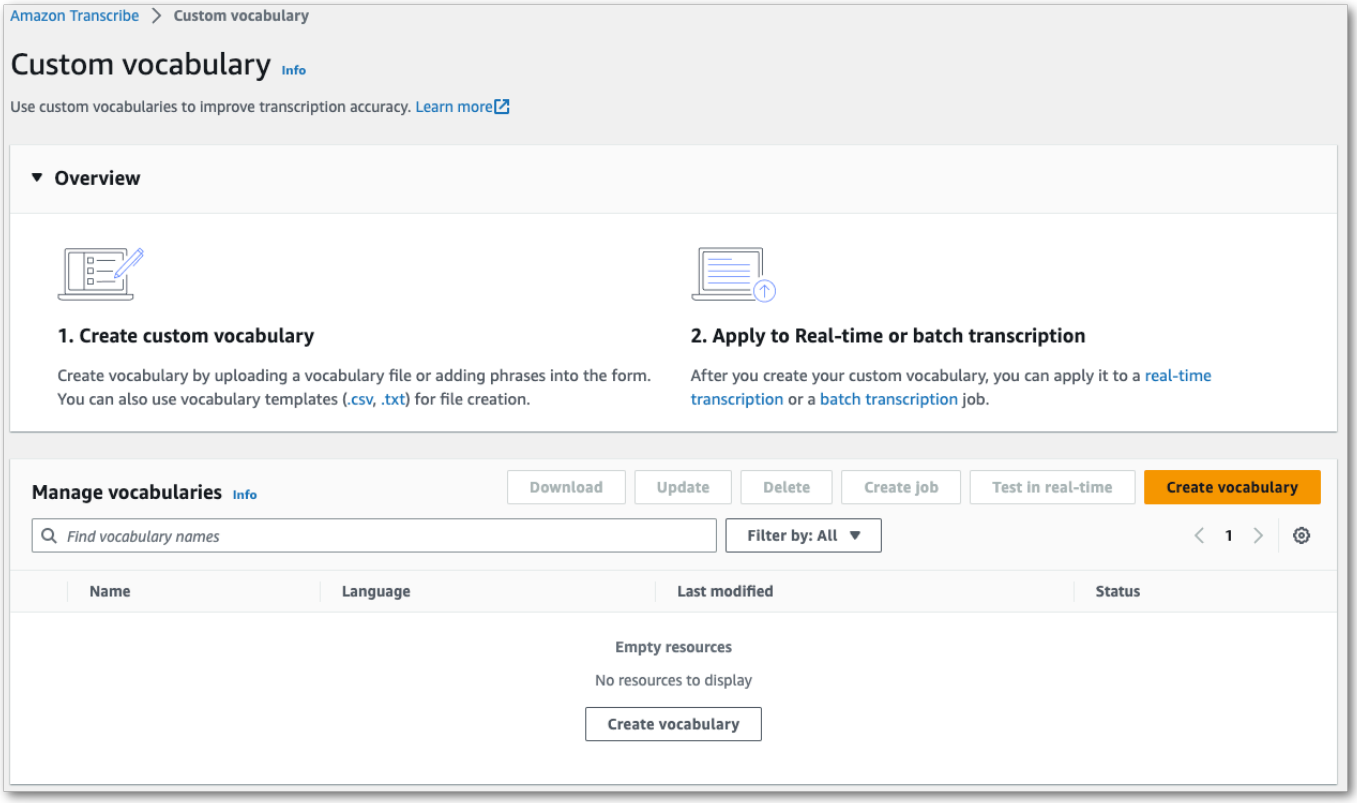
```
Phrase          [TAB]SoundsLike         [TAB]IPA                [TAB]DisplayAs
Los-Angeles     [TAB]                   [TAB]                   [TAB]Los Angeles
Eva-Maria       [TAB]                   [TAB]                   [TAB]
A.B.C.-s        [TAB]                   [TAB]                   [TAB]ABCs
amazon-dot-com  [TAB]                   [TAB]                   [TAB]amazon.com
C.L.I.          [TAB]                   [TAB]                   [TAB]CLI
Andorra-la-Vella[TAB]                   [TAB]                   [TAB]Andorra la Vella
Dynamo-D.B.     [TAB]                   [TAB]                   [TAB]DynamoDB
V.X.-zero-two   [TAB]                   [TAB]                   [TAB]VX02
V.X.-zero-two-Q.[TAB]                   [TAB]                   [TAB]VX02Q
```

创建自定义词汇表表格

要处理用于 Amazon Transcribe 的自定义词汇表表格，请参阅以下示例：

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择自定义词汇表。此时将会打开自定义词汇表页面，您可以在该页面中查看现有词汇表或创建新词汇表。
3. 选择创建词汇表。

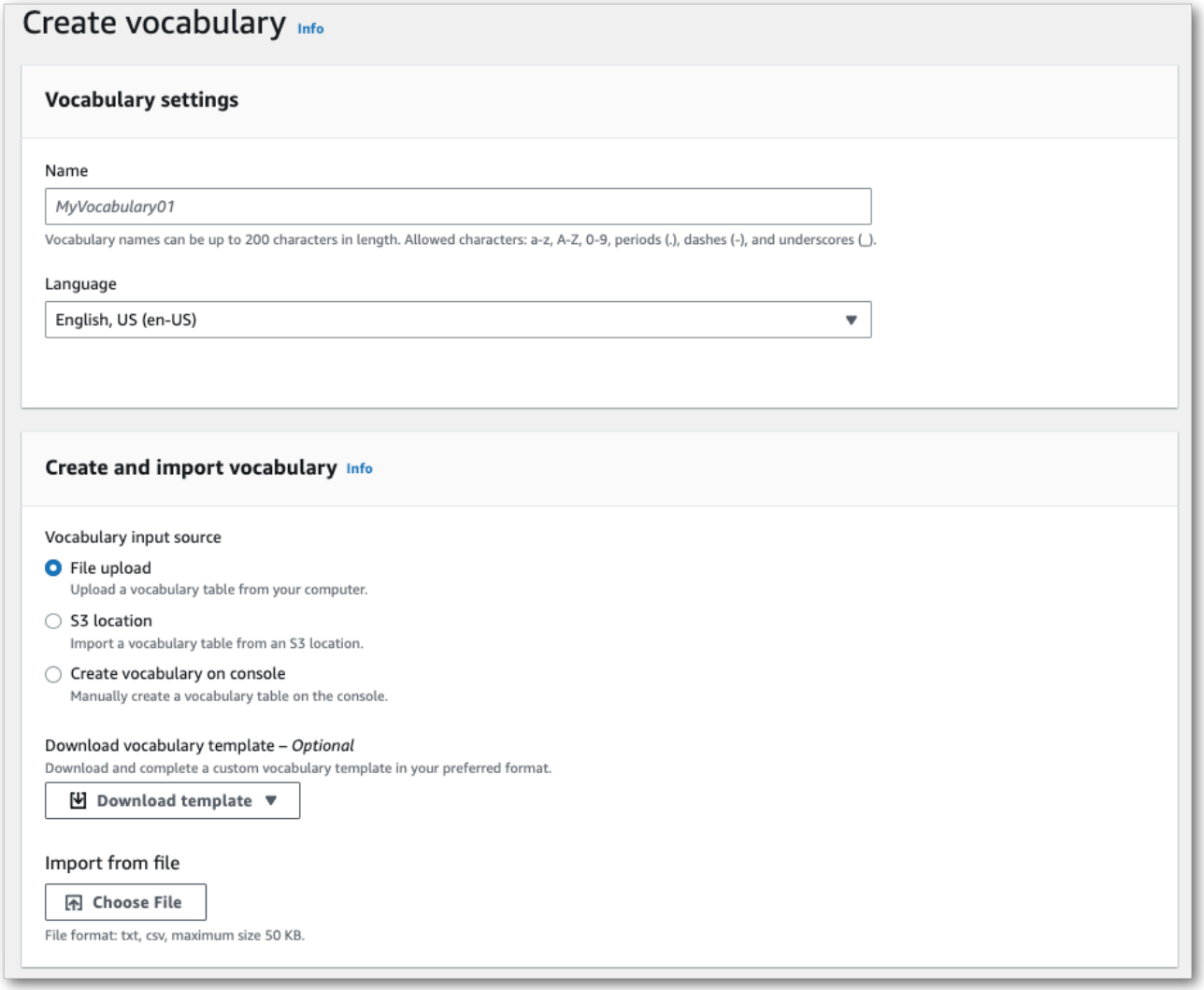


这会将您带至创建词汇表页面。为新的自定义词汇表输入名称。

在此，您有三种选择：

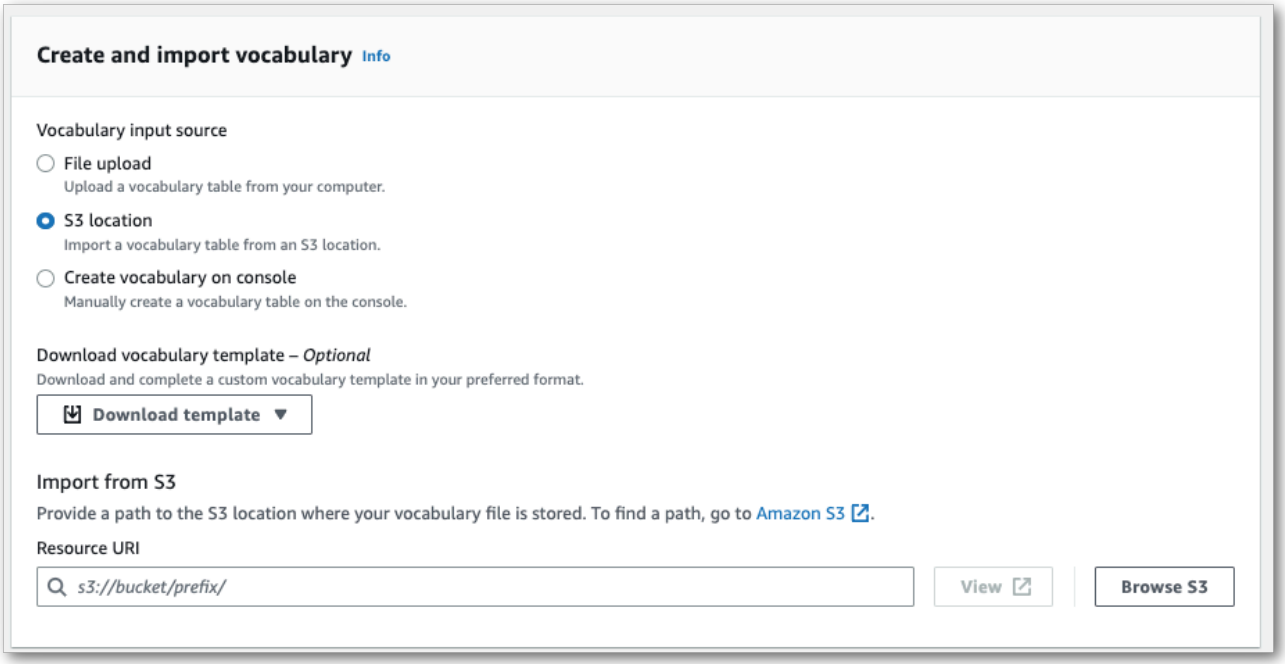
a. 从您的计算机上传 txt 或 csv 文件。

   您可以从头开始创建自定义词汇表，也可以下载模板来帮助您入门。然后，您的词汇表将自动填充到查看和编辑词汇表窗格中。



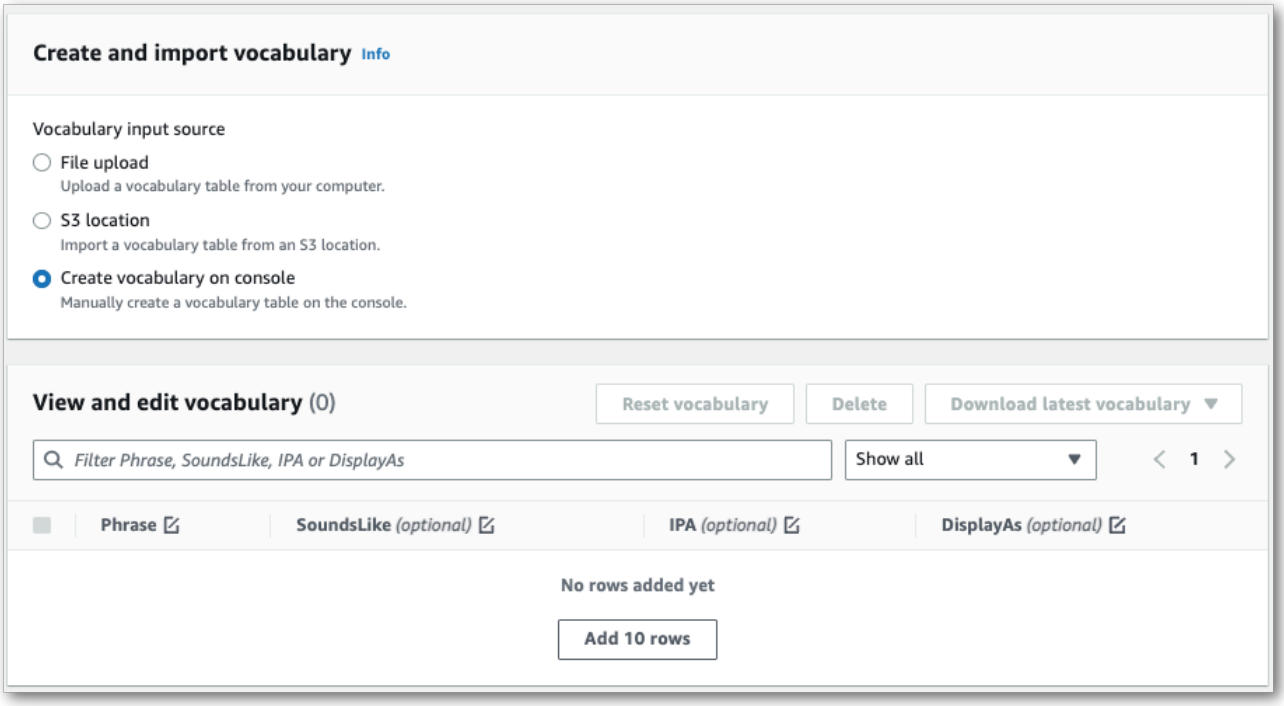
b. 从某个 Amazon S3 位置导入 txt 或 csv 文件。

   您可以从头开始创建自定义词汇表，也可以下载模板来帮助您入门。将完成的词汇表文件上传到 Amazon S3 存储桶，并在请求中指定其 URI。然后，您的词汇表将自动填充到查看和编辑词汇表窗格中。

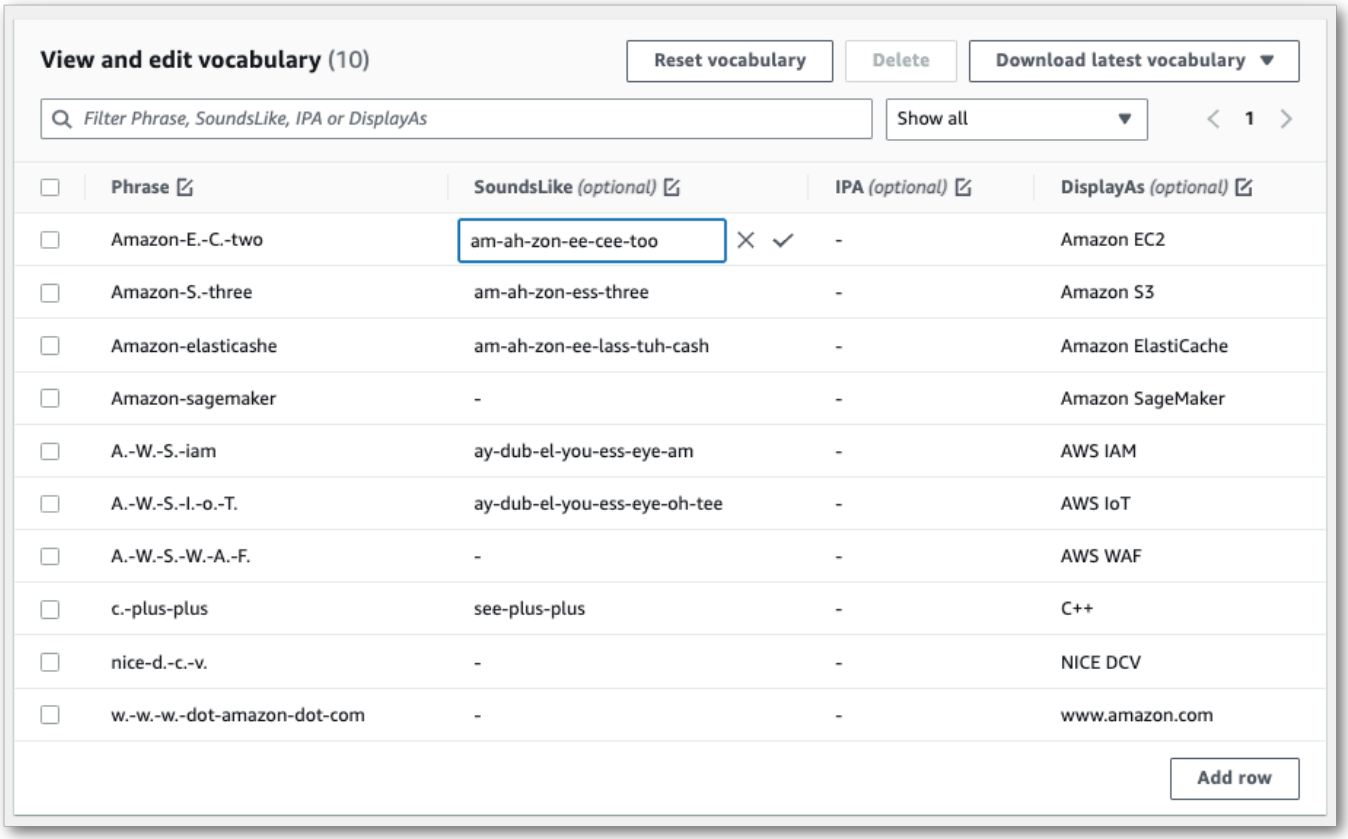


c. 在控制台中手动创建词汇表。

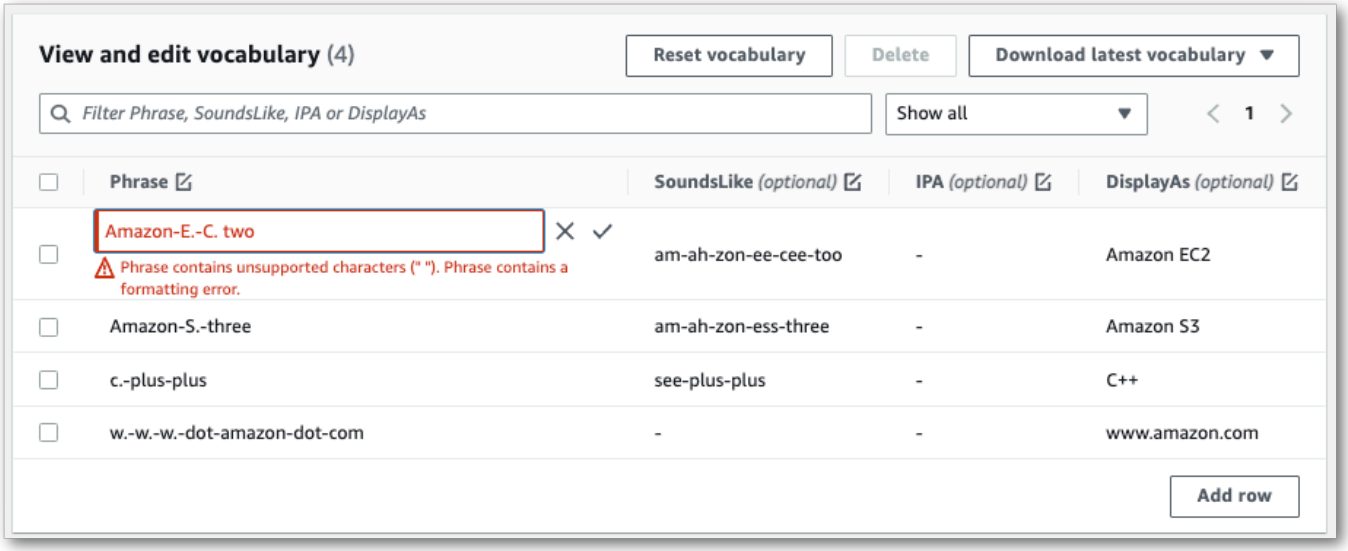
滚动到查看和编辑词汇表窗格并选择添加 10 行。现在，您可以手动输入术语。



4. 您可以在查看和编辑词汇表窗格中编辑您的词汇表。要进行更改，请单击要修改的条目。

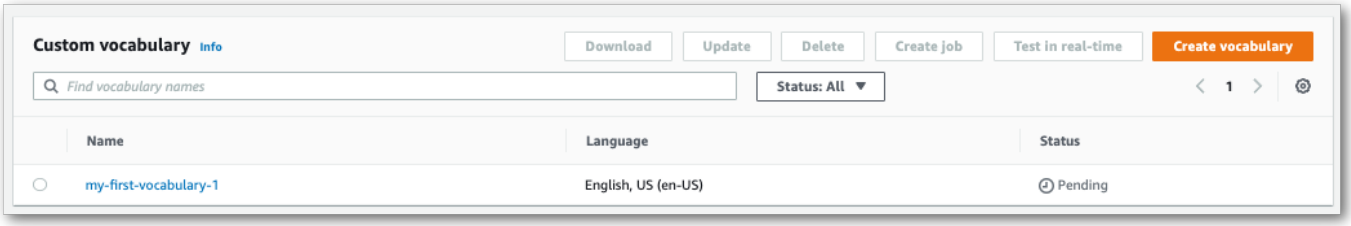


如果您遇到错误，会收到一条详细的错误消息，这样您就可以在处理词汇表之前更正所有问题。请注意，如果您在选择创建词汇表之前没有更正所有错误，则您的词汇表请求将失败。

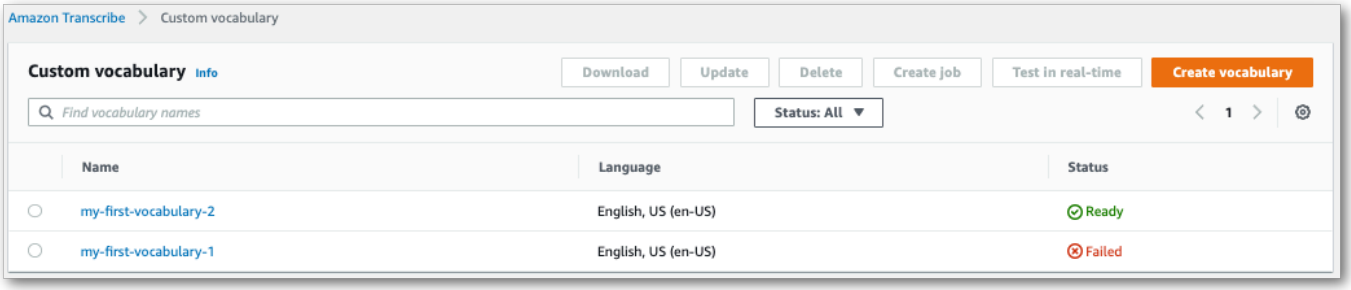


选择对勾标记 (✓) 保存您的更改，或选择“X”放弃更改。

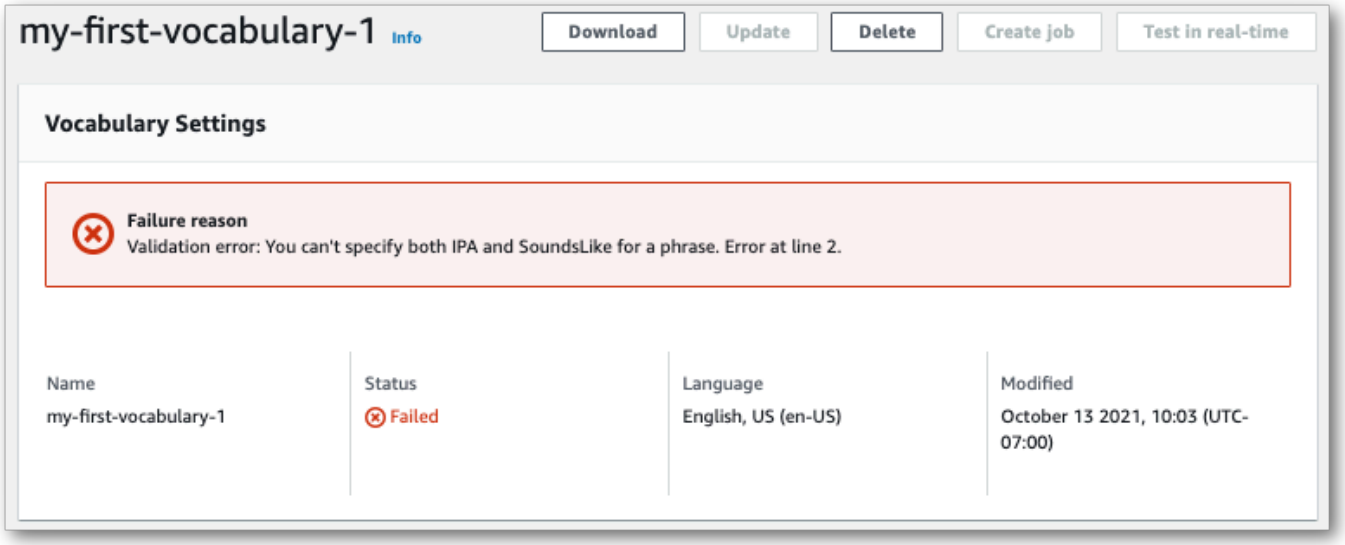
5. 您也可以为自定义词汇表添加标签。完成所有字段并对自己的词汇表感到满意后，请选择页面底部的创建词汇表。此时您将看到自定义词汇表页面，您可以在该页面中查看自定义词汇表的状态。当状态从“待处理”变为“就绪”时，您的自定义词汇表就可以用于转录了。



6. 如果状态更改为“失败”，请选择您的自定义词汇表的名称以进入其信息页面。



本页顶部有一个失败原因横幅，提供了有关您的自定义词汇表失败原因的信息。请更正文本文件中的错误，然后重试。



AWS CLI

该示例结合表格格式的词汇表文件使用 create-vocabulary 命令。有关更多信息，请参阅CreateVocabulary。

要在转录作业中使用现有的自定义词汇表，请在调用 StartTranscriptionJob 操作时在 Settings 字段中设置 VocabularyName，或者从 AWS Management Console中的下拉列表中选择自定义词汇表。

```
aws transcribe create-vocabulary \
--vocabulary-name my-first-vocabulary \
--vocabulary-file-uri s3://DOC-EXAMPLE-BUCKET/my-vocabularies/my-vocabulary-file.txt \
--language-code en-US
```

以下是另一个使用 create-vocabulary 命令的示例，以及创建自定义词汇表的请求正文。

```
aws transcribe create-vocabulary \
--cli-input-json file://filepath/my-first-vocab-table.json
```

my-first-vocab-table.json 文件包含以下请求正文。

```
{
  "VocabularyName": "my-first-vocabulary",
  "VocabularyFileUri": "s3://DOC-EXAMPLE-BUCKET/my-vocabularies/my-vocabulary-
table.txt",
  "LanguageCode": "en-US"
}
```

VocabularyState 从 PENDING 变为 READY 后，您的自定义词汇表就可以用于转录了。要查看自定义词汇表的当前状态，请运行：

```
aws transcribe get-vocabulary \
--vocabulary-name my-first-vocabulary
```

AWS SDK for Python (Boto3)

该示例通过 create_vocabulary 方法使用 AWS SDK for Python (Boto3) 以表格格式创建自定义词汇表。有关更多信息，请参阅CreateVocabulary。

要在转录作业中使用现有的自定义词汇表，请在调用 StartTranscriptionJob 操作时在 Settings 字段中设置 VocabularyName，或者从 AWS Management Console中的下拉列表中选择自定义词汇表。

有关使用 AWS SDK 的其它示例，包括特定特征、场景和跨服务示例，请参阅使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS一章。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
vocab_name = "my-first-vocabulary"
response = transcribe.create_vocabulary(
    LanguageCode = 'en-US',
    VocabularyName = vocab_name,
    VocabularyFileUri = 's3://DOC-EXAMPLE-BUCKET/my-vocabularies/my-vocabulary-
table.txt'
)

while True:
    status = transcribe.get_vocabulary(VocabularyName = vocab_name)
    if status['VocabularyState'] in ['READY', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

Note

如果您为自定义词汇表文件创建新的 Amazon S3 存储桶，请确保发出 CreateVocabulary 请求的 IAM 角色有权访问此存储桶。如果角色不具备相应的权限，则请求将失败。您可以选择在请求中包含 DataAccessRoleArn 参数来指定 IAM 角色。有关 Amazon Transcribe 中的 IAM 角色和策略的更多信息，请参阅Amazon Transcribe 基于身份的策略示例。

使用列表创建自定义词汇

Important

列表格式的自定义词汇表已过时，因此，如果您要创建新的自定义词汇表，我们强烈建议使用表格格式。

您可以使用AWS Management Console、AWS CLI或 AWS SDK 从列表中创建自定义词汇表。

- AWS Management Console：您必须创建并上传包含您的自定义词汇表的文本文件。您可以使用行分隔或逗号分隔的条目。请注意，您的列表必须保存为文本 (*.txt) 格式LF文件。如果您使用任何其他格式，例如CRLF，您的自定义词汇表不被接受Amazon Transcribe。
- AWS CLI和 AWSSDK：您必须使用标志在 API 调用中以逗号分隔的条目形式包含自定义词汇。`Phrases`

如果一个条目包含多个单词，则必须用连字符连接每个单词。例如，您将 “洛杉矶” 添加为，将 “安道尔城**`Los-Angeles`**” 添加为。**`Andorra-la-Vella`**

以下是两种有效列表格式的示例。有关特定方法创建自定义词汇表的示例，请参阅。

- 以逗号分隔的条目：

```
Los-Angeles,CLI,Eva-Maria,ABCs,Andorra-la-Vella
```

- 以行分隔的条目：

```
Los-Angeles
CLI
Eva-Maria
ABCs
Andorra-la-Vella
```

> **Important**
>
> 您只能使用您的语言支持的字符。有关详细信息，请参阅您的语言的字符集。

该`CreateMedicalVocabulary`操作不支持自定义词汇表。如果创建自定义医学词汇表，则必须使用表格格式；有关说明，使用表格创建自定义词汇表请参阅。

创建自定义词汇表

要处理自定义词汇表以供使用Amazon Transcribe，请参阅以下示例：

AWS CLI

此示例使用带有列表格式的自定义词汇文件的 creat e-worksward 命令。有关更多信息，请参阅`CreateVocabulary`：

```
aws transcribe create-vocabulary \
--vocabulary-name my-first-vocabulary \
--language-code en-US \
--phrases {CLI,Eva-Maria,ABCs}
```

这是使用 create-worksarch 命令的另一个示例，以及一个创建自定义词汇表的请求正文。

```
aws transcribe create-vocabulary \
--cli-input-json file://filepath/my-first-vocab-list.json
```

my-first-vocab-list.json 文件包含以下请求正文。

```
{
  "VocabularyName": "my-first-vocabulary",
  "LanguageCode": "en-US",
  "Phrases": [
        "CLI","Eva-Maria","ABCs"
  ]
}
```

从VocabularyState更改PENDING为后READY，您的自定义词汇便可与转录一起使用。要查看自定义词汇表的当前状态，请运行：

```
aws transcribe get-vocabulary \
--vocabulary-name my-first-vocabulary
```

AWS SDK for Python (Boto3)

此示例使用使用 create _workash 方法从列表中创建自定义词汇表。AWS SDK for Python (Boto3)有关更多信息，请参阅CreateVocabulary：

有关使用 AWS SDK 的其他示例，包括特定功能、场景和跨服务示例，请参阅本章。使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
vocab_name = "my-first-vocabulary"
```

```
response = transcribe.create_vocabulary(
    LanguageCode = 'en-US',
    VocabularyName = vocab_name,
    Phrases = [
        'CLI','Eva-Maria','ABCs'
    ]
)

while True:
    status = transcribe.get_vocabulary(VocabularyName = vocab_name)
    if status['VocabularyState'] in ['READY', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

> **Note**
>
> 如果您为自定义词汇文件创建新的Amazon S3存储桶，请确保CreateVocabulary提出请求的IAM角色有权访问此存储桶。如果该角色没有正确的权限，则您的请求会失败。您可以选择通过添加DataAccessRoleArn参数在请求中指定IAM角色。有关中的IAM角色和策略的更多信息Amazon Transcribe，请参阅Amazon Transcribe 基于身份的策略示例。

使用自定义词汇

创建自定义词汇表后，您可以将其包含在转录请求中；有关示例，请参阅以下部分。

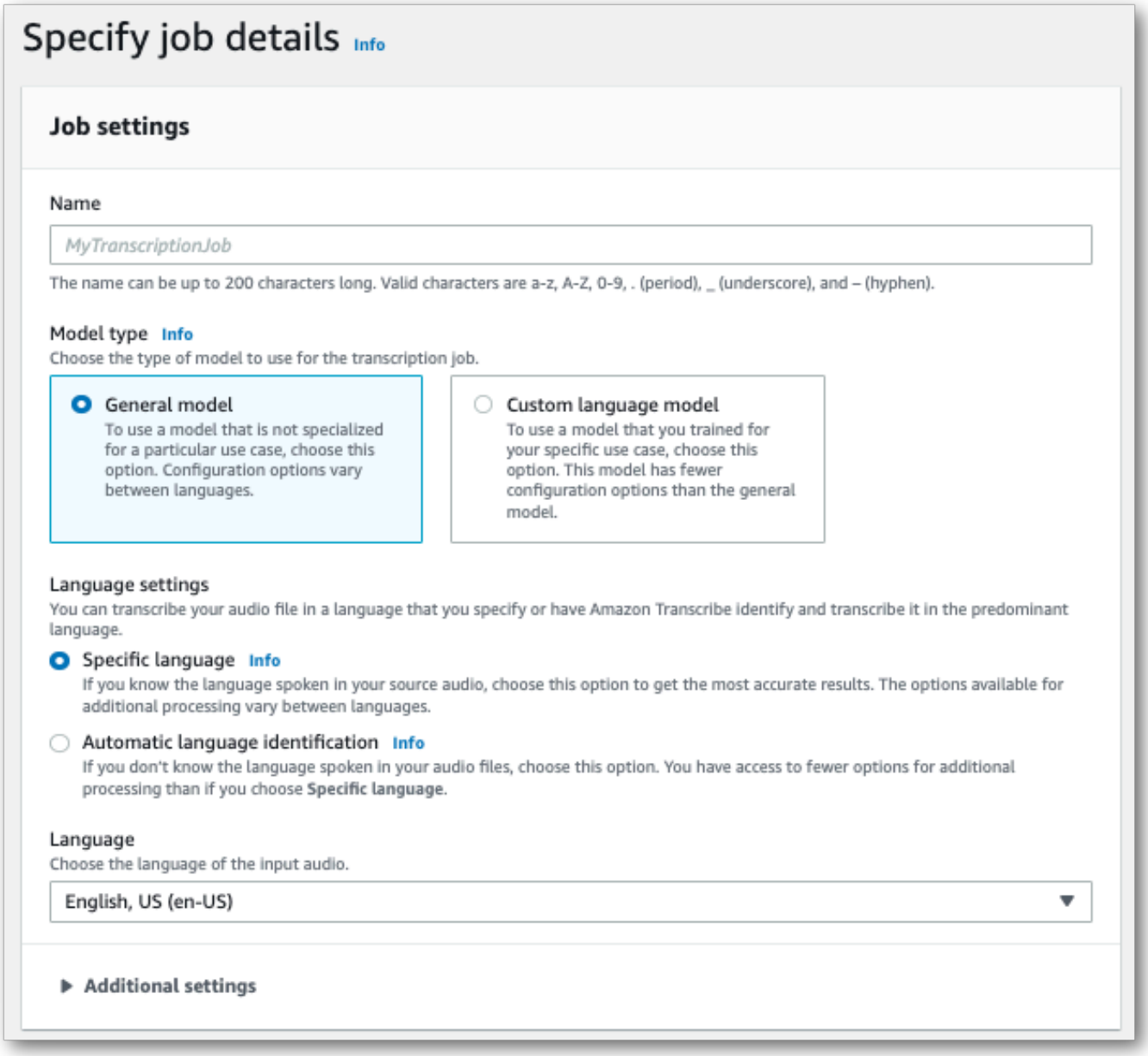
您在请求中包含的自定义词汇的语言必须与您为媒体指定的语言代码相匹配。如果语言不匹配，则您的自定义词汇不会应用于您的转录，也不会出现警告或错误。

在批量转录中使用自定义词汇

要在批量转录中使用自定义词汇，请参阅以下示例：

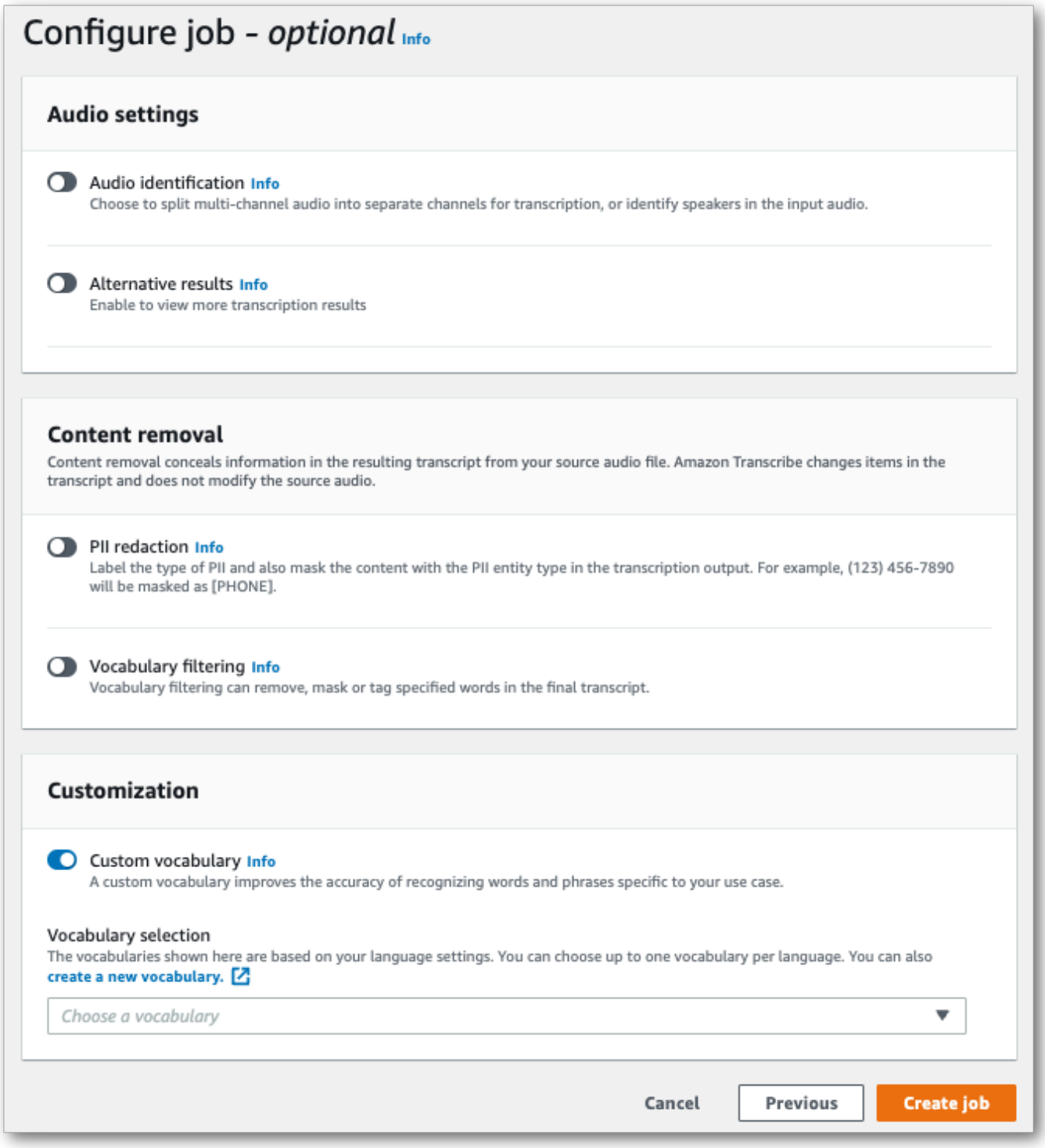
AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。



命名您的任务并指定您的输入媒体。（可选）包括任何其他字段，然后选择 “下一步”。

3. 在 “配置作业” 页面底部的 “自定义” 面板中，打开 “自定义词汇表”。



4. 从下拉菜单中选择您的自定义词汇。

   选择 “创建作业” 以运行转录作业。

AWS CLI

此示例使用带有VocabularyName子Settings参数的start-transcription-job命令和参数。有关更多信息，请参阅 StartTranscriptionJob 和 Settings。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
--settings VocabularyName=my-first-vocabulary
```

这是另一个使用start-transcription-job命令的示例，以及一个包含该作业的自定义词汇表的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://my-first-vocabulary-job.json
```

my-first-vocabulary-job.json 文件包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "Settings": {
        "VocabularyName": "my-first-vocabulary"
   }
}
```

AWS SDK for Python (Boto3)

此示例使用 start_transcription_job 方法的Settings参数来包含自定义词汇表。AWS SDK for Python (Boto3)有关更多信息，请参阅 StartTranscriptionJob 和 Settings。

有关使用AWS软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS章。

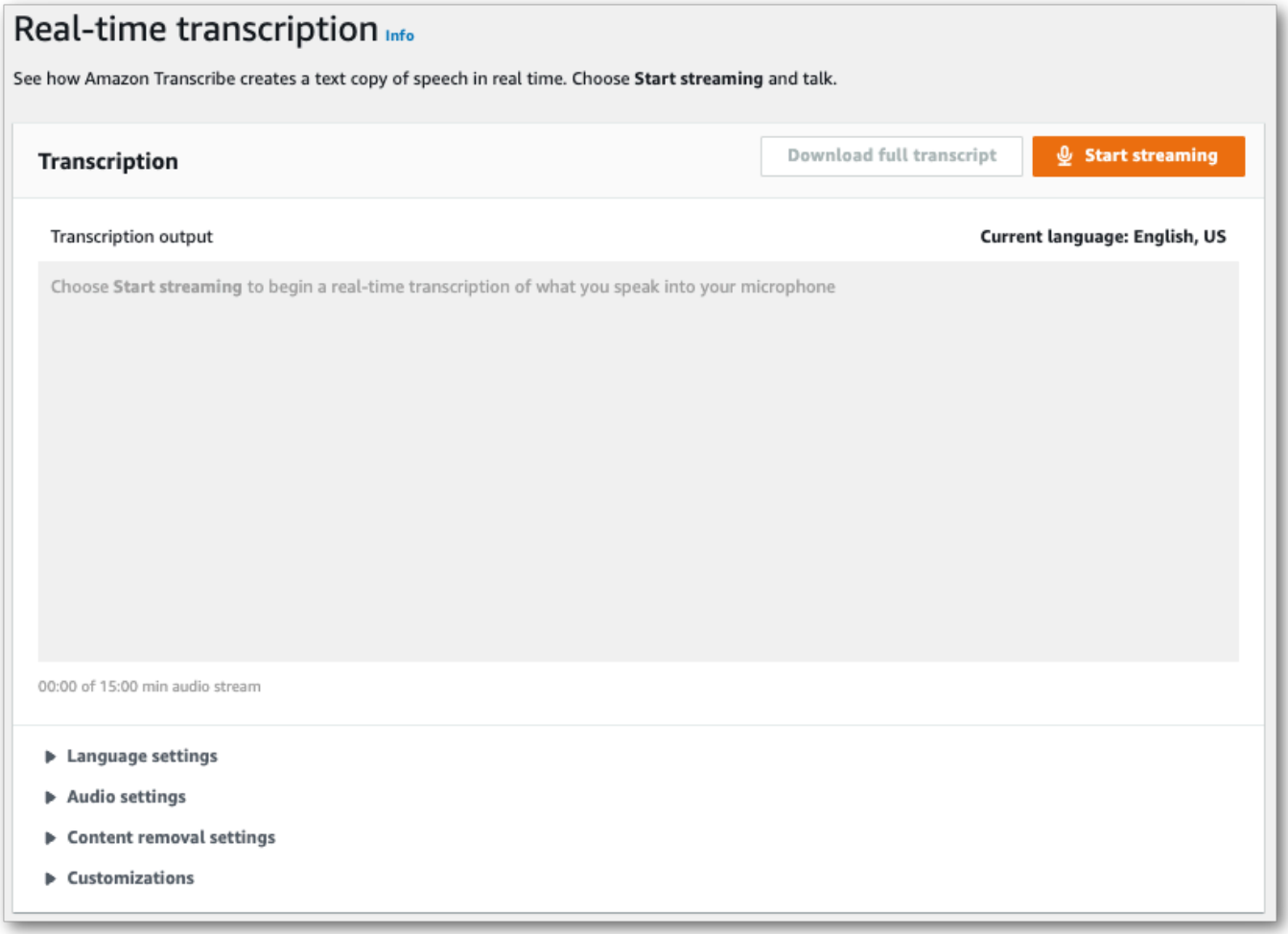
```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Settings = {
        'VocabularyName': 'my-first-vocabulary'
   }
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

在流媒体转录中使用自定义词汇

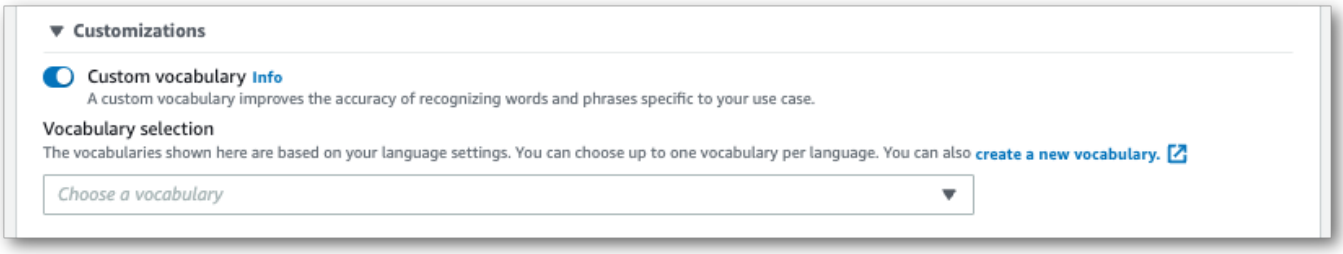
要在流媒体转录中使用自定义词汇，请参阅以下示例：

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择 Real-time transcription (实时转录)。向下滚动到 “自定义”，如果该字段最小化，则将其展开。



3. 打开“自定义词汇”，然后从下拉菜单中选择一个自定义词汇。



包括您希望应用于串流的设置。

4. 您现在已准备好。选择“开始直播”并开始讲话。要结束听写，请选择“停止直播”。

HTTP/2 串流转至

此示例创建了一个包含您的自定义词汇表的 HTTP/2 请求。有关使用 HTTP/2 流式传输的更多信息Amazon Transcribe，请参阅设置 HTTP/2 音频流。有关特定参数和标题的更多详细信息Amazon Transcribe，请参阅StartStreamTranscription。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: 20220208T235959Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key/20220208/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-transcribe-vocabulary-name: my-first-vocabulary
transfer-encoding: chunked
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

WebSocket 流

此示例创建了一个预签名 URL，用于将您的自定义词汇表应用于 WebSocket直播。为了便于阅读，已增加了换行符。有关将 WebSocket直播与一起使用的更多信息Amazon Transcribe，请参阅设置直 WebSocket 播。有关参数的更多详细信息，请参阅StartStreamTranscription。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
```

```
&vocabulary-name=my-first-vocabulary
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

自定义语言模型

自定义语言模型旨在提高特定领域语音的转录准确性。这包括您在正常的日常对话中听到的内容之外的任何内容。例如，如果您要转录科学会议的会议记录，则标准转录不太可能识别主持人使用的许多科学术语。在这种情况下，您可以训练自定义语言模型以识别您的学科中使用的专业术语。

与自定义词汇不同，自定义词汇通过提供提示（例如发音）来提高单词的识别能力，而自定义语言模型则学习与给定单词相关的上下文。这包括单词的使用方式和时间，以及单词与其他单词的关系。例如，如果你使用气候科学研究论文训练模型，你的模型可能会得知 “浮冰” 比 “冰流” 更有可能成为词对。

要查看自定义语言模型支持的语言，请参阅支持的语言和特定语言的特征。请注意，如果您在请求中包含自定义语言模型，则无法启用语言识别（必须指定语言代码）。

> 特定于自定义语言模型的 API 操作
>
> CreateLanguageModel, DeleteLanguageModel, DescribeLanguageModel, ListLanguageModels

数据源

您可以使用任何类型的文本数据来训练模型。但是，您的文本内容与音频内容越接近，您的模型就越准确。因此，在与音频相同的上下文中选择使用相同术语的文本数据非常重要。

训练模型的最佳数据是准确的记录文本。这被视为域内数据。域内文本数据的术语、用法和上下文与您要转录的音频完全相同。

如果您没有准确的笔录，请使用期刊文章、技术报告、白皮书、会议记录、指导手册、新闻文章、网站内容以及任何其他包含与音频环境相似的所需术语的文本。这被视为与域名相关的数据。

创建强大的自定义语言模型可能需要大量的文本数据，其中必须包含音频中使用的术语。您可以Amazon Transcribe提供多达 2 GB 的文本数据来训练模型，这称为训练数据。或者，如果您没有（或很少）域内转录，则可以Amazon Transcribe提供多达 200 MB 的文本数据来调整模型，这称为调整数据。

训练与调整数据

训练数据的Amazon Transcribe目的是教导人们识别新术语并了解这些术语的使用上下文。为了创建可靠的模型，Amazon Transcribe可能需要大量的相关文本数据。强烈建议提供尽可能多的训练数据，不超过 2 GB 的限制。

调整数据的目的是帮助完善和优化从训练数据中学到的情境关系。创建自定义语言模型不需要调整数据。

由您决定如何最好地选择训练和调整数据（可选）。每种情况都是独一无二的，取决于您拥有的数据类型和数量。当您缺少域内训练数据时，建议使用调整数据。

如果您选择同时包含这两种数据类型，请勿重叠训练和调整数据；训练和调整数据应是唯一的。重叠数据可能会使您的自定义语言模型产生偏差和偏差，从而影响其准确性。

作为一般指导，我们建议尽可能使用准确的域内文本作为训练数据。以下是一些一般场景，按优先顺序列出：

- 如果您有超过 10,000 字的准确域内转录文本，请将其用作训练数据。在这种情况下，无需包含调整数据。这是训练自定义语言模型的理想方案。
- 如果您有准确的域内转录文本，包含的单词少于 10,000 个单词且未获得预期的结果，请考虑使用与领域相关的书面文本（例如技术报告）来补充您的训练数据。在这种情况下，保留一小部分（10-25%）的域内转录数据用作调整数据。
- 如果您没有域内转录文本，请上传所有与域名相关的文本作为训练数据。在这种情况下，转录样式的文本比书面文本更可取。这是训练自定义语言模型的最不有效的方案。

准备好创建模型时，请参见创建自定义语言模型。

创建自定义语言模型

在创建自定义语言模型之前，您必须：

- 准备数据。数据必须以纯文本格式保存，不能包含任何特殊字符。
- 将您的数据上传到Amazon S3存储桶中。建议为训练和调整数据创建单独的文件夹。
- 确保Amazon Transcribe有权访问您的Amazon S3存储桶。您必须指定具有访问权限的IAM角色才能使用您的数据。

准备数据

您可以将所有数据编译到一个文件中，也可以将其保存为多个文件。请注意，如果您选择包含调整数据，则必须将其与训练数据保存在单独的文件中。

您使用多少文本文件作为训练或调整数据都无关紧要。上传一个包含 100,000 个字的文件与上传 10 个包含 10,000 个字的文件产生的结果相同。以最方便的方式准备文本数据。

确保您的所有数据文件符合以下标准：

- 它们都与你要创建的模型使用相同的语言。例如，如果您想创建一个用美国英语 (en-US) 转录音频的自定义语言模型，则所有文本数据都必须是美国英语。
- 它们采用纯文本格式，采用 UTF-8 编码。
- 它们不包含任何特殊字符或格式，例如 HTML 标签。
- 它们相当于训练数据的最大总大小为 2 GB，调整数据的最大总大小为 200 MB。

如果未满足其中任何条件，您的模型将失败。

上传您的数据

在上传数据之前，请为训练数据创建一个新文件夹。如果使用调整数据，请创建另一个单独的文件夹。

您的存储桶的 URI 可能如下所示：

- `s3://DOC-EXAMPLE-BUCKET/my-model-training-data/`
- `s3://DOC-EXAMPLE-BUCKET/my-model-tuning-data/`

将您的训练和调整数据上传到相应的存储分区中。

您可以稍后向这些存储桶添加更多数据。但是，如果您这样做，则需要使用新数据重新创建模型。无法使用新数据更新现有模型。

允许访问您的数据

要创建自定义语言模型，必须指定有权访问您的Amazon S3存储桶的IAM角色。如果您还没有有权访问放置训练数据的Amazon S3存储桶的角色，则必须创建一个。您创建角色后，您可以附加策略以授予该角色权限的策略来授予该角色权限。不要将策略附加到用户。

有关示例策略，请参阅 Amazon Transcribe 基于身份的策略示例。

要了解如何创建新IAM身份，请参阅IAM身份（用户、用户组和角色）。

要了解有关策略的更多信息，请参阅：

- 中的策略和权限 IAM
- 创建IAM策略
- 适用于 AWS 资源的访问管理

创建您的自定义语言模型

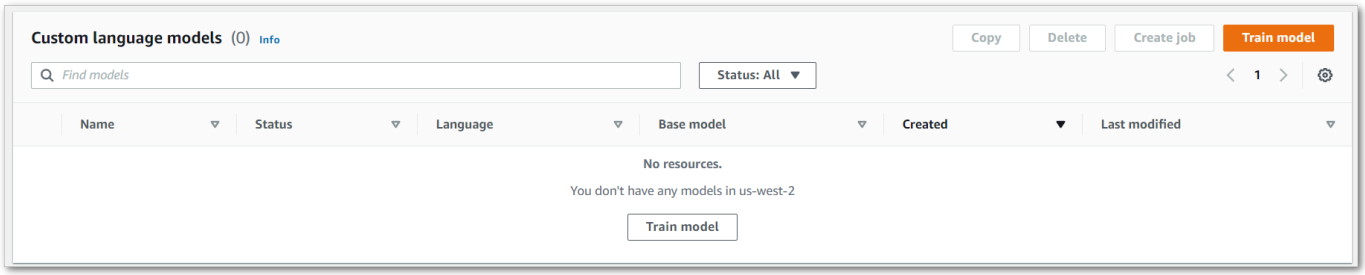
创建自定义语言模型时，必须选择基础模型。有两个基本模型选项：

- `NarrowBand`：将此选项用于采样率低于 16,000 Hz 的音频。此模型类型通常用于以 8,000 Hz 录制的电话通话。
- `WideBand`：将此选项用于采样率大于或等于 16,000 Hz 的音频。

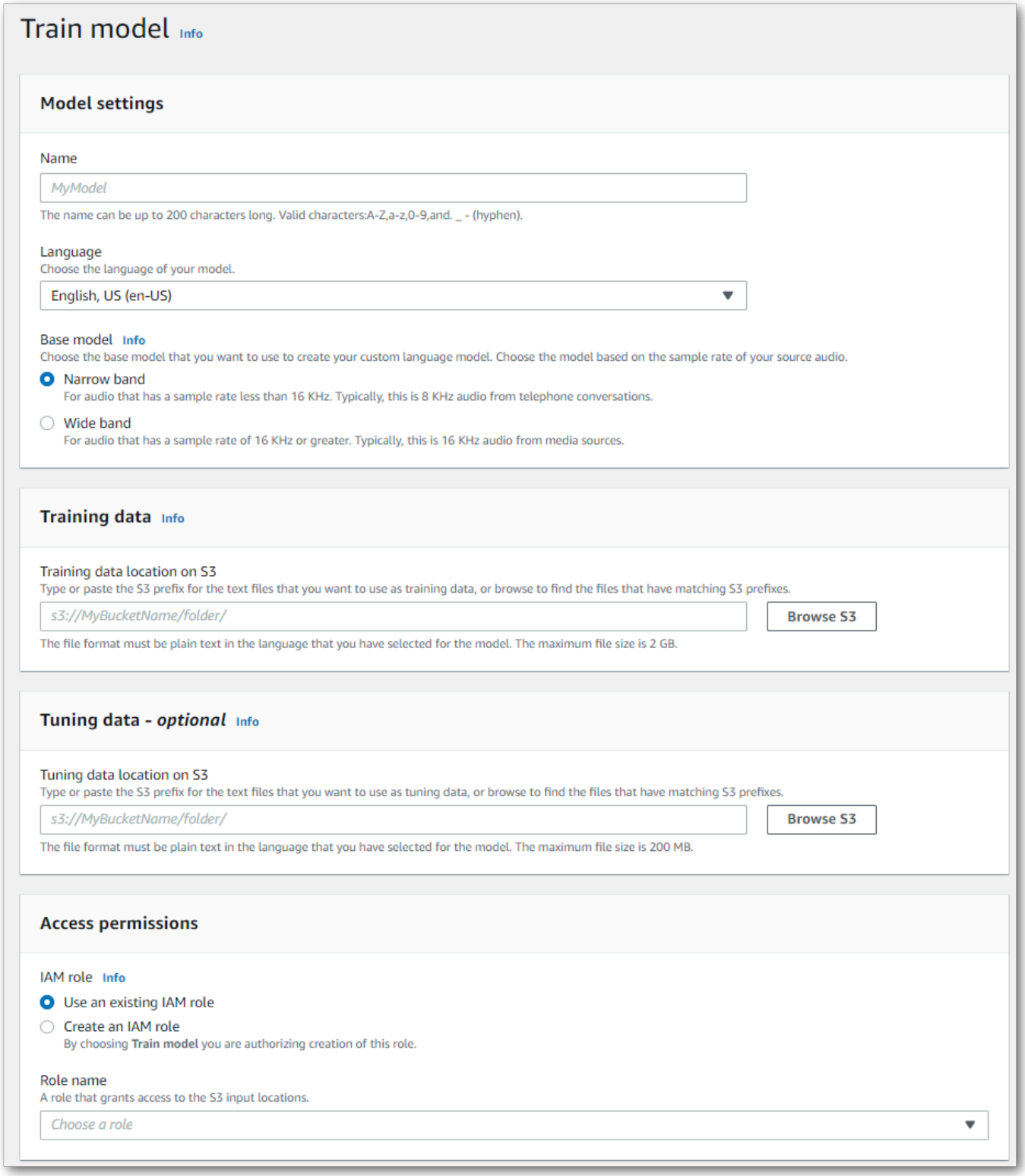
您可以使用AWS Management Console、AWS CLI或 AWS SDK 创建自定义语言模型。参见以下示例：

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择 Cstom 语言模型）。这将打开自定义语言模型页面，您可以在其中查看现有的自定义语言模型或训练新的自定义语言模型。
3. 要训练新模型，请选择训练模型。



这将带您进入火车模型页面。添加名称，指定语言，然后为模型选择所需的基本模型。然后，添加训练路径，也可以添加调整数据。您必须包含有权访问您的数据的IAM角色。



4. 完成所有字段后，选择页面底部的 T rain model。

AWS CLI

此示例使用create-language-model命令。有关更多信息，请参阅 CreateLanguageModel 和 LanguageModel。

```
aws transcribe create-language-model \
--base-model-name NarrowBand \
--model-name my-first-language-model \
--input-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET/my-clm-training-
data/,TuningDataS3Uri=s3://DOC-EXAMPLE-BUCKET/my-clm-tuning-
data/,DataAccessRoleArn=arn:aws:iam::111122223333:role/ExampleRole \
--language-code en-US
```

这是使用create-language-model命令的另一个示例，以及创建自定义语言模型的请求正文。

```
aws transcribe create-language-model \
--cli-input-json file://filepath/my-first-language-model.json
```

my-first-language-model.json 文件包含以下请求正文。

```
{
  "BaseModelName": "NarrowBand",
  "ModelName": "my-first-language-model",
  "InputDataConfig": {
         "S3Uri": "s3://DOC-EXAMPLE-BUCKET/my-clm-training-data/",
         "TuningDataS3Uri"="s3://DOC-EXAMPLE-BUCKET/my-clm-tuning-data/",
         "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole"
    },
  "LanguageCode": "en-US"
}
```

AWS SDK for Python (Boto3)

此示例使用使用 create_l AWS SDK for Python (Boto3) anguag e_model 方法创建 CLM。有关更多信息，请参阅 CreateLanguageModel 和 LanguageModel。

有关使用 AWS SDK 的其他示例，包括特定功能、场景和跨服务示例，请参阅本章。使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS

```
from __future__ import print_function
import time
```

```
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
model_name = 'my-first-language-model',
transcribe.create_language_model(
    LanguageCode = 'en-US',
    BaseModelName = 'NarrowBand',
    ModelName = model_name,
    InputDataConfig = {
        'S3Uri':'s3://DOC-EXAMPLE-BUCKET/my-clm-training-data/',
        'TuningDataS3Uri':'s3://DOC-EXAMPLE-BUCKET/my-clm-tuning-data/',
        'DataAccessRoleArn':'arn:aws:iam::111122223333:role/ExampleRole'
    }
)

while True:
    status = transcribe.get_language_model(ModelName = model_name)
    if status['LanguageModel']['ModelStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

更新您的自定义语言模型

Amazon Transcribe不断更新可用于自定义语言模型的基本模型。为了从这些更新中受益，我们建议每 6 到 12 个月训练一次新的自定义语言模型。

要查看您的自定义语言模型是否使用最新的基础模型，DescribeLanguageModel请使用AWS CLI或 S AWS DK 运行请求，然后在响应中找到相应UpgradeAvailability字段。

如果UpgradeAvailability是true，则您的模型未运行基本模型的最新版本。要在自定义语言模型中使用最新的基础模型，必须创建新的自定义语言模型。自定义语言模型无法升级。

使用自定义语言模型

创建自定义语言模型后，即可将其包含在转录请求中；有关示例，请参阅以下部分。

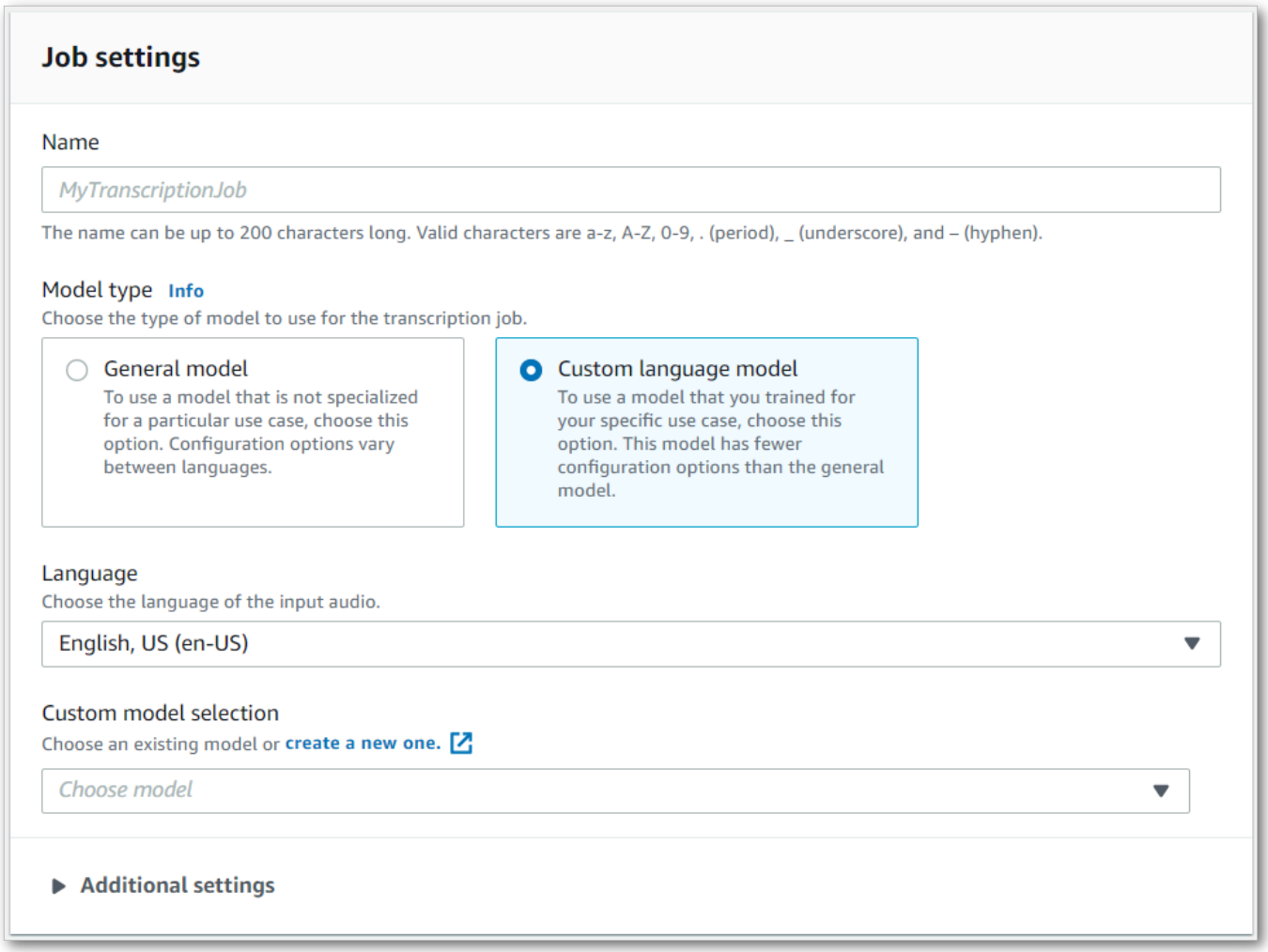
您在请求中包含的模型的语言必须与您为媒体指定的语言代码相匹配。如果语言不匹配，则您的自定义语言模型不会应用于您的转录，也不会出现警告或错误。

在批量转录中使用自定义语言模型

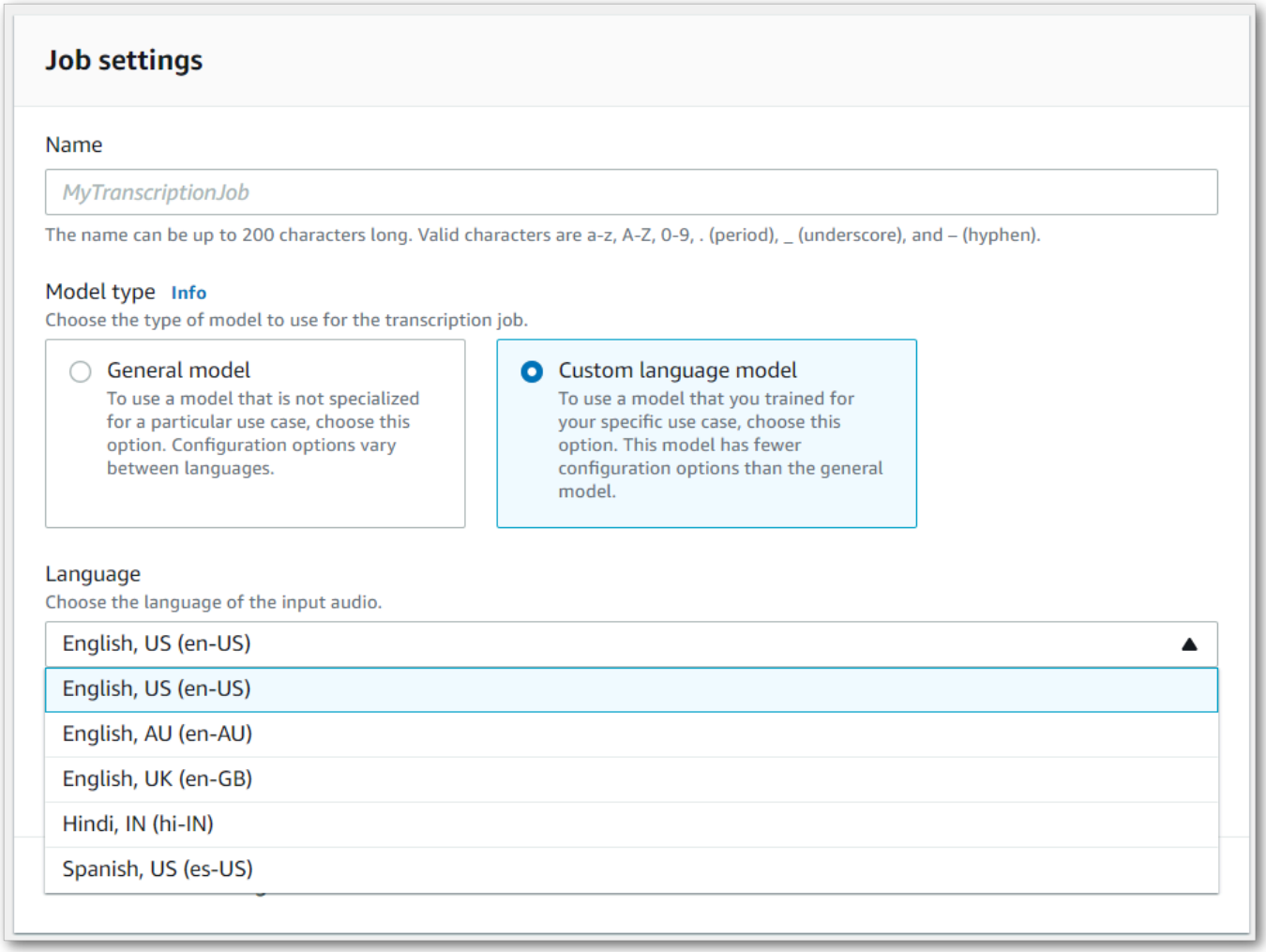
要使用带有批量转录的自定义语言模型，请参阅以下示例：

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。
3. 在 “模型类型” 下的 “Job 设置” 面板中，选择 “自定义语言模型” 框。



您还必须从下拉菜单中选择一种输入语言。



4. 在“自定义模型选择”下，从下拉菜单中选择现有的自定义语言模型或创建新模型。

   在输入数据面板中添加输入文件Amazon S3的位置。

5. 选择“下一步”以获取其他配置选项。

   选择“创建作业”以运行转录作业。

AWS CLI

此示例使用带有VocabularyName子ModelSettings参数的start-transcription-job命令和参数。有关更多信息，请参阅 StartTranscriptionJob 和 ModelSettings。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
```

```
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
--model-settings LanguageModelName=my-first-language-model
```

这是使用start-transcription-job命令的另一个示例，以及一个包含该任务的自定义语言模型的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://my-first-model-job.json
```

my-first-model-job.json 文件包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "ModelSettings": {
        "LanguageModelName": "my-first-language-model"
   }
}
```

AWS SDK for Python (Boto3)

此示例使用使用 start_transcription_job 方法的ModelSettings参数来包含自定义语言模型。AWS SDK for Python (Boto3)有关更多信息，请参阅 StartTranscriptionJob 和 ModelSettings。

有关使用AWS软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS章。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
```

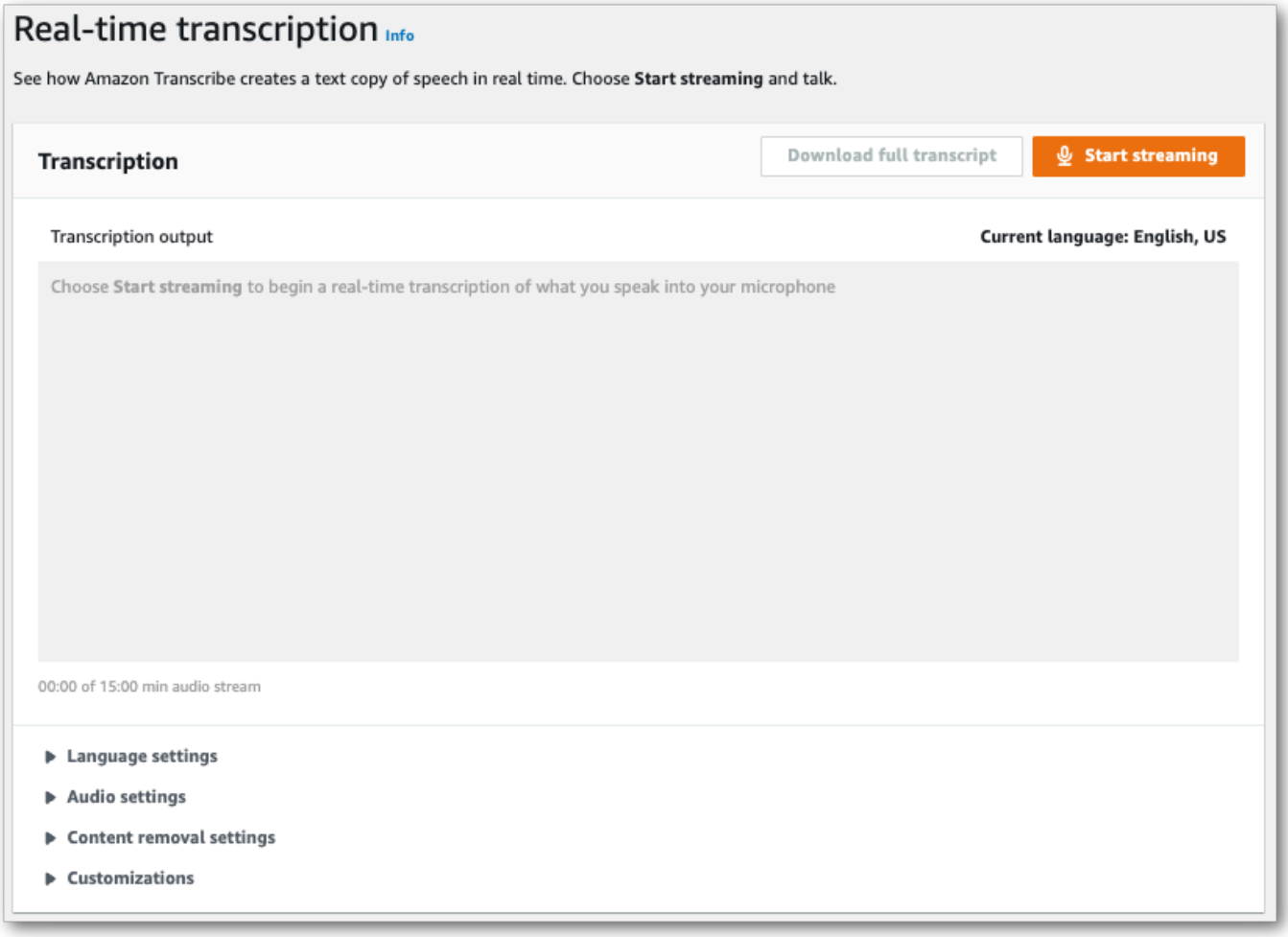
```
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    ModelSettings = {
        'LanguageModelName': 'my-first-language-model'
   }
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

在流媒体转录中使用自定义语言模型

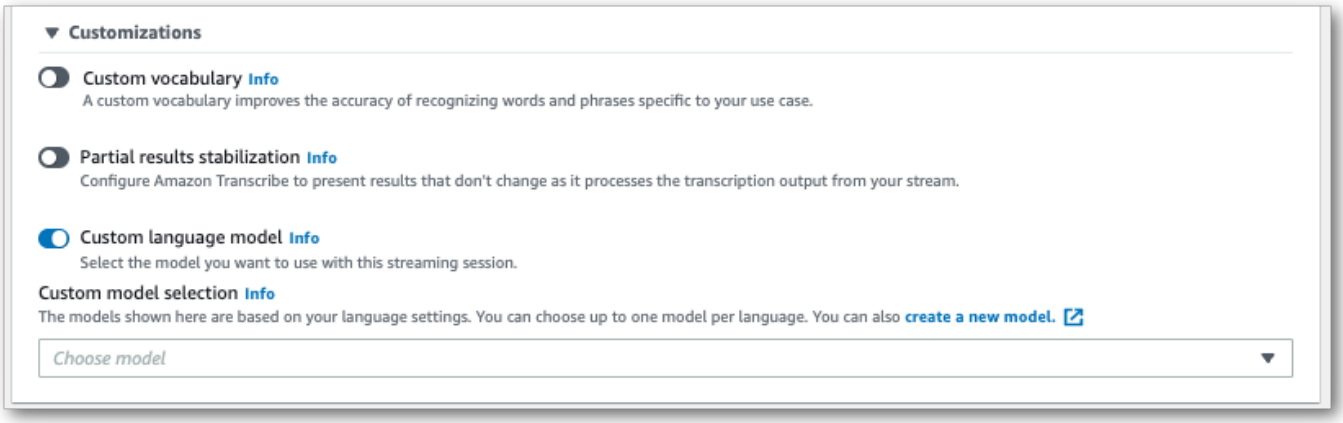
要使用自定义语言模型进行流式转录，请参阅以下示例：

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择 Real-time transcription (实时转录)。向下滚动到 “自定义”，如果该字段最小化，则将其展开。



3. 开启自定义语言模型，然后从下拉菜单中选择一个模型。



添加要应用于串流的所有其他设置。

4. 要已将要转录要要要要要要要使用要素的要素要要要要要 选择 “开始直播” 并开始讲话。要结束听写，请选择 “停止直播”。

HTTP/2 串流要

此示例创建了一个包含您的自定义语言模型的 HTTP/2 请求。有关使用 HTTP/2 流式传输的更多信息Amazon Transcribe，请参阅设置 HTTP/2 音频流。有关特定参数和标题的更多详细信息Amazon Transcribe，请参阅StartStreamTranscription。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: 20220208T235959Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key/20220208/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-transcribe-language-model-name: my-first-language-model
transfer-encoding: chunked
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

WebSocket 流

此示例创建了一个预签名 URL，该网址将您的自定义语言模型应用于 WebSocket 直播。为了便于阅读，已增加了换行符。有关将 WebSocket 直播与一起使用的更多信息Amazon Transcribe，请参阅设置直 WebSocket 播。有关参数的更多详细信息，请参阅StartStreamTranscription。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
```

```
&language-model-name=my-first-language-model
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

使用自定义词汇过滤器删除、屏蔽或举报单词

自定义词汇过滤器是一个文本文件，其中包含您要在转录输出中修改的单个单词的自定义列表。

一个常见的用例是删除攻击性或亵渎性术语；但是，自定义词汇过滤器是完全自定义的，因此你可以选择任何你想要的单词。例如，如果您即将推出新产品，则可以在会议记录中掩盖该产品名称。在这种情况下，您可以保密利益相关者， up-to-date 同时将产品名称保密，直到发布为止。

词汇筛选有三种显示方法：maskremove、和tag。请参阅以下示例，了解每个示例的工作原理。

- 掩码：用三个星号 (***) 替换指定单词。

```
"transcript": "You can specify a list of *** or *** words, and *** *** removes them
 from transcripts automatically."
```

- 删除：删除指定的单词，不留任何位置。

```
"transcript": "You can specify a list of or words, and removes them from transcripts
 automatically."
```

- 标签：为每个指定单词添加标签 ("vocabularyFilterMatch": true)，但不改变单词本身。标记允许快速替换和编辑脚本。

```
"transcript": "You can specify a list of profane or offensive words, and amazon
 transcribe removes them from transcripts automatically."
...
    "alternatives": [
        {
            "confidence": "1.0",
            "content": "profane"
        }
    ],
    "type": "pronunciation",
    "vocabularyFilterMatch": true
```

提交转录请求时，您可以指定自定义词汇过滤器和要应用的过滤方法。 Amazon Transcribe然后，根据您指定的过滤方法，修改文本中出现的确切单词匹配项。

自定义词汇过滤器可以应用于批量和流式转录请求。要了解如何创建自定义词汇序列，请参阅创建词汇过滤器。要了解如何应用自定义词汇过滤器，请参阅使用自定义词汇过滤器。

> **Note**
>
> Amazon Transcribe自动屏蔽种族敏感术语，但您可以通过联系AWS技术 Support 选择退出此默认过滤器。

有关词汇筛选的视频演练，请参阅使用词汇过滤器。

> **特定于词汇过滤的 API 操作**
>
> `CreateVocabularyFilter`, `DeleteVocabularyFilter`, `GetVocabularyFilter`, `ListVocabularyFilters`, `UpdateVocabularyFilter`

创建词汇过滤器

有两个选项可用于创建自定义词汇表：

1. 使用 UTF-8 编码将行分隔的单词列表保存为纯文本文件。
   - 您可以将此方法与AWS Management ConsoleAWS CLI、或AWS SDK 一起使用。
   - 如果使用AWS Management Console，则可以为自定义词汇文件提供本地路径或Amazon S3 URI。
   - 如果使用AWS CLI或AWS SDK，则必须将自定义词汇文件上传到Amazon S3存储桶并在请求中包含Amazon S3 URI。
2. 直接在您的 API 请求中包含以逗号分隔的单词列表。
   - 您可以使用此方法将此方法与使用`Words`参数的AWS CLI或AWS SDK 一起使用。

有关每种方法的示例，请参阅创建自定义词汇过滤器

创建自定义词汇过滤器时需要注意的事项：

- 单词不区分大小写。例如，“诅咒” 和 “诅咒” 的待遇相同。
- 仅筛选完全匹配的单词。例如，如果您的过滤器包含 “发誓”，但您的媒体包含 “发誓” 或 “发誓” 一词，则不会对其进行过滤。只有 “发誓” 的实例才会被过滤。因此，您必须包括要过滤的单词的所有变体。
- 过滤器不适用于换句话中包含的单词。例如，如果自定义词汇过滤器包含 “海军陆战队” 而不包含 “潜艇”，则笔录中的 “潜艇” 不会被更改。

- 每个条目只能包含一个单词（无空格）。
- 如果您将自定义词汇过滤器另存为文本文件，则它必须采用 UTF-8 编码的纯文本格式。
- 每个过滤器最多可以有 100 个自定义词汇过滤器AWS 账户，每个过滤器的大小可达 50 Kb。
- 您只能使用您的语言支持的字符。有关详细信息，请参阅您的语言的字符集。

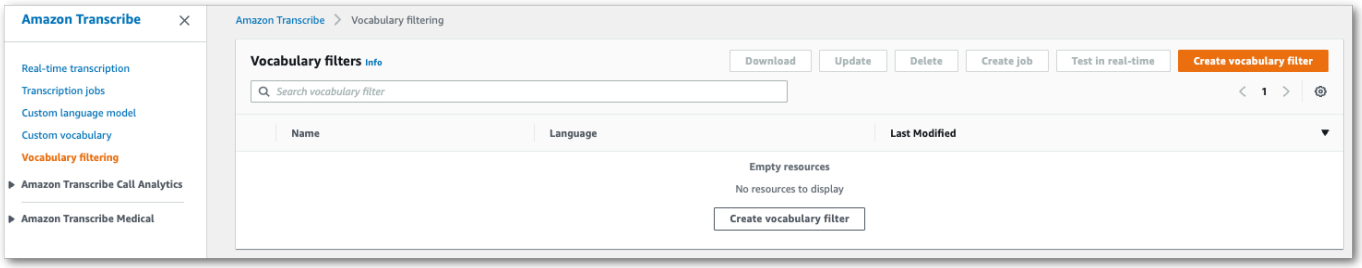
创建自定义词汇过滤器

要处理自定义词汇过滤器以供使用Amazon Transcribe，请参阅以下示例：

AWS Management Console

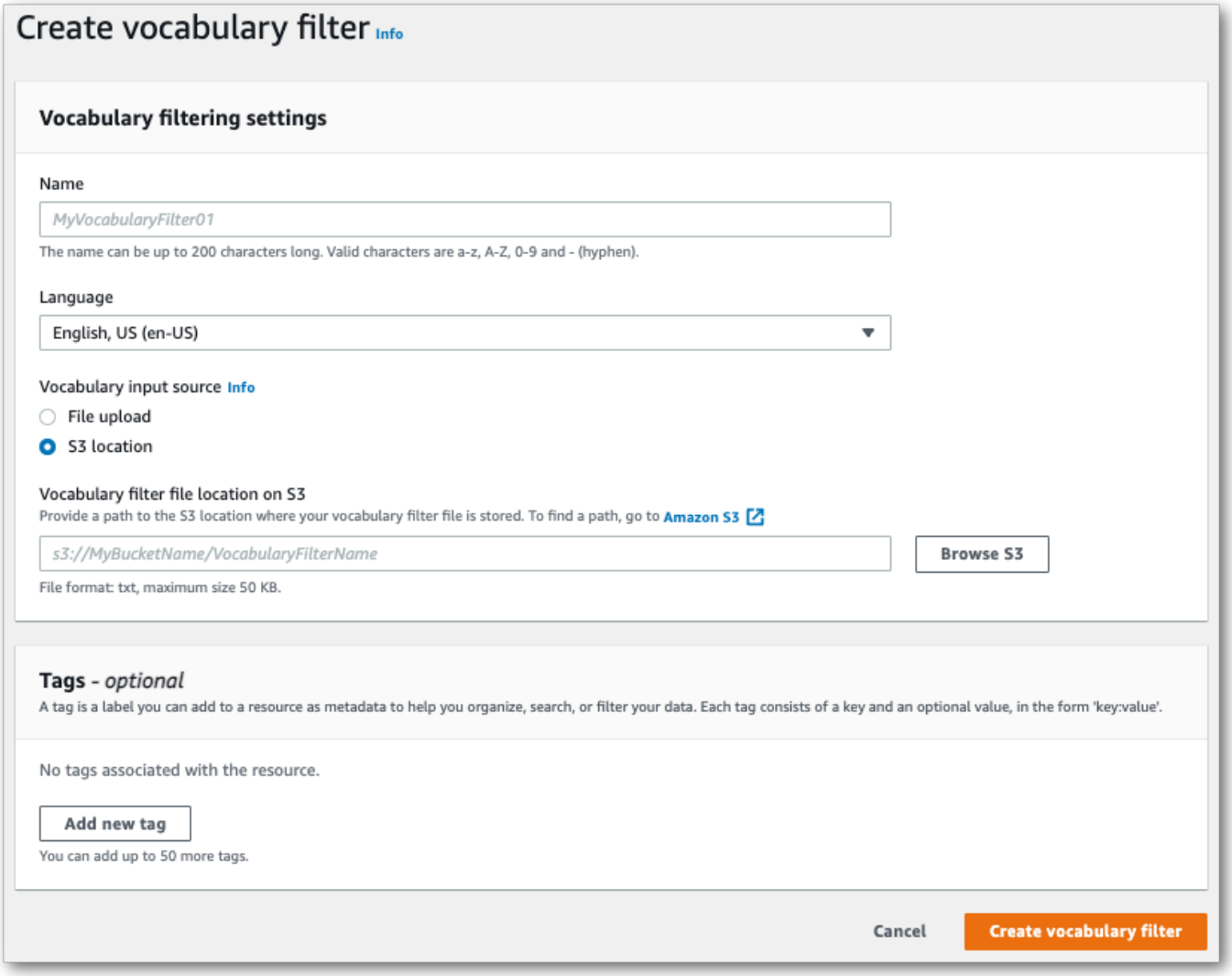
在继续之前，请将您的自定义词汇过滤器另存为文本 (*.txt) 文件。您可以选择将文件上传到Amazon S3存储桶。

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择 Vocabulary filtering (词汇筛选)。这将打开词汇过滤器页面，您可以在其中查看现有的自定义词汇过滤器或创建新的词汇过滤器。
3. 选择 “创建词汇过滤器”。



这将带您进入创建词汇过滤器页面。为新的自定义词汇过滤器输入名称。

在词汇输入源下选择文件上传或 S3 位置选项。然后指定您的自定义词汇文件的位置。



4. （可选）向您的自定义词汇过滤器添加标签。完成所有字段后，选择页面底部的创建词汇过滤器。如果在处理文件时没有错误，这会将你带回词汇过滤器页面。

   现在您的自定义词汇过滤器已准备好使用。

AWS CLI

此示例使用create-vocabulary-filter命令将单词列表处理成可用的自定义词汇过滤器。有关更多信息，请参阅CreateVocabularyFilter：

选项 1：您可以使用words参数将单词列表包含在请求中。

```
aws transcribe create-vocabulary-filter \
--vocabulary-filter-name my-first-vocabulary-filter \
```

```
--language-code en-US \
--words profane,offensive,Amazon,Transcribe
```

选项 2：您可以将单词列表保存为文本文件并将其上传到Amazon S3存储桶，然后使用vocabulary-filter-file-uri参数将文件的 URI 包含在您的请求中。

```
aws transcribe create-vocabulary-filter \
--vocabulary-filter-name my-first-vocabulary-filter \
--language-code en-US \
--vocabulary-filter-file-uri s3://DOC-EXAMPLE-BUCKET/my-vocabulary-filters/my-vocabulary-filter.txt
```

这是使用create-vocabulary-filter命令的另一个示例，以及一个创建自定义词汇过滤器的请求正文。

```
aws transcribe create-vocabulary-filter \
--cli-input-json file://filepath/my-first-vocab-filter.json
```

my-first-vocab-filter.json 文件包含以下请求正文。

选项 1：您可以使用Words参数将单词列表包含在请求中。

```
{
  "VocabularyFilterName": "my-first-vocabulary-filter",
  "LanguageCode": "en-US",
  "Words": [
        "profane","offensive","Amazon","Transcribe"
  ]
}
```

选项 2：您可以将单词列表保存为文本文件并将其上传到Amazon S3存储桶，然后使用VocabularyFilterFileUri参数将文件的 URI 包含在您的请求中。

```
{
  "VocabularyFilterName": "my-first-vocabulary-filter",
  "LanguageCode": "en-US",
  "VocabularyFilterFileUri": "s3://DOC-EXAMPLE-BUCKET/my-vocabulary-filters/my-vocabulary-filter.txt"
}
```

> **Note**
>
> 如果您在请求VocabularyFilterFileUri中包含，则无法使用Words；必须选择其中一个。

AWS SDK for Python (Boto3)

此示例使用 create_vocqual ary_filter 方法创建自定义词汇过滤器。AWS SDK for Python (Boto3)有关更多信息，请参阅CreateVocabularyFilter：

有关使用AWS软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS章。

选项 1：您可以使用Words参数将单词列表包含在请求中。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
vocab_name = "my-first-vocabulary-filter"
response = transcribe.create_vocabulary_filter(
    LanguageCode = 'en-US',
    VocabularyFilterName = vocab_name,
    Words = [
        'profane','offensive','Amazon','Transcribe'
    ]
)
```

选项 2：您可以将单词列表保存为文本文件并将其上传到Amazon S3存储桶，然后使用VocabularyFilterFileUri参数将文件的 URI 包含在您的请求中。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
vocab_name = "my-first-vocabulary-filter"
response = transcribe.create_vocabulary_filter(
    LanguageCode = 'en-US',
    VocabularyFilterName = vocab_name,
    VocabularyFilterFileUri = 's3://DOC-EXAMPLE-BUCKET/my-vocabulary-filters/my-vocabulary-filter.txt'
```

```
)
```

> **Note**
>
> 如果您在请求`VocabularyFilterFileUri`中包含，则无法使用`Words`；必须选择其中一个。

> **Note**
>
> 如果您为自定义词汇筛选文件创建新的Amazon S3存储桶，请确保`CreateVocabularyFilter`发出请求的IAM角色有权访问此存储桶。如果该角色没有正确的权限，则您的请求将失败。您可以选择通过添加`DataAccessRoleArn`参数在请求中指定IAM角色。有关中的IAM角色和策略的更多信息Amazon Transcribe，请参阅Amazon Transcribe 基于身份的策略示例。

使用自定义词汇过滤器

创建自定义词汇过滤器后，您可以将其包含在转录请求中；有关示例，请参阅以下部分。

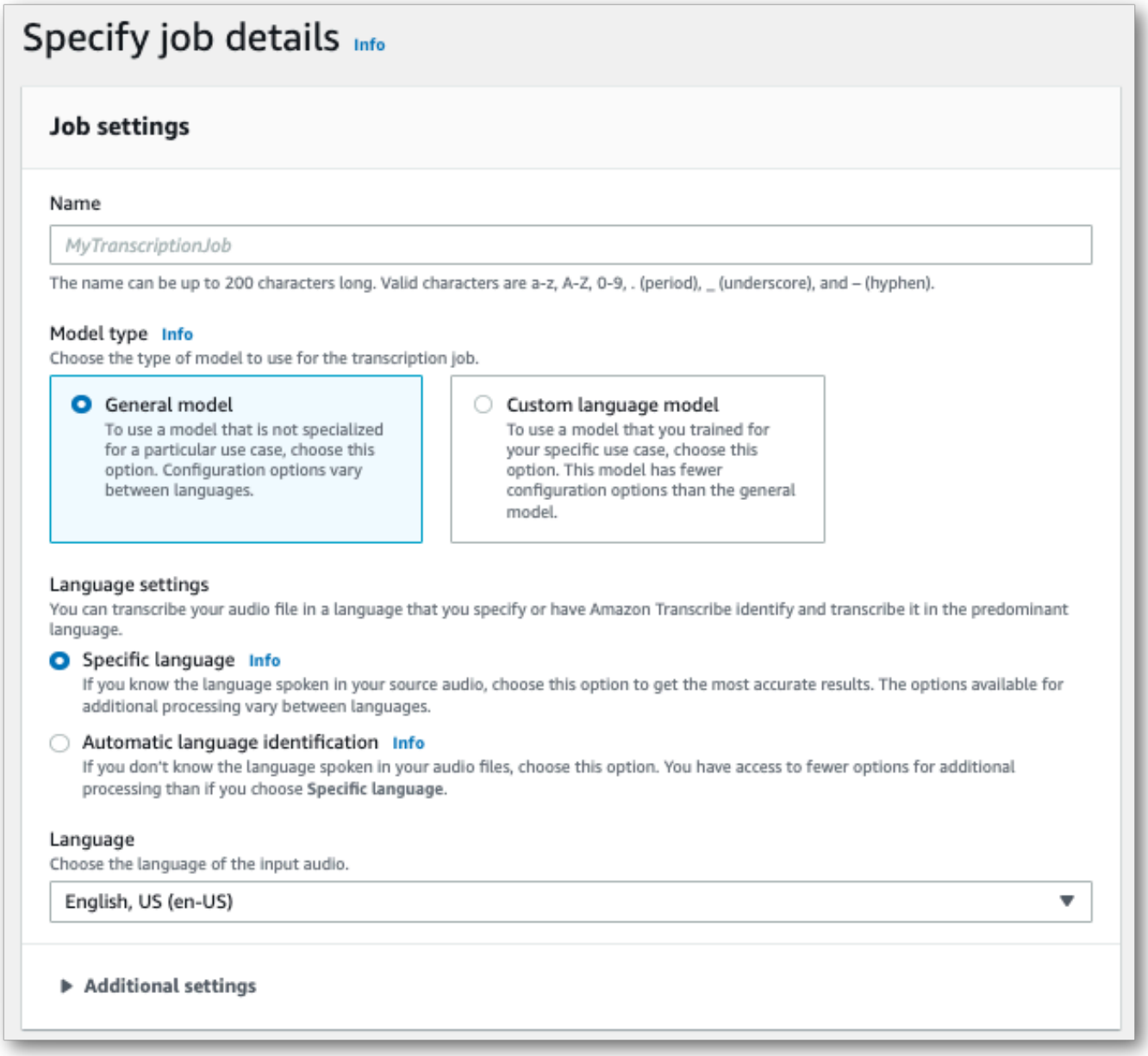
您在请求中包含的自定义词汇过滤器的语言必须与您为媒体指定的语言代码相匹配。如果您使用语言识别并指定多种语言选项，则可以为每种指定语言添加一个自定义词汇过滤器。如果您的自定义词汇过滤器的语言与音频中识别的语言不匹配，则您的过滤器不会应用于您的转录，也不会出现警告或错误。

在批量转录中使用自定义词汇过滤器

要使用带有批量转录的自定义词汇过滤器，请参阅以下示例：

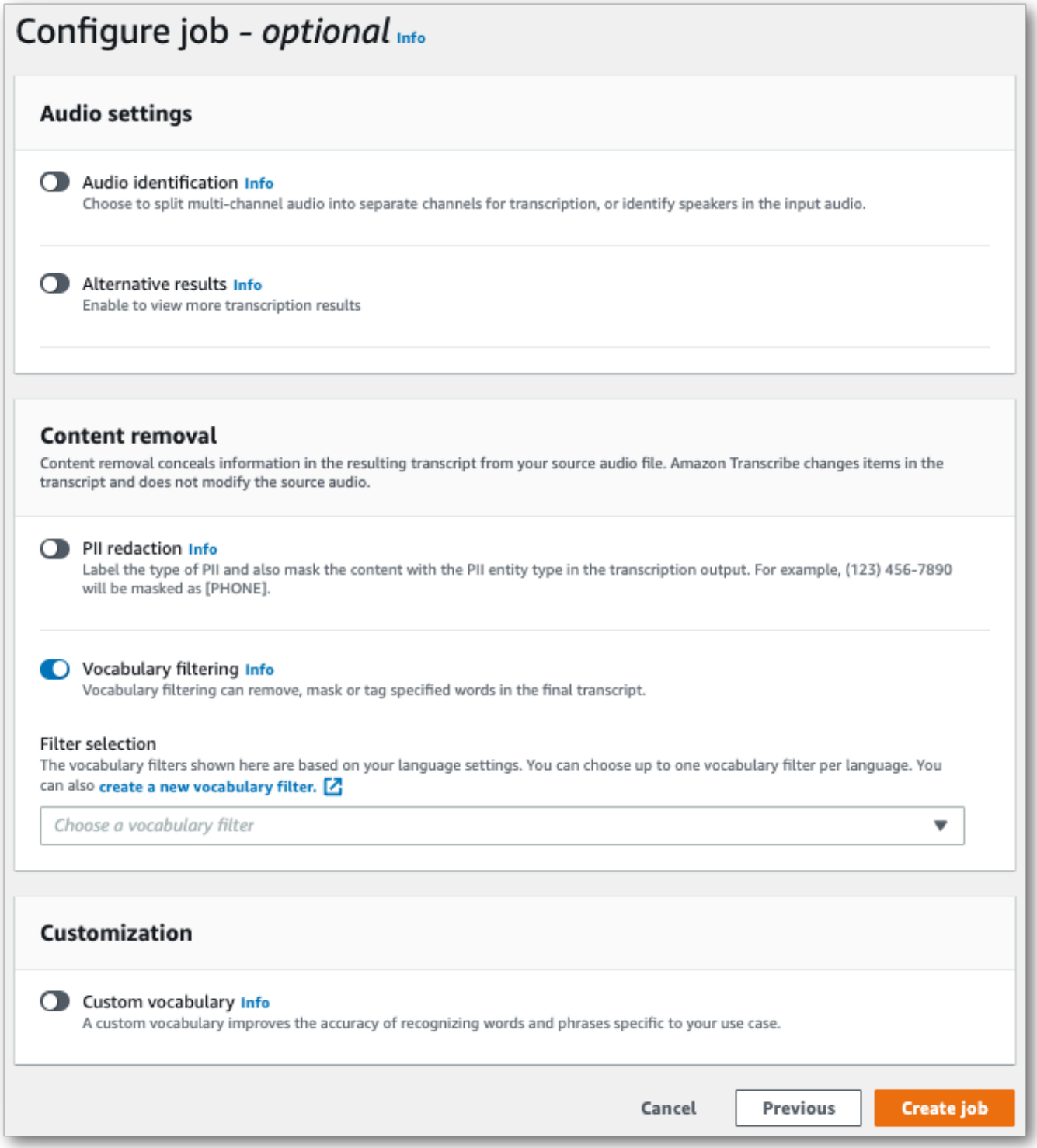
AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。

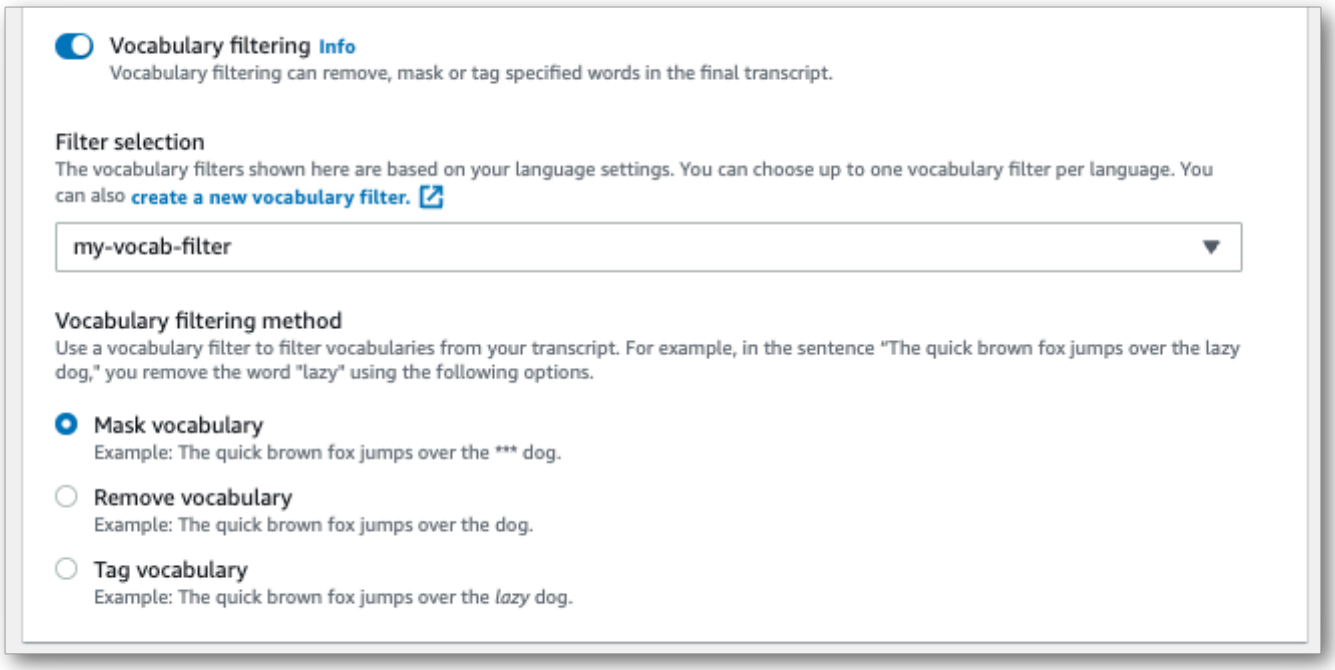


命名您的任务并指定您的输入媒体。（可选）包括任何其他字段，然后选择 “下一步”。

3. 在配置作业页面的内容删除面板中，开启词汇筛选。



4. 从下拉菜单中选择您的自定义词汇过滤器并指定过滤方法。



5. 选择 “创建作业” 以运行转录作业。

AWS CLI

此示例使用带有和VocabularyFilterMethod子Settings参数的start-transcription-job命令VocabularyFilterName和参数。有关更多信息，请参阅 StartTranscriptionJob 和 Settings。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
--settings VocabularyFilterName=my-first-vocabulary-filter,VocabularyFilterMethod=mask
```

这是使用该start-transcription-job命令的另一个示例，以及一个包含该作业的自定义词汇过滤器的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://my-first-vocabulary-filter-job.json
```

文件 my-first-vocabulary-filter-job.json 包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "Settings": {
        "VocabularyFilterName": "my-first-vocabulary-filter",
        "VocabularyFilterMethod": "mask"
   }
}
```

AWS SDK for Python (Boto3)

此示例使用使用 start_transcription_job 方法的Settings参数来添加自定义词汇过滤器。AWS SDK for Python (Boto3)有关更多信息，请参阅 StartTranscriptionJob 和 Settings。

有关使用AWS软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS章。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Settings = {
        'VocabularyFilterName': 'my-first-vocabulary-filter',
        'VocabularyFilterMethod': 'mask'
   }
)
```

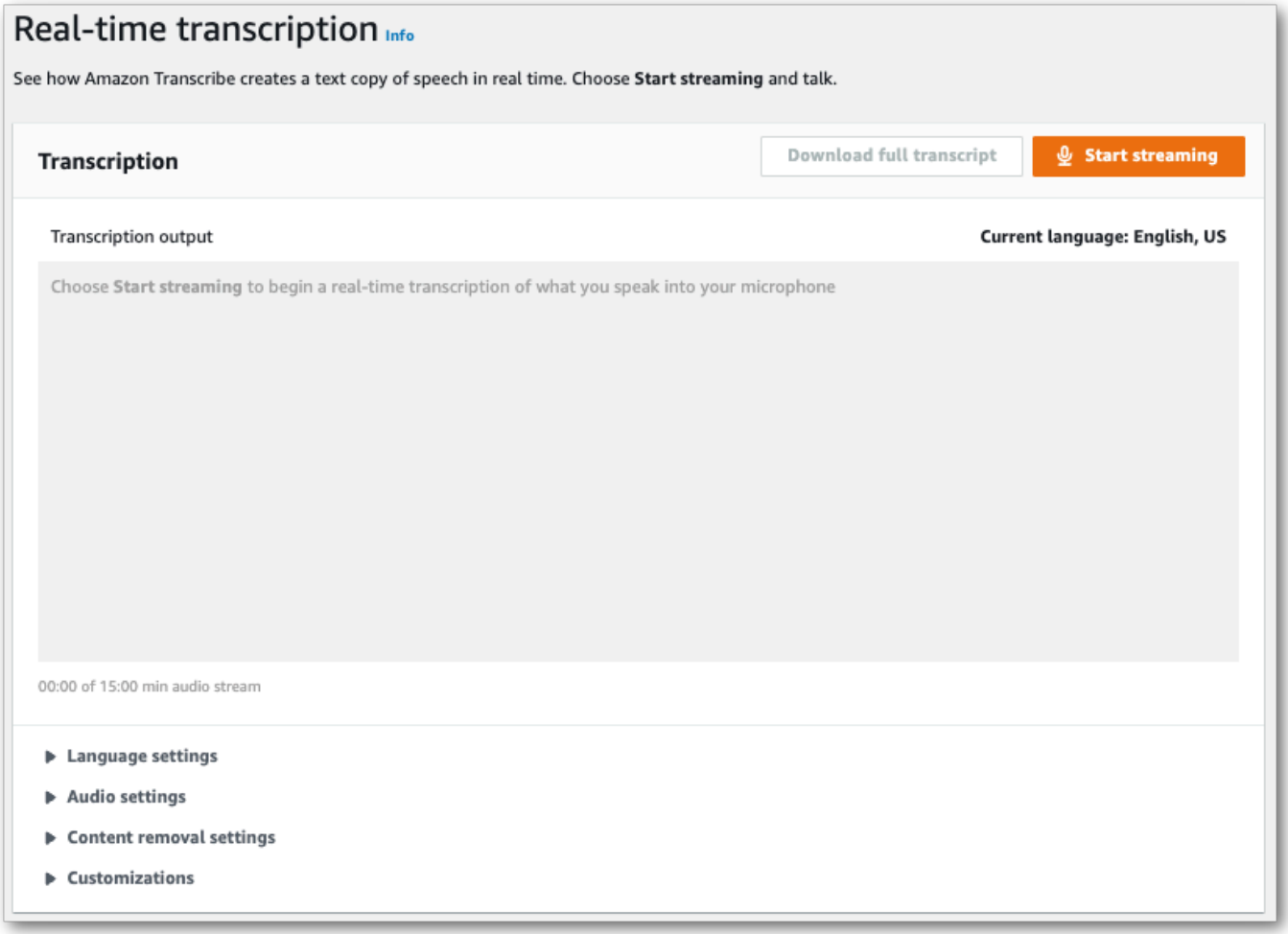
```
while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

在直播转录中使用自定义词汇过滤器

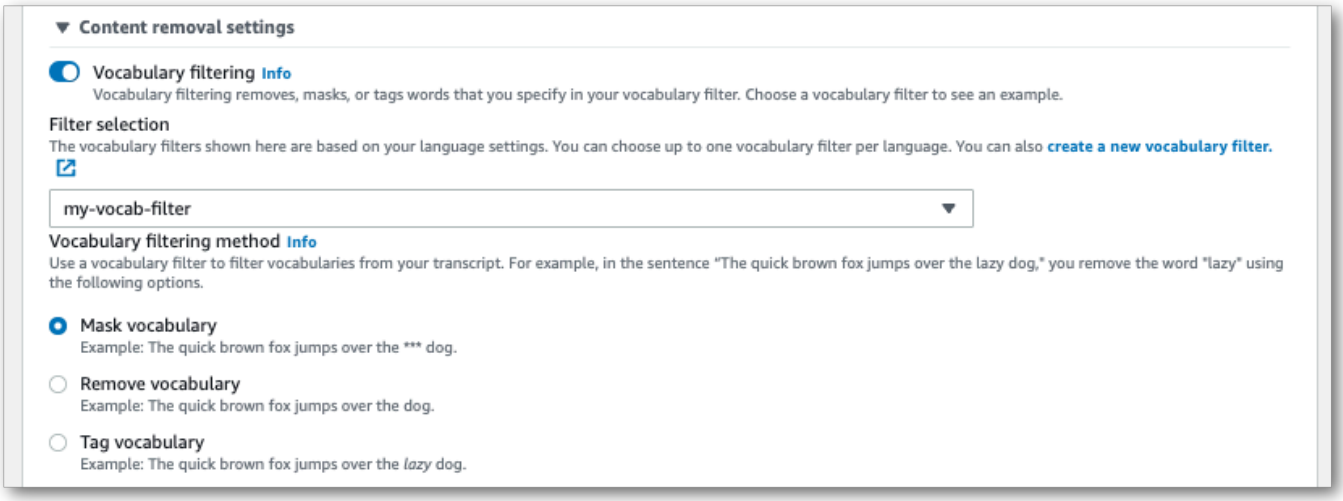
要在流式转录中使用自定义词汇过滤器，请参阅以下示例：

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择 Real-time transcription (实时转录)。向下滚动到内容删除设置，如果该字段已最小化，请将其展开。



3. 开启词汇筛选。从下拉菜单中选择一个自定义词汇过滤器，然后指定过滤方法。



包括您希望应用于直播的所有其他设置。

4. 您现在已准备好，可以转录您的直播。选择 “开始直播” 并开始讲话。要结束听写，请选择 “停止直播”。

HTTP/2 直播

此示例创建了一个 HTTP/2 请求，其中包括您的自定义词汇过滤器和筛选方法。有关使用 HTTP/2 流式传输的更多信息Amazon Transcribe，请参阅设置 HTTP/2 音频流。有关特定参数和标题的更多详细信息Amazon Transcribe，请参阅StartStreamTranscription。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: 20220208T235959Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key/20220208/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-transcribe-vocabulary-filter-name: my-first-vocabulary-filter
x-amzn-transcribe-vocabulary-filter-method: mask
transfer-encoding: chunked
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

WebSocket 流

此示例创建了一个预签名 URL，将您的自定义词汇过滤器应用于 WebSocket 直播。为了便于阅读，已增加了换行符。有关使用 WebSocket 直播的更多信息Amazon Transcribe，请参阅设置直 WebSocket 播。有关参数的更多详细信息，请参阅StartStreamTranscription。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
```

```
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
&vocabulary-filter-name=my-first-vocabulary-filter
&vocabulary-filter-method=mask
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

检测有毒言语

有毒语音检测旨在帮助审核涉及以下内容的社交媒体平台peer-to-peer对话，例如在线游戏和社交聊天平台。使用有害言论可能对个人、同龄群体和社区造成严重伤害。举报有害语言有助于组织保持对话的文明性，维护一个安全和包容的在线环境，供用户自由创建、分享和参与。

Amazon Transcribe毒性检测利用基于音频和文本的线索来识别和分类基于语音的有毒内容，分为七个类别，包括性骚扰、仇恨言论、威胁、虐待、亵渎、侮辱和图片。除了文字，Amazon Transcribe毒性检测使用语音线索（例如语气和音调）来深入了解语音中的毒性意图。这与标准内容审核系统相比有所改进，标准内容审核系统旨在仅关注特定条款，不考虑意图。

Amazon Transcribe标记有害语音并对其进行分类，从而最大限度地减少必须手动处理的数据量。这使内容主持人能够快速有效地管理其平台上的话语。

有毒言语类别包括：

- 亵渎：包含不礼貌、粗俗或令人反感的单词、短语或首字母缩略词的言论。
- 仇恨言论：基于身份（例如种族、民族、性别、宗教、性取向、能力和国籍）批评、侮辱、谴责个人或群体或群体或使其失去人性的言论。
- 性的：通过直接或间接提及身体部位、身体特征或性别来表达性兴趣、活动或性兴奋的言论。
- 侮辱: 包括贬低、羞辱、嘲笑、侮辱或贬低语言的言论。这种语言也被标记为欺凌。
- 暴力或威胁：包括试图对个人或群体造成痛苦、伤害或敌意的威胁的言论。
- 图形：使用视觉描述性且不愉快的生动图像的演讲。这种语言通常是故意使用冗长的，以加剧接受者的不适感。
- 骚扰或虐待：旨在影响接受者心理健康的言论，包括贬低和客观化的术语。这种语言也被标记为骚扰。

毒性检测会分析语音片段（自然停顿之间的语音），并为这些片段分配可信度分数。置信度分数是介于 0 和 1 之间的值。置信度分数越大表示该内容在相关类别中成为有害言语的可能性越大。您可以使用这些置信度分数为您的用例设置适当的毒性检测阈值。

> **Note**
>
> 毒性检测仅适用于美国英语的批量转录(en-US)。

查看输出示例采用 JSON 格式。

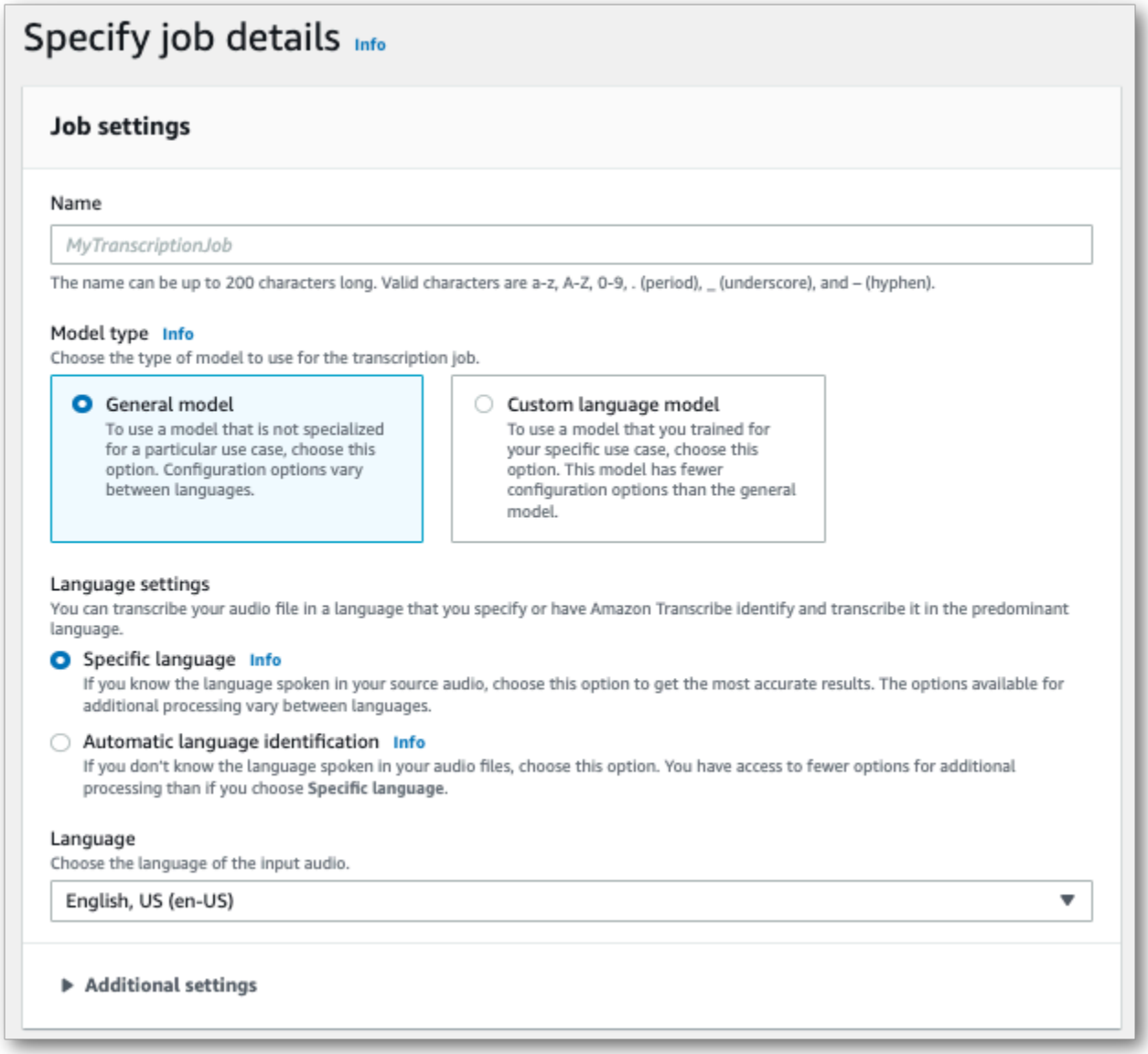
使用有毒语音检测

在批量转录中使用有毒语音检测

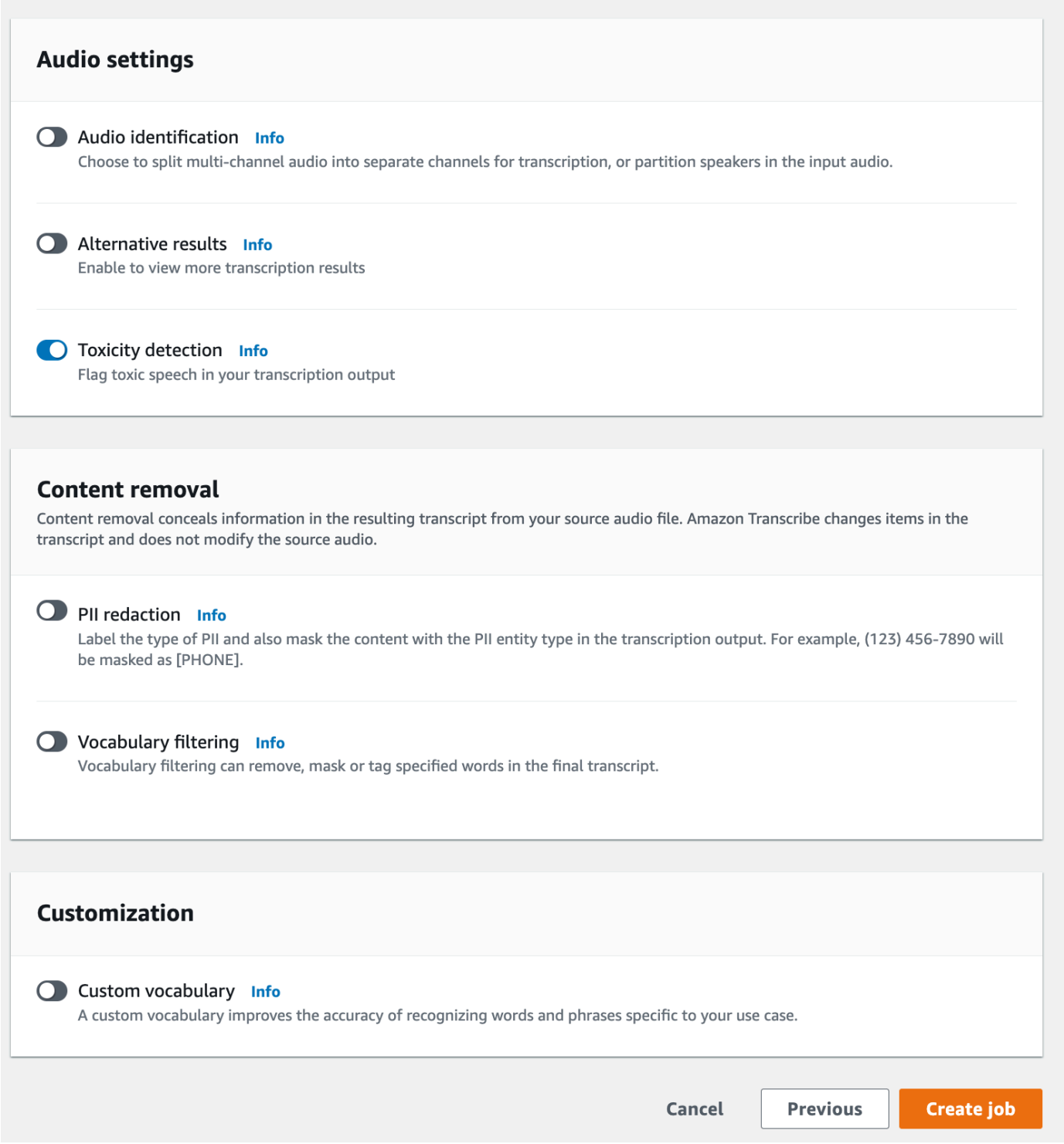
要在批量转录中使用有毒语音检测，请参阅以下示例：

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录职位，然后选择创建作业（右上）。这会打开指定作业详情页面。



3. 在指定作业详情页面，如果你愿意，你也可以启用 PII 编辑。请注意，毒性检测不支持列出的其他选项。选择下一步。这会带你去配置作业-可选页面。在里面音频设置面板，选择毒性检测。



4. 选择创建作业来运行你的转录工作。

5. 转录工作完成后，您可以从以下地址下载成绩单下载转录作业详情页面中的下拉菜单。

AWS CLI

此示例使用start-transcription-job命令和ToxicityDetection参数。有关更多信息，请参阅 StartTranscriptionJob 和 ToxicityDetection。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
--toxicity-detection ToxicityCategories=ALL
```

这是另一个使用start-transcription-job命令，以及包括毒性检测在内的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://filepath/my-first-toxicity-job.json
```

这个文件my-first-toxicity-job.json包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "ToxicityDetection": [
      {
         "ToxicityCategories": [ "ALL" ]
      }
   ]
}
```

AWS SDK for Python (Boto3)

此示例使用AWS SDK for Python (Boto3)启用ToxicityDetection对于开始转录作业方法。有关更多信息，请参阅 StartTranscriptionJob 和 ToxicityDetection。

有关其他示例，请使用AWS软件开发工具包，包括特定功能、场景和跨服务示例，请参阅使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS章。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    ToxicityDetection = [
        {
            'ToxicityCategories': ['ALL']
        }
    ]
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

输出示例

有毒语音会在您的转录输出中进行标记和分类。对每个有害言语实例进行分类并分配置信度分数（介于 0 到 1 之间的值）。置信度值越大表示该内容在指定类别中成为有毒言语的可能性越大。

输出示例 (JSON)

以下是 JSON 格式的示例输出，显示了分类的有害语音以及相关的可信度分数。


```
{
    "jobName": "my-toxicity-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [...],
        "items":[...],
        "toxicity_detection": [
            {
                "text": "What the * are you doing man? That's why I didn't want to play
 with your * .  man it was a no, no I'm not calming down * man. I well I spent I spent
 too much * money on this game.",
                "toxicity": 0.7638,
                "categories": {
                    "profanity": 0.9913,
                    "hate_speech": 0.0382,
                    "sexual": 0.0016,
                    "insult": 0.6572,
                    "violence_or_threat": 0.0024,
                    "graphic": 0.0013,
                    "harassment_or_abuse": 0.0249
                },
                "start_time": 8.92,
                "end_time": 21.45
            },
            Items removed for brevity
            {
                "text": "What? Who? What the * did you just say to me? What's your
 address? What is your * address? I will pull up right now on your * * man. Take your *
 back to , tired of this **.",
                "toxicity": 0.9816,
                "categories": {
                    "profanity": 0.9865,
                    "hate_speech": 0.9123,
                    "sexual": 0.0037,
                    "insult": 0.5447,
                    "violence_or_threat": 0.5078,
                    "graphic": 0.0037,
                    "harassment_or_abuse": 0.0613
                },
```

```
                "start_time": 43.459,
                "end_time": 54.639
            },
        ]
    },
    ...
    "status": "COMPLETED"
}
```

个人身份信息个人身份信息个人身份信息个人身份信息个人身份信息

Redaction 用于掩盖或删除您的笔录中以个人身份信息 (PII) 为形式的敏感内容。批量转录和流式转录之间Amazon Transcribe可以编辑的 PII 类型各不相同。要查看每种转录方法的 PII 列表，请参阅在批处理作业中编辑 PII和编辑或识别实时流中的 PII。使用流式转录，您还可以选择在不编辑 PII 的情况下标记 PII；有关输出示例，请参阅。PII 识别输出示例

启用密文后，您可以选择仅生成已编辑的笔录，或者同时生成已编辑的记录和未编辑的记录。如果您选择仅生成经过编辑的笔录，请注意，您的媒体是唯一存储完整对话的地方。如果您删除原始媒体，则不会记录未编辑的 PII。因此，谨慎的做法可能是除了生成经过编辑的笔录外，还生成未经编辑的笔录。

要了解有关使用批量转录进行 PII 编辑的更多信息，请参阅：在批处理作业中编辑 PII。

要了解有关PII编辑或通过流媒体转录进行识别的更多信息，请参阅：编辑或识别实时流中的 PII。

> **Important**
>
> 编辑功能旨在识别和删除敏感数据。但是，由于机器学习的预测性质，Amazon Transcribe可能无法识别和删除记录中敏感数据的所有实例。我们强烈建议您查看所有经过编辑的输出，以确保其满足您的需求。
> 《1996 年Health 保险可携性与责任法》(HIPAA) 等医疗隐私法 (例如《1996 年健康保险可携性与责任法》(HIPAA) 等医疗隐私法 (HIPAA) 的个人身份证号码 (HIPAA) 等医疗隐私法 (HIPAA)

有关编辑功能Amazon Transcribe的视频演练，请参阅使用内容编辑来识别和编辑 PII。

在批处理作业中编辑 PII

在批量转录作业期间编辑成绩单中的个人身份信息 (PII) 时，将记录正文中的每个已识别的 PII 实例Amazon Transcribe替换为成绩单正文[PII]中的每个已识别的 PII 实例。您还可以查看转录输出 word-for-word 部分中编辑的 PII 类型。有关输出示例，请参见编辑后的输出示例（批处理）。

美式英语支持批量转录编辑 (en-US)。编辑与语言识别不兼容。

已编辑和未编辑的记录都存储在同一个输出存储Amazon S3桶中。 Amazon Transcribe将它们存储在您指定的存储段或服务管理的默认Amazon S3存储段中。

Amazon Transcribe可以识别用于批量转录的 PII 类型

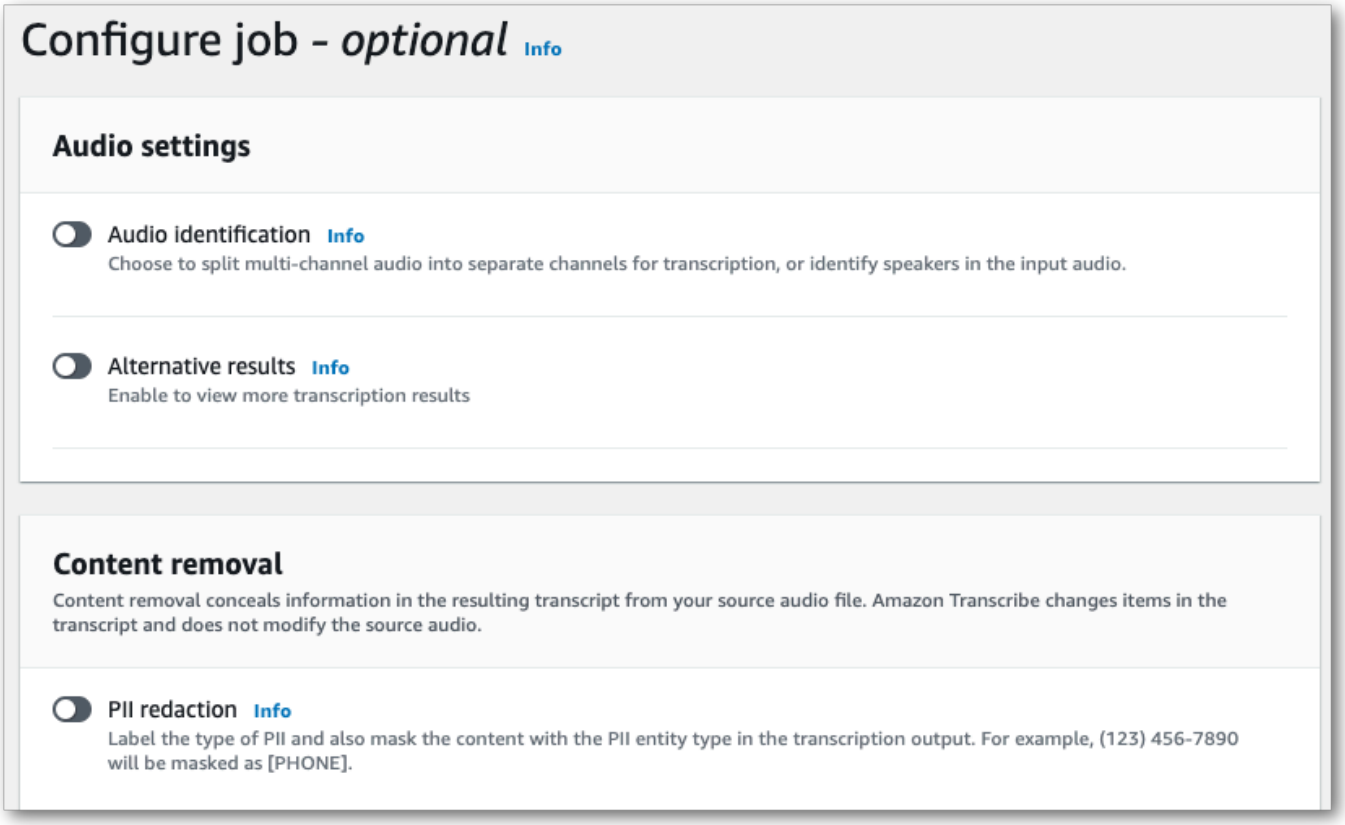
个人身份信息类型	描述
ADDRESS	物理地址，例如美国安尼敦大街 100 号或 123 号楼 #12 套房。地址可以包括街道、建筑物、位置、城市、州、国家、县、邮政编码、区域、社区等。
ALL	编辑或识别此表中列出的所有 PII 类型。
BANK_ACCOUNT_NUMBER	美国银行账号。它们的长度通常介于 10 到 12 位数之间，但当仅存在最后 4 位数字时，Amazon Transcribe也可以识别银行账号。
BANK_ROUTING	美国银行账户的路由号码。它们的长度通常为 9 位数，但当仅存在最后 4 位数字时Amazon Transcribe也可以识别路由号码。
CREDIT_DEBIT_CVV	VISA、 MasterCard Discover 信用卡和借记卡上存在的 3 位数信用卡验证码 (CVV)。在美国运通信用卡或借记卡中，它是一个 4 位数的数字代码。
CREDIT_DEBIT_EXPIRY	信用卡或借记卡的到期日。此数字通常为 4 位数，格式为月/年或 MM/YY。例如，Amazon Transcribe可以识别到期日期，例如 1 月 21 日、2021 年 1 月和 2021 年 1 月。
CREDIT_DEBIT_NUMBER	信用卡或借记卡的号码。这些数字的长度可以从 13 到 16 位不等，但当仅存在最后 4 位数字时，Amazon Transcribe也可以识别信用卡或借记卡号。
EMAIL	电子邮件地址，例如 efua.owusu@email.com。
NAME	个人的名字。此实体类型不包括先生、夫人、小姐或博士等头衔。Amazon Transcribe不将此实体类型应用于组织或地址所属的姓名。例如，

个人身份信息类型	描述
	将 John Doe 组织视为组织，将 Jane Doe Stre etAmazon Transcribe 识别为地址。
PHONE	电话号码。此实体类型还包括传真和寻呼机号码。
PIN	一个 4 位数的个人识别码 (PIN)，允许某人访问其银行账户信息。
SSN	社会安全号码 (SSN) 是一个 9 位数的数字，发放给美国公民、永久居民和临时工作居民。Amazon Transcribe当仅存在最后 4 位数字时，还能识别社会安全号码。

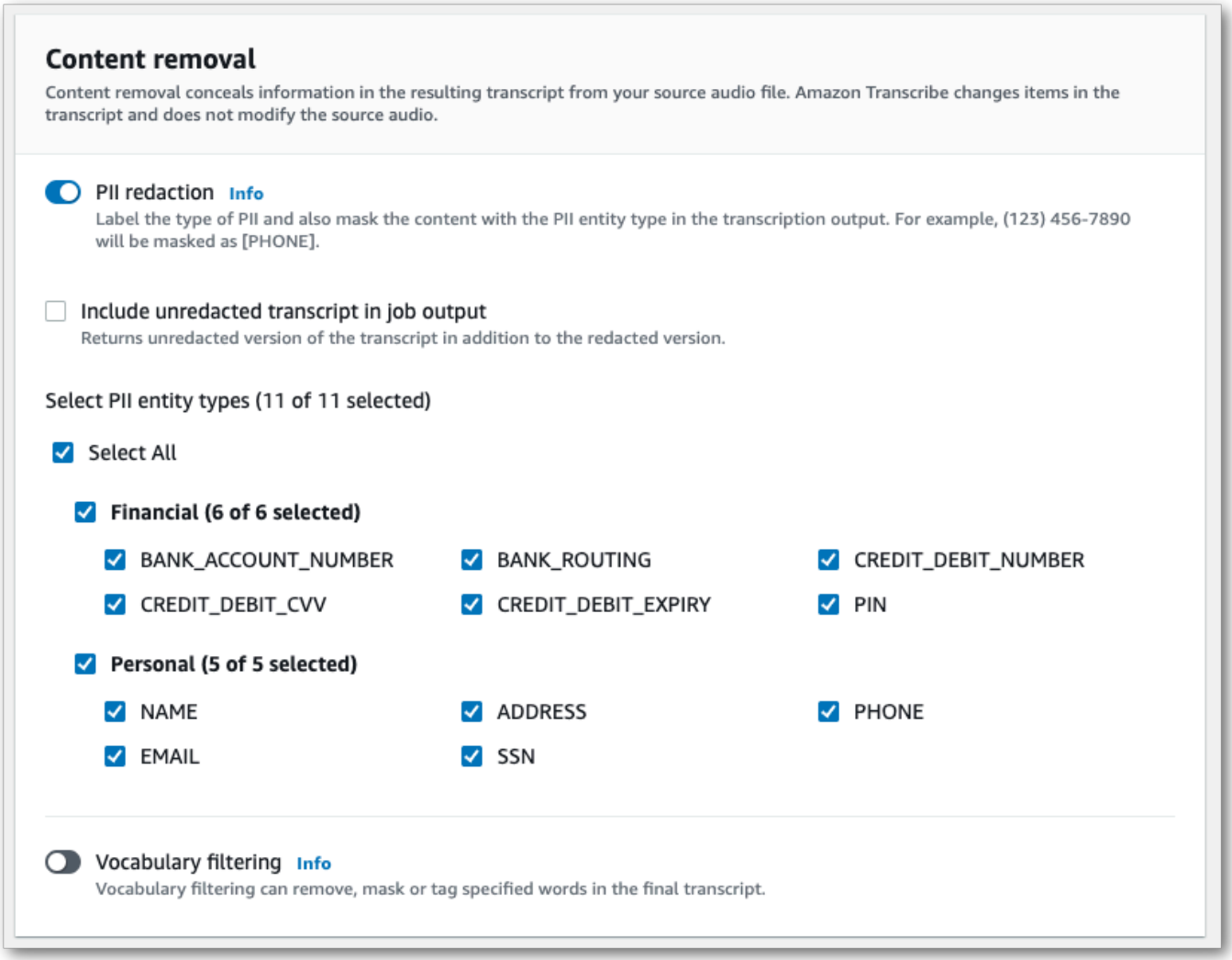
您可以使用AWS Management Console、AWS CLI或AWS SDK 启动批量转录作业。

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。
3. 在 “指定作业详细信息” 页面上填写所需字段后，选择 “下一步” 转到 “配置作业-可选” 页面。在这里，您将找到带有 PII 编辑开关的 “内容删除” 面板。



4. 选择 PII 编辑后，您可以选择要编辑的所有 PII 类型。如果您选择 “在作业输出中包含未编辑的脚本” 框，也可以选择使用未编辑的脚本。



5. 选择 “创建作业” 以运行转录作业。

AWS CLI

此示例使用start-transcription-job命令和content-redaction参数。有关更多信息，请参阅 StartTranscriptionJob 和 ContentRedaction。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
--language-code en-US \
```

```
--content-redaction
 RedactionType=PII,RedactionOutput=redacted,PiiEntityTypes=NAME,ADDRESS,BANK_ACCOUNT_NUMBER
```

这是使用该start-transcription-job方法的另一个示例，请求正文编辑了该任务的 PII。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://filepath/my-first-redaction-job.json
```

my-first-redaction-job.json 文件包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
      "MediaFileUri":  "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "ContentRedaction": {
      "RedactionOutput":"redacted",
      "RedactionType":"PII",
      "PiiEntityTypes": [
           "NAME",
           "ADDRESS",
           "BANK_ACCOUNT_NUMBER"
      ]
  }
}
```

AWS SDK for Python (Boto3)

此示例使用使用 start_transcription_job 方法的ContentRedaction参数来编辑内容。AWS SDK for Python (Boto3)有关更多信息，请参阅 StartTranscriptionJob 和 ContentRedaction。

有关使用AWS软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS章。

```
from __future__ import print_function
import time
import boto3
```

```
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    ContentRedaction = {
        'RedactionOutput':'redacted',
        'RedactionType':'PII',
        'PiiEntityTypes': [
            'NAME','ADDRESS','BANK_ACCOUNT_NUMBER'
        ]
    }
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

> **Note**
>
> 仅支持批量作业的 PIIAWS 区域：亚太地区（香港）、亚太地区（孟买）、亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）、亚太地区（东京）、（美国西部）、加拿大 GovCloud （中部）、欧洲（法兰克福）、欧洲（伦敦）、欧洲（巴黎）、中东（巴林）、南美洲（圣保罗）、美国东部（弗吉尼亚北部）、美国东部（俄亥俄州）、美国西部（俄勒冈）和美国西部（加利福尼亚北部）。

编辑或识别实时流中的 PII

在编辑直播转录中的个人身份信息 (PII) 时，将记录中的每个已识别的 PII 实例Amazon Transcribe替换为笔录`[PII]`中的每个已识别的 PII 实例。

用于直播转录的另一个选项是 PII 识别。激活 PII 识别时，将转录结果中的 PIIAmazon Transcribe 标记在Entities对象下方。有关输出示例，请参阅经过编辑的直播输出示例和PII 识别输出示例。

使用以下英语方言可以编辑和识别 PII，例如澳大利亚 (en-AU)、英国 (en-GB) 和美国 (en-US)。

只有在音频片段完成转录后，才能对直播任务执行 PII 识别和编辑。

Amazon Transcribe可以识别用于流式转录的 PII 类型

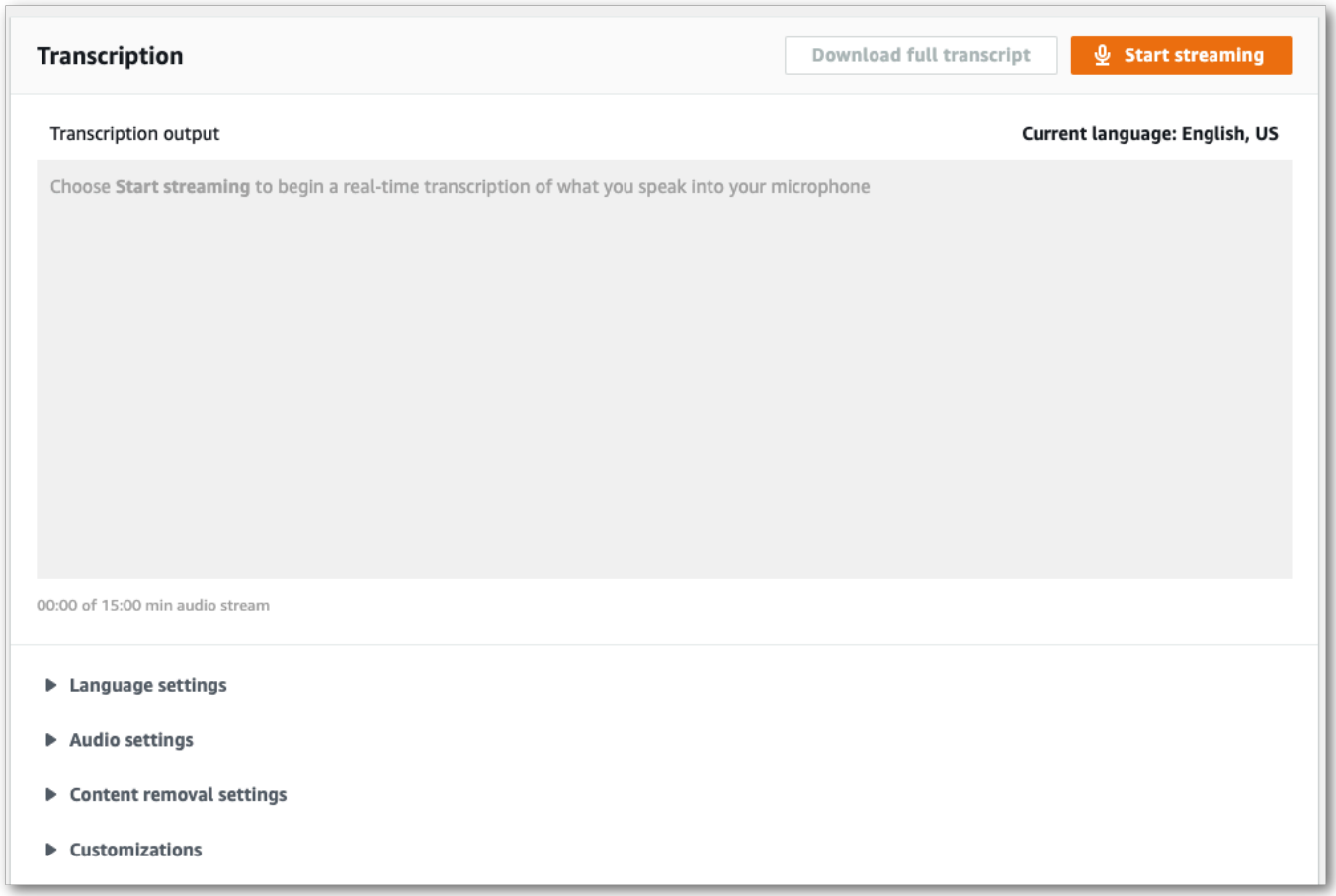
个人身份信息类型	描述
ADDRESS	物理地址，例如美国安尼敦大街 100 号或 123 号楼 #12 套房。地址可以包括街道、建筑物、位置、城市、州、国家、县、邮政编码、区域、社区等。
ALL	编辑或识别此表中列出的所有 PII 类型。
BANK_ACCOUNT_NUMBER	美国银行账号。它们的长度通常介于 10 到 12 位数之间，但当仅存在最后 4 位数字时，Amazon Transcribe也可以识别银行账号。
BANK_ROUTING	美国银行账户的路由号码。它们的长度通常为 9 位数，但当仅存在最后 4 位数字时Amazon Transcribe也可以识别路由号码。
CREDIT_DEBIT_CVV	VISA、 MasterCard Discover 信用卡和借记卡上存在的 3 位数信用卡验证码 (CVV)。在美国运通信用卡或借记卡中，它是一个 4 位数的数字代码。
CREDIT_DEBIT_EXPIRY	信用卡或借记卡的到期日。此数字通常为 4 位数，格式为月/年或 MM/YY。例如，Amazon Transcribe可以识别到期日期，例如 1 月 21 日、2021 年 1 月和 2021 年 1 月。
CREDIT_DEBIT_NUMBER	信用卡或借记卡的号码。这些数字的长度可以从 13 到 16 位不等，但当仅存在最后 4 位数字时，Amazon Transcribe也可以识别信用卡或借记卡号。

个人身份信息类型	描述
EMAIL	电子邮件地址，例如 efua.owusu@email.com。
NAME	个人的名字。此实体类型不包括先生、夫人、小姐或博士等头衔。Amazon Transcribe不将此实体类型应用于组织或地址中的姓名。例如，Amazon Transcribe将 John Doe 组织视为组织，将 Jane Doe Street 视为地址。
PHONE	电话号码。此实体类型还包括传真和寻呼机号码。
PIN	一个 4 位数的个人识别码 (PIN)，允许某人访问其银行账户信息。
SSN	社会安全号码 (SSN) 是一个 9 位数的数字，发放给美国公民、永久居民和临时工作居民。Amazon Transcribe当仅存在最后 4 位数字时，还能识别社会安全号码。

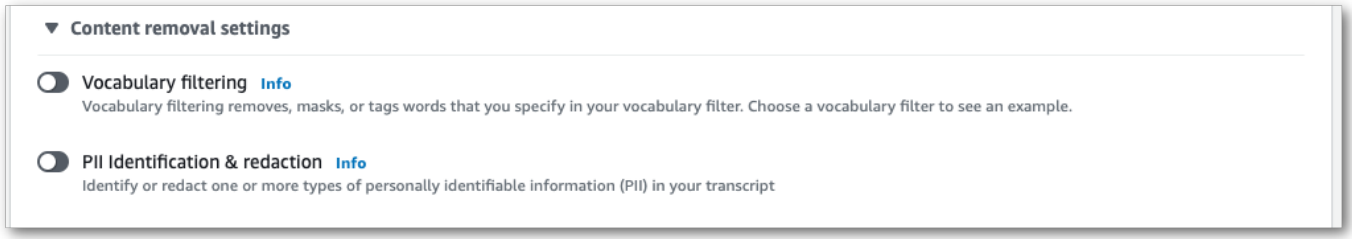
您可以使用AWS Management Console、 WebSocket或 HTTP/2 开始流式转录。

AWS Management Console

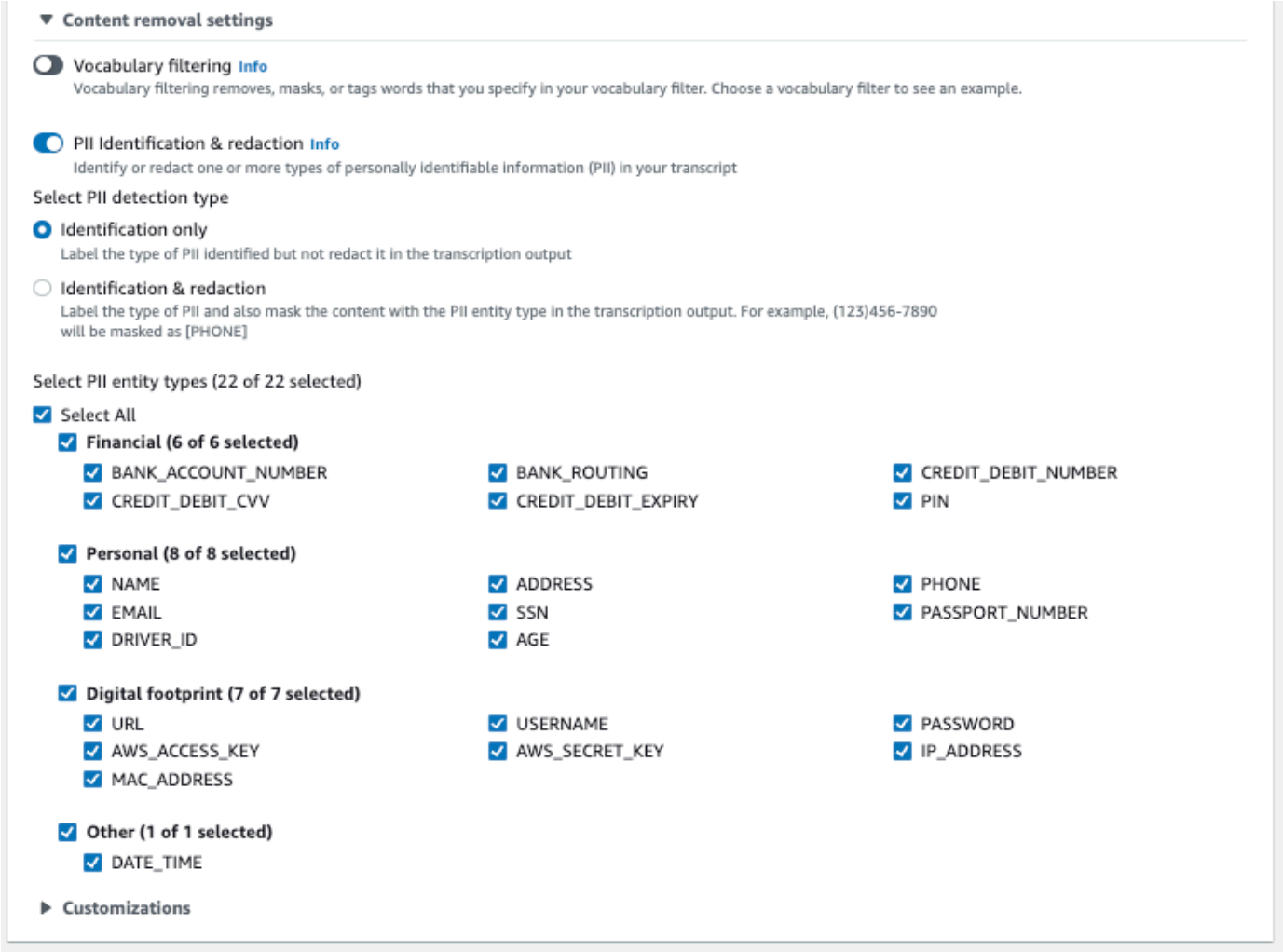
1. 登录到 AWS Management Console。
2. 在导航窗格中，选择 Real-time transcription (实时转录)。向下滚动到内容删除设置，如果该字段已最小化，请将其展开。



3. 打开 PII 识别和编辑。



4. 选择 “仅限识别” 或 “识别和编辑”，然后选择要在笔录中识别或编辑的 PII 实体类型。



5. 您现在已准备好。选择 “开始直播” 并开始讲话。要结束听写，请选择 “停止直播”。

WebSocket 流

此示例创建了一个在 WebSocket 数据流中使用 PII 编辑（或 PII 识别）的预签名 URL。为了便于阅读，已增加了换行符。有关将 WebSocket 直播与一起使用的更多信息Amazon Transcribe，请参阅设置直 WebSocket 播。有关参数的更多详细信息，请参阅StartStreamTranscription。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
```

```
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
&pii-entity-types=NAME,ADDRESS
&content-redaction-type=PII (or &content-identification-type=PII)
```

您不能在同一个请求content-redaction-type中同时使用content-identification-type和。

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

HTTP/2 流转录

此示例创建了一个启用 PII 识别或 PII 编辑的 HTTP/2 请求。有关使用 HTTP/2 流式传输的更多信息Amazon Transcribe，请参阅设置 HTTP/2 音频流。有关特定参数和标题的更多详细信息Amazon Transcribe，请参阅StartStreamTranscription。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: 20220208T235959Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key/20220208/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-transcribe-content-identification-type: PII (or x-amzn-transcribe-content-
redaction-type: PII)
x-amzn-transcribe-pii-entity-types: NAME,ADDRESS
transfer-encoding: chunked
```

您不能在同一个请求content-redaction-type中同时使用content-identification-type和。

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

> **Note**
>
> 只有亚太地区（首尔）、亚太地区（伦敦）、美国东部（弗吉尼亚州北部）、欧洲地区（伦敦）、美国东部（弗吉尼亚州北部）、欧洲地区（伦敦）、美国东部（弗吉尼亚州北部）、美国东部（弗吉尼亚州北部）、美国东部（俄亥俄州）、美国西部（俄亥俄州）、和美国西部（俄勒冈州）。AWS 区域

PII 编辑和识别输出示例

以下示例显示了批处理和流式处理作业的编辑输出，以及流式处理作业的 PII 识别。

使用内容编辑的转录作业会生成两种类型的`confidence`值。自动语音识别 (ASR) 置信度表示 `type` 为 `pronunciation` 或 `punctuation` 的项目是特定的发音。在以下脚本输出中，`Good`该单词的 `aconfidence` 为`1.0`。该置信度Amazon Transcribe值表明，100% 确信该笔录中说的 “好” 一词。`[PII]` 标签的 `confidence` 值是对于标记为要修订的语音确实是 PII 的置信度。在以下记录输出中`confidence`，Amazon Transcribeof`0.9999` 表示有 99.99% 的人确信其在记录中编辑的实体是 PII。

编辑后的输出示例（批处理）

```
{
    "jobName": "my-first-transcription-job",
    "accountId": "111122223333",
    "isRedacted": true,
    "results": {
        "transcripts": [
            {
                "transcript": "Good morning, everybody. My name is [PII], and today I
 feel like
                sharing a whole lot of personal information with you. Let's start with
 my Social
                Security number [PII]. My credit card number is [PII] and my C V V code
 is [PII].
                I hope that Amazon Transcribe is doing a good job at redacting that
 personal
                information away. Let's check."
            }
        ],
        "items": [
```

```
            {
                "start_time": "2.86",
                "end_time": "3.35",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Good"
                    }
                ],
                "type": "pronunciation"
            },
            Items removed for brevity
            {
                "start_time": "5.56",
                "end_time": "6.25",
                "alternatives": [
                    {
                        "content": "[PII]",
                        "redactions": [
                            {
                                "confidence": "0.9999",
                                "type": "NAME",
                                "category": "PII"
                            }
                        ]
                    }
                ],
                "type": "pronunciation"
            },
            Items removed for brevity
        ],
    },
    "status": "COMPLETED"
}
```

以下是未经编辑的笔录以供比较：

```
{
    "jobName": "job id",
    "accountId": "111122223333",
    "isRedacted": false,
    "results": {
        "transcripts": [
```

```
            {
                "transcript": "Good morning, everybody. My name is Mike, and today I
 feel like
                sharing a whole lot of personal information with you. Let's start with
 my Social
                Security number 000000000. My credit card number is 5555555555555555
                and my C V V code is 000. I hope that Amazon Transcribe is doing a good
 job
                at redacting that personal information away. Let's check."
            }
        ],
        "items": [
            {
                "start_time": "2.86",
                "end_time": "3.35",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Good"
                    }
                ],
                "type": "pronunciation"
            },
            Items removed for brevity
            {
                "start_time": "5.56",
                "end_time": "6.25",
                "alternatives": [
                    {
                        "confidence": "0.9999",
                        "content": "Mike",
                     {
                ],
                "type": "pronunciation"
            },
            Items removed for brevity
        ],
    },
    "status": "COMPLETED"
}
```

经过编辑的直播输出示例


```
{
    "TranscriptResultStream": {
        "TranscriptEvent": {
            "Transcript": {
                "Results": [
                    {
                        "Alternatives": [
                            {
                                "Transcript": "my name is [NAME]",
                                "Items": [
                                    {
                                        "Content": "my",
                                        "EndTime": 0.3799375,
                                        "StartTime": 0.0299375,
                                        "Type": "pronunciation"
                                    },
                                    {
                                        "Content": "name",
                                        "EndTime": 0.5899375,
                                        "StartTime": 0.3899375,
                                        "Type": "pronunciation"
                                    },
                                    {
                                        "Content": "is",
                                        "EndTime": 0.7899375,
                                        "StartTime": 0.5999375,
                                        "Type": "pronunciation"
                                    },
                                    {
                                        "Content": "[NAME]",
                                        "EndTime": 1.0199375,
                                        "StartTime": 0.7999375,
                                        "Type": "pronunciation"
                                    }
                                ],
                                "Entities": [
                                    {
                                        "Content": "[NAME]",
                                        "Category": "PII",
                                        "Type": "NAME",
                                        "StartTime" : 0.7999375,
```

```
                                    "EndTime" : 1.0199375,
                                    "Confidence": 0.9989
                                }
                            ]
                        }
                    ],
                    "EndTime": 1.02,
                    "IsPartial": false,
                    "ResultId": "12345a67-8bc9-0de1-2f34-a5b678c90d12",
                    "StartTime": 0.0199375
                }
            ]
        }
    }
}
```

PII 识别输出示例

PII 识别是一项附加功能，可用于直播转录作业。已识别的 PII 列在每个分段的Entities部分中。

```
{
    "TranscriptResultStream": {
        "TranscriptEvent": {
            "Transcript": {
                "Results": [
                    {
                        "Alternatives": [
                            {
                                "Transcript": "my name is mike",
                                "Items": [
                                    {
                                        "Content": "my",
                                        "EndTime": 0.3799375,
                                        "StartTime": 0.0299375,
                                        "Type": "pronunciation"
                                    },
                                    {
                                        "Content": "name",
                                        "EndTime": 0.5899375,
                                        "StartTime": 0.3899375,
                                        "Type": "pronunciation"
                                    },
```

```
                                    {
                                        "Content": "is",
                                        "EndTime": 0.7899375,
                                        "StartTime": 0.5999375,
                                        "Type": "pronunciation"
                                    },
                                    {
                                        "Content": "mike",
                                        "EndTime": 0.9199375,
                                        "StartTime": 0.7999375,
                                        "Type": "pronunciation"

                                    }
                                ],
                                "Entities": [
                                    {
                                        "Content": "mike",
                                        "Category": "PII",
                                        "Type": "NAME",
                                        "StartTime" : 0.7999375,
                                        "EndTime" : 1.0199375,
                                        "Confidence": 0.9989
                                    }
                                ]
                            }
                        ],
                        "EndTime": 1.02,
                        "IsPartial": false,
                        "ResultId": "12345a67-8bc9-0de1-2f34-a5b678c90d12",
                        "StartTime": 0.0199375
                    }
                ]
            }
        }
    }
}
```

创建视频字幕

Amazon Transcribe支持 WebVTT (*.vtt) 和 SubRip (*.srt) 输出用作视频字幕。设置批量视频转录作业时，您可以选择一种或两种文件类型。使用字幕功能时，将生成您选择的字幕文件和常规脚本文件（包含其他信息）。字幕和转录文件输出到同一个目的地。

字幕在朗读文字的同时显示，在自然暂停或说话者说完话之前，字幕一直可见。请注意，如果您在转录请求中启用字幕且音频不包含语音，则不会创建字幕文件。

> **Important**
>
> Amazon Transcribe字幕输出使用默认0的起始索引，这与更广泛使用的值不同1。如果您需要的起始索引为1，则可以在AWS Management Console或 API 请求中使用`OutputStartIndex`参数指定此索引。

使用不正确的起始索引可能会导致与其他服务的兼容性错误，因此在创建字幕之前，请务必验证所需的起始索引。如果您不确定要使用哪个值，我们建议您选择1。有关更多信息 `Subtitles`，请参阅。

字幕支持的功能：

- 内容编辑- 任何经过编辑的内容在字幕和常规脚本输出文件中都反映为 `PII` “”。音频没有改变。
- 词汇过滤器 — 字幕文件由转录文件生成，因此您在标准转录输出中筛选的任何单词也会在字幕中过滤。过滤后的内容以空格形式或***在您的脚本和字幕文件中显示。音频没有改变。
- 扬声器分布 — 如果给定的字幕片段中有多个扬声器，则使用短划线来区分每个说话者。这适用于WebVTT 和SubRip格式；例如：
  - --第 1 人说的文字
  - --第 2 人说的文字

字幕文件存储在与转录输出相同Amazon S3的位置。

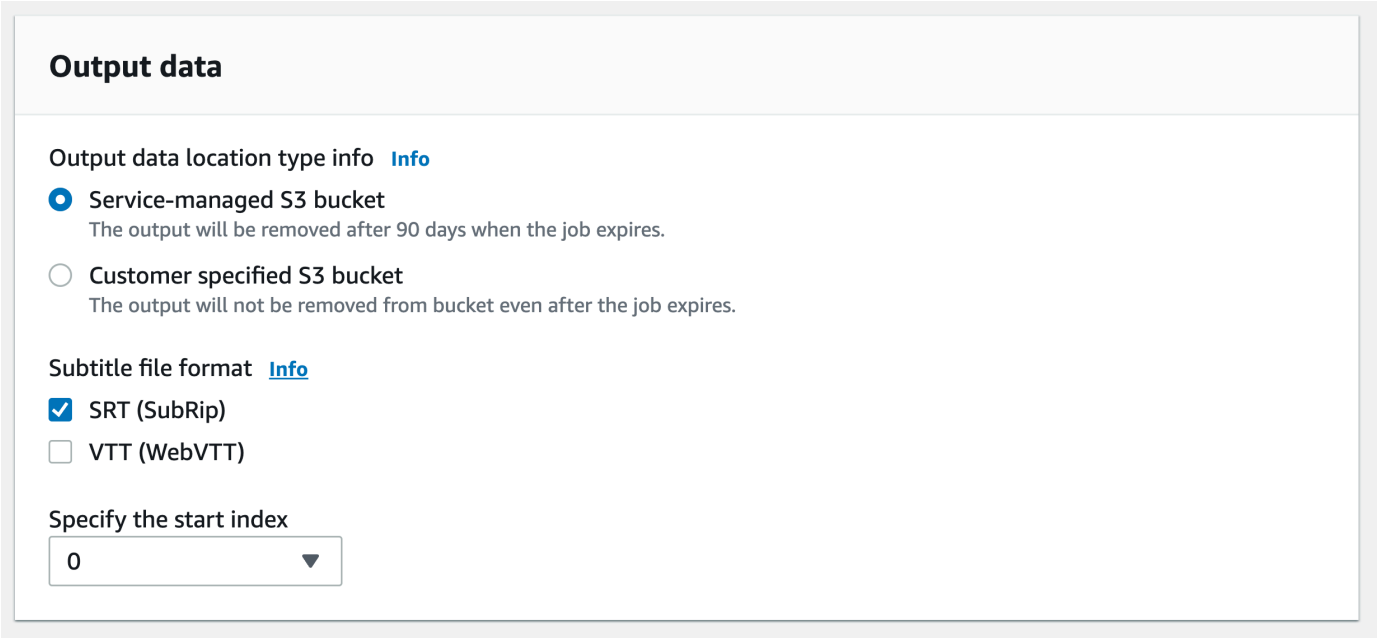
有关创建字幕的视频演练，请参阅亚马逊 Transcribe Video Snacks：无需编写任何代码即可创建视频字幕。

生成字幕文件

您可以使用AWS Management Console、AWS CLI或 AWSSDK 创建字幕文件；请参阅以下示例：

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择转录作业，然后选择创建作业（右上角）。这将打开 “指定作业详细信息” 页面。字幕选项位于输出数据面板中。
3. 为字幕文件选择所需的格式，然后为起始索引选择一个值。请注意，Amazon Transcribe默认值为0，但1使用得更广泛。如果您不确定要使用哪个值，我们建议您选择1，因为这可能会提高与其他服务的兼容性。



4. 在 “指定作业详细信息” 页面上填写您想要包含的任何其他字段，然后选择 “下一步”。这将带您进入配置作业-可选页面。
5. 选择 “创建作业” 以运行您的转录作业。

AWS CLI

此示例使用start-transcription-job命令和Subtitles参数。有关更多信息，请参阅 StartTranscriptionJob 和 Subtitles。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--output-key my-output-files/ \
```

```
--language-code en-US \
--subtitles Formats=vtt,srt,OutputStartIndex=1
```

这是使用start-transcription-job命令的另一个示例，以及为该任务添加字幕的请求正文。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://my-first-subtitle-job.json
```

my-first-subtitle-job.json 文件包含以下请求正文。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "Subtitles": {
        "Formats": [
            "vtt","srt"
        ],
        "OutputStartIndex": 1
   }
}
```

AWS SDK for Python (Boto3)

此示例使用 start_transcription_ job 方法的Subtitles参数来添加字幕。AWS SDK for Python (Boto3) 有关更多信息，请参阅 StartTranscriptionJob 和 Subtitles。

有关使用 AWS SDK 的其他示例，包括特定功能、场景和跨服务示例，请参阅本章。使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
```

```
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Subtitles = {
        'Formats': [
            'vtt','srt'
        ],
        'OutputStartIndex': 1
   }
)

while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

使用“呼叫分析”分析呼叫中心音频

使用 Amazon Transcribe 呼叫分析深入了解客户与代理的互动。呼叫分析专为呼叫中心音频而设计，可自动为您提供与每个呼叫和每个参与者相关的宝贵数据。您还可以在仔细研究通话过程中特定时刻的数据。例如，您可以将通话前几秒钟的客户情绪与通话进行到最后四分之一时的客户情绪进行比较，以了解您的座席是否提供了积极正面的客户体验。下一节中列出了其它用例示例。

呼叫分析可用于通话后转录和实时转录。如果您正在转录 Amazon S3 存储桶中的文件，则是在进行通话后转录。如果您要转录音频流，则将执行实时转录。这两种转录方法提供了不同的呼叫分析见解和特征。有关每种方法的更多详细信息，请参阅通话后分析和实时通话分析。

借助实时呼叫分析转录，您还可以在请求中加入通话后分析。您的电话后分析记录存储在您在请求中指定的 Amazon S3 存储桶中。有关更多信息，请参阅通过实时转录进行通话后分析。

> 专用于呼叫分析的 API 操作
>
> 通话
> 后：CreateCallAnalyticsCategory、DeleteCallAnalyticsCategory、DeleteCallAnaly
> 实时：StartCallAnalyticsStreamTranscription，
> StartCallAnalyticsStreamTranscriptionWebSocket

常见使用案例

通话后转录：

- 监控一段时间内的问题频率：使用通话分类来识别转录中重复出现的关键字。
- 深入了解您的客户服务体验：使用通话特点（非通话时间、通话时间、中断、音量、通话速度）和情绪分析，来确定客户问题在通话期间是否得到了适当解决。
- 确保监管合规或遵守公司政策：为公司特定的问候语或免责声明设置关键字和短语，以验证您的座席行为是否符合监管要求。
- 改善对客户个人数据的处理：在转录输出或音频文件中使用 PII 编辑来帮助保护客户隐私。
- 改善员工培训：使用标准（情绪、非通话时间、中断、通话速度）来标记可以用作正面或负面客户互动示例的转录。
- 衡量员工在创造积极客户体验方面的效率：使用情绪分析来衡量您的座席是否能够随着通话的进行，将负面的客户情绪转化为正面积极的客户情绪。

- 改善数据整理：根据自定义类别（包括关键字和短语、情绪、通话时间和中断）对通话进行标记和排序。
- 使用生成式人工智能总结通话的重要方面：使用生成式通话摘要获取转录的简明摘要，其中包括通话中讨论的问题、操作项目和结果等关键组成部分。

实时转录：

- 实时减少上报：为关键短语（例如客户说“要与经理交谈”）设置实时警报，以便在客户开始上报时进行标记。您可以使用实时类别匹配来创建实时警报。
- 改善对客户数据的处理：在转录输出中使用 PII 识别或 PII 编辑，来帮助保护客户隐私。
- 识别自定义关键字和短语：使用自定义类别来标记通话中的特定关键字。
- 自动识别问题：使用自动问题检测功能简要概述通话中发现的所有问题。
- 衡量员工在创造积极客户体验方面的效率：使用情绪分析来衡量您的座席是否能够随着通话的进行，将负面的客户情绪转化为正面积极的客户情绪。
- 设置座席助手：利用您选择的见解为座席提供主动协助，帮助其解决客户来电。有关更多信息，请参阅使用亚马逊语言人工智能服务为您的联络中心提供实时呼叫分析和座席协助。

要将呼叫分析提供的功能与呼叫分析 Amazon Transcribe 和 Amazon Transcribe 医疗的功能进行比较，请参阅功能表。

要开始使用，请参阅启动通话后分析转录和启动实时呼叫分析转录。呼叫分析输出与标准转录作业的输出类似，但包含其它分析数据。要查看示例输出，请参阅通话后分析输出和实时呼叫分析输出。

注意事项和其它信息

在使用呼叫分析之前，请注意：

- 呼叫分析仅支持双声道音频，即座席在一个声道上，而客户在另一个声道上。
- 任务队列列任务队列始终为通话后分析作业启用，因此您只能同时进行 100 个呼叫分析作业。如果您想请求增加配额，请参阅 AWS 服务配额。
- 通话后分析作业的输入文件不能超过 500 MB，并且必须少于 4 小时。
- 如果使用类别，则必须先创建所有所需的类别，然后才能开始呼叫分析转录。任何新类别都不能应用于现有转录。要了解如何创建新类别，请参阅为通话后转录创建类别和为实时转录创建类别。
- 某些呼叫分析配额 Amazon Transcribe 与 Amazon Transcribe 医疗配额不同；有关详细信息，请参阅AWS 一般参考。

通过 Machine Learn AWS ing 博客深入了解

要了解呼叫分析选项的更多信息，请参阅：

- 使用亚马逊语言人工智能服务为您的联络中心提供通话后分析
- 使用亚马逊语言人工智能服务为您的联络中心提供实时呼叫分析和座席协助

要查看呼叫分析输出和功能示例，请查看我们的GitHub演示。我们还提供 JSON 转 Word 文档应用程序，用于将您的成绩单转换为 easy-to-read 格式。

区域可用性和配额

以下各项支持呼叫分析 AWS 区域：

区域	转录类型
ap-northeast-1 (东京)	post-call, real-time
ap-northeast-2 (首尔)	post-call, real-time
ap-south-1（孟买）	post-call
ap-southeast-1 (新加坡)	post-call
ap-southeast-2 (悉尼)	post-call, real-time
ca-central-1（加拿大中部）	post-call, real-time
eu-central-1 (法兰克福)	post-call, real-time
eu-west-2 (伦敦)	post-call, real-time
us-east-1（弗吉尼亚州北部）	post-call, real-time
us-west-2 (俄勒冈)	post-call, real-time

请注意，Amazon Transcribe、Amazon Transcribe Medical 和呼叫分析的区域支持有所不同。

如遇获取每个支持的区域的端点，请参阅《AWS 一般参考》中的服务端点。

有关与您的转录相关的配额列表，请参阅《AWS 一般参考》中的服务配额。有些配额可以根据要求进行更改。如果可调整列包含“是”，则可以请求增加。为此，请选择提供的链接。

通话后分析

呼叫分析提供通话后分析，这对于监控客户服务情况非常有用。

通话后转录提供了以下见解：

- 通话特征，包括通话时间、非通话时间、发言者音量、中断、通话速度、问题、结果和操作项目
- 生成式通话摘要，这会创建整个通话的简明摘要
- 自定义分类，您可以使用其中的规则来仔细研究特定的关键字和标准
- PII 编辑，可对您的文本转录和音频文件进行此编辑
- 说话者情绪，每个来电者在通话中不同时刻表现出的情绪

通话后的见解

本节详细介绍了可用于通话后分析转录的见解。

通话特点

通话特点特征使用以下标准来衡量座席与客户互动的质量：

- 中断：衡量一个参与者是否以及何时打断了另一个参与者的话。频繁中断可能与粗鲁或愤怒相关，也可能与一个或两个参与者的消极情绪相关。
- 音量：衡量每位参与者的说话音量。使用此指标来查看呼叫者或座席是否大声说话或者大喊大叫，这通常表示生气。该指标表示为标准化值（给定片段中每秒的话音水平），范围从 0 到 100，其中值越高表示声音越大。
- 非通话时间：衡量不含语音的时间段。使用此指标来查找是否存在长时间的静默，例如座席让客户等待的时间过长。
- 通话速度：衡量两个参与者的说话速度。如果一个参与者说话太快，理解力就会受到影响。该指标以每分钟单词数来衡量。
- 通话时间：衡量每个参与者在通话期间的说话时长，以毫秒为单位。使用此指标来帮助识别是否有一名参与者在通话中占据主导地位，或者对话是否平衡。
- 问题、结果和操作项目：从通话转录中识别问题、结果和操作项目。

以下是一个输出示例。

生成式通话摘要

生成式通话摘要（预览版）创建整个通话的简明摘要，从而捕获关键组成部分，例如通话原因、为解决问题而采取的步骤以及后续步骤。

通过使用生成式通话摘要，您可以：

- 减少在通话期间和通话后手动记笔记的需求。
- 提高客服人员的效率，因为他们可以花更多的时间与排队等待的通话者交谈，而不是处理通话后的工作。
- 加快主管审核速度，因为通话摘要的审核速度比整个转录的审核速度快得多。

要在通话后分析作业中使用生成式通话摘要，请参阅启用生成式通话摘要。有关示例输出，请参阅生成式通话摘要输出示例。生成式通话摘要是单独定价的（请参阅定价页面）。

> **Note**
>
> 目前，在 us-east-1 和 us-west-2 中提供了生成式通话摘要。以下英语方言支持该功能：澳大利亚英语 (en-AU)、英国英语 (en-GB)、印度英语 (en-IN)、爱尔兰英语 (en-IE)、苏格兰英语 (en-AB)、美国英语 (en-US) 和威尔士英语 (en-WL)。

自定义分类

使用通话分类可以标记通话中的关键字、短语、情绪或操作。我们的分类选项有助于您对上报情况进行分类，例如经常中断的负面情绪通话，或者将通话按特定类别进行整理，例如公司部门。

您可以添加到一个类别的标准包括：

- 非通话时间：客户和座席都不说话的时段。
- 中断：当客户或座席打断对方时。
- 客户或座席情绪：客户或座席在指定时间段内的感受。如果在指定时间段内，至少 50% 的对话回合（两个发言者 back-and-forth 之间）与指定的情绪相符，则 Amazon Transcribe 认为该情绪是匹配的。
- 关键字或短语：根据精确的短语匹配部分转录。例如，如果您为“我要和经理交谈”这句话设置了过滤器，则 Amazon Transcribe 会过滤出该短语的确切内容。

您也可以标记与之前的标准相反的情况（通话时间、没有中断、不存在情绪以及没有特定的短语）。

以下是一个输出示例。

有关类别的更多信息或要了解如何创建新类别，请参阅为通话后转录创建类别。

敏感数据编辑

敏感数据编辑会取代文本转录和音频文件中的个人身份信息 (PII)。经过编辑的转录将原始文本替换为 [PII]；经过编辑的音频文件将说出的个人信息替换为无声音频。此参数对于保护客户信息很有用。

> **Note**
>
> 美国英语 (en-US) 支持通话后 PII 编辑。

要查看使用此特征进行了编辑的 PII 列表，或要了解有关使用 Amazon Transcribe进行编辑的更多信息，请参阅 个人身份信息个人身份信息个人身份信息个人身份信息个人身份信息。

以下是一个输出示例。

情绪分析

情绪分析可以评估客户和座席在整个通话过程中的感受。该指标既可以表示为定量值（范围为 5 到 -5），也可以表示为定性值（positive、neutral、mixed 或 negative）。定量值按四等份之一和按单次通话提供；定性值按回合提供。

该指标可以帮助确定您的座席是否能够在通话结束时让心烦意乱的客户感到满意。

情感分析有效 out-of-the-box ，因此不支持自定义，例如模型训练或自定义类别。

以下是一个输出示例。

为通话后转录创建类别

通话后分析支持创建自定义类别，使您能够自定义转录分析，以便于极大程度地满足您的特定业务需求。

您可以根据需要创建任意数量的类别，以囊括一系列不同的场景。对于您创建的每个类别，必须创建 1 到 20 条规则。每条规则都基于以下四个标准之一：中断、关键字、非通话时间或情绪。有关在

CreateCallAnalyticsCategory 操作中使用这些条件的更多详细信息，请参阅通话后分析类别的规则标准一节。

如果您的媒体内容符合您在给定类别中指定的所有规则，则 Amazon Transcribe 会使用该类别标记您的输出。有关 JSON 输出中类别匹配的示例，请参阅呼叫分类输出。

以下是一些关于如何使用自定义类别的示例：

- 隔离具有特定特点的通话，例如以负面客户情绪结束的通话
- 通过标记和跟踪特定的一组关键字来识别客户问题的动态
- 监控合规性，例如座席在通话的最初几秒钟内说出（或省略）特定短语
- 通过标记多次出现座席中断和负面客户情绪的通话，深入了解客户体验
- 比较多个类别以衡量相关性，例如分析座席使用欢迎短语是否与积极的客户情绪相关

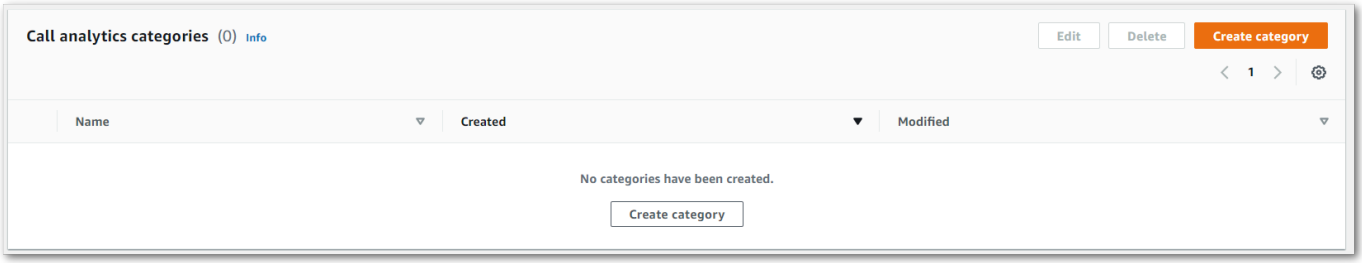
通话后类别与实时类别

创建新类别时，您可以指定是要将其创建为通话后分析类别 (POST_CALL)，还是创建为实时呼叫分析类别 (REAL_TIME)。如果您没有指定选项，则默认情况下，您的类别将创建为通话后类别。通话后分析转录完成后，您的输出中就会显示通话后分析类别的匹配项。

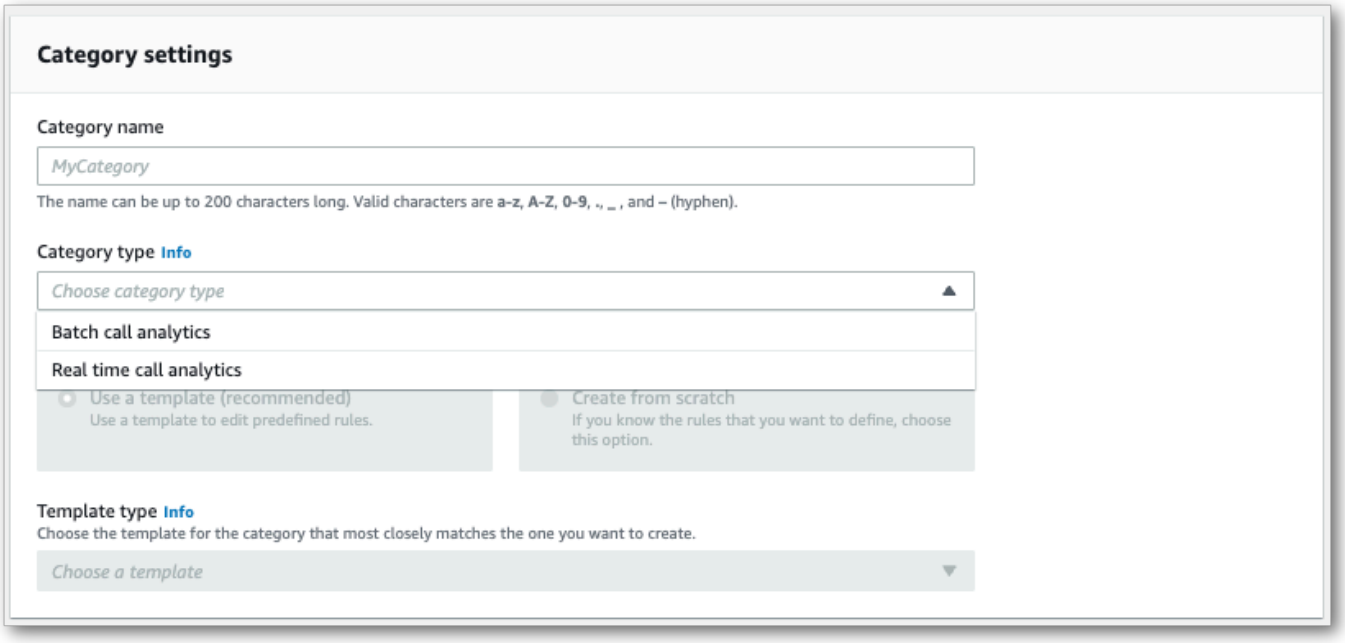
要为通话后分析创建新类别，您可以使用 AWS Management Console、AWS CLI 或 AWS SDK；有关示例，请参阅以下内容：

AWS Management Console

1. 在导航窗格中 Amazon Transcribe，选择Amazon Transcribe 呼叫分析。
2. 选择呼叫分析类别，之后您将进入呼叫分析类别页面。选择创建类别。

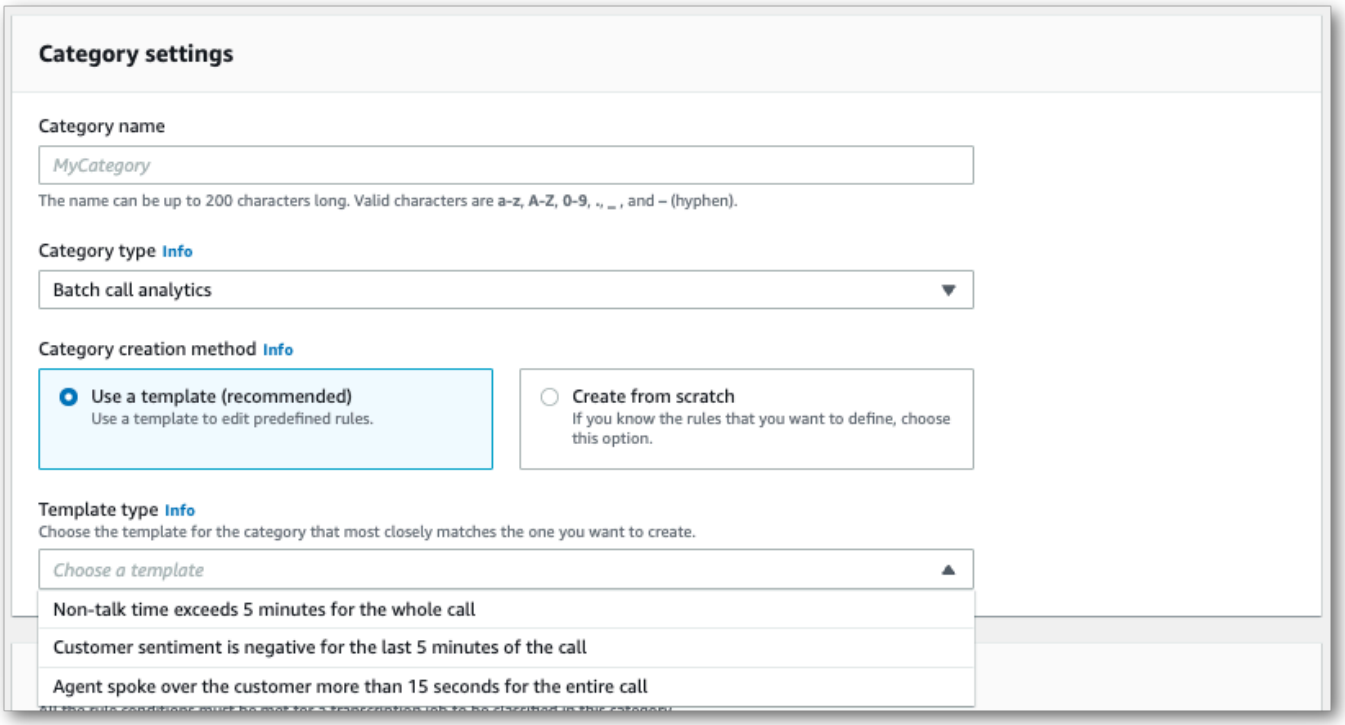


3. 您现在已进入创建类别页面。输入类别的名称，然后在类别类型下拉菜单中选择“批量呼叫分析”。

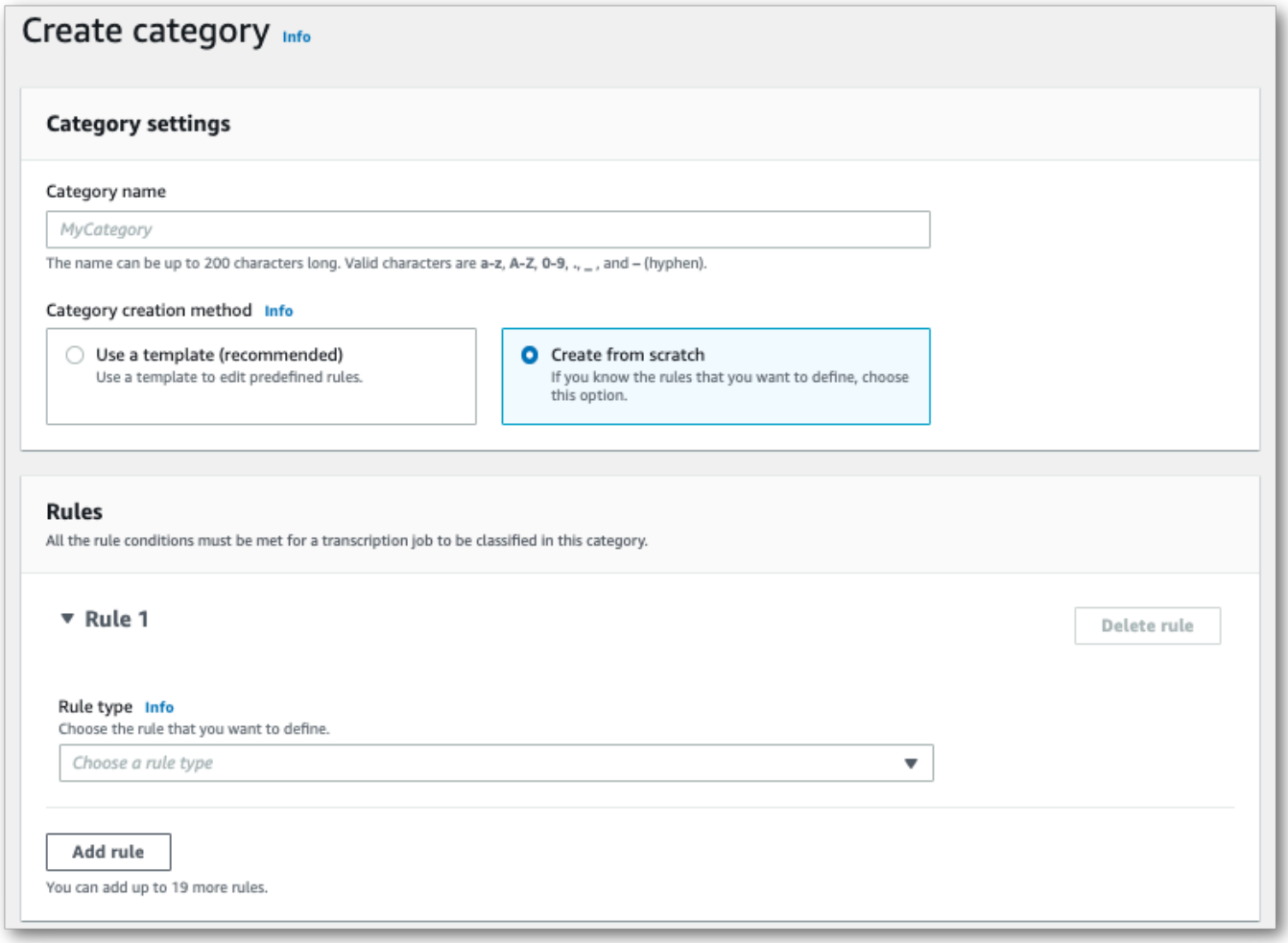


4. 您可以选择一个模板来创建您的类别，也可以从头开始制作一个模板。

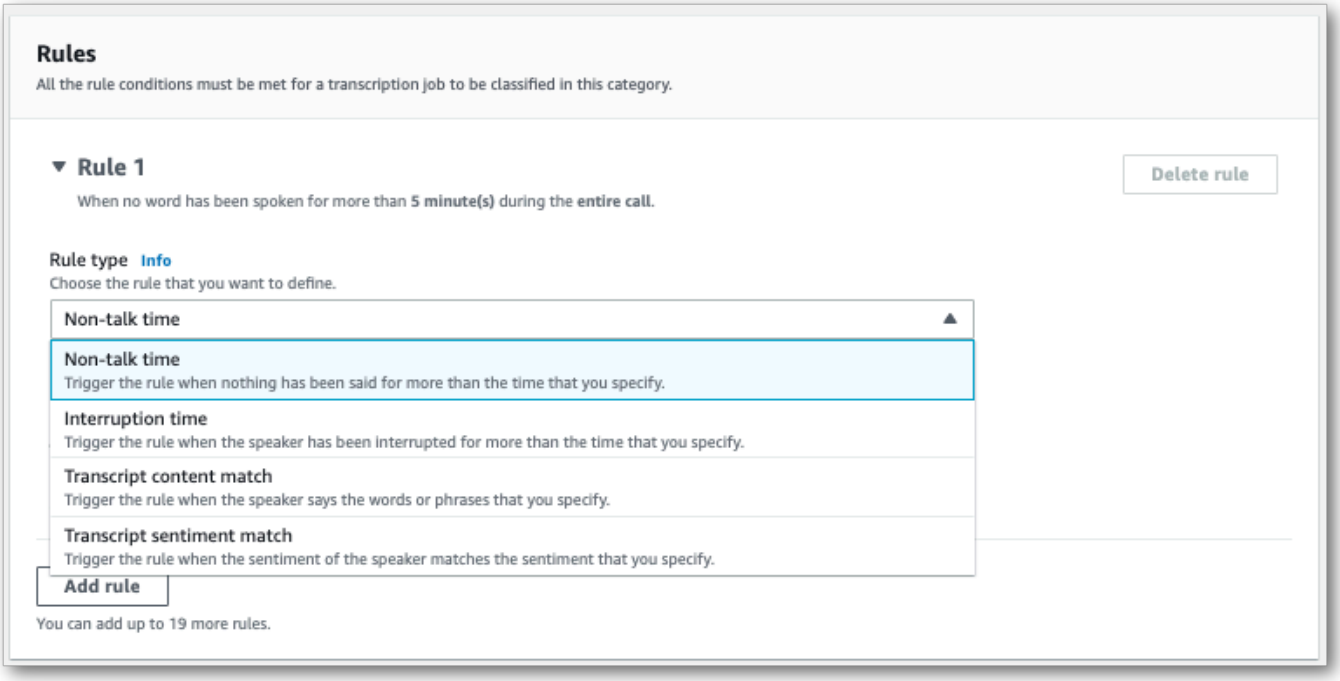
   如果使用模板：选择使用模板(建议)，选择所需的模板，然后选择创建类别。



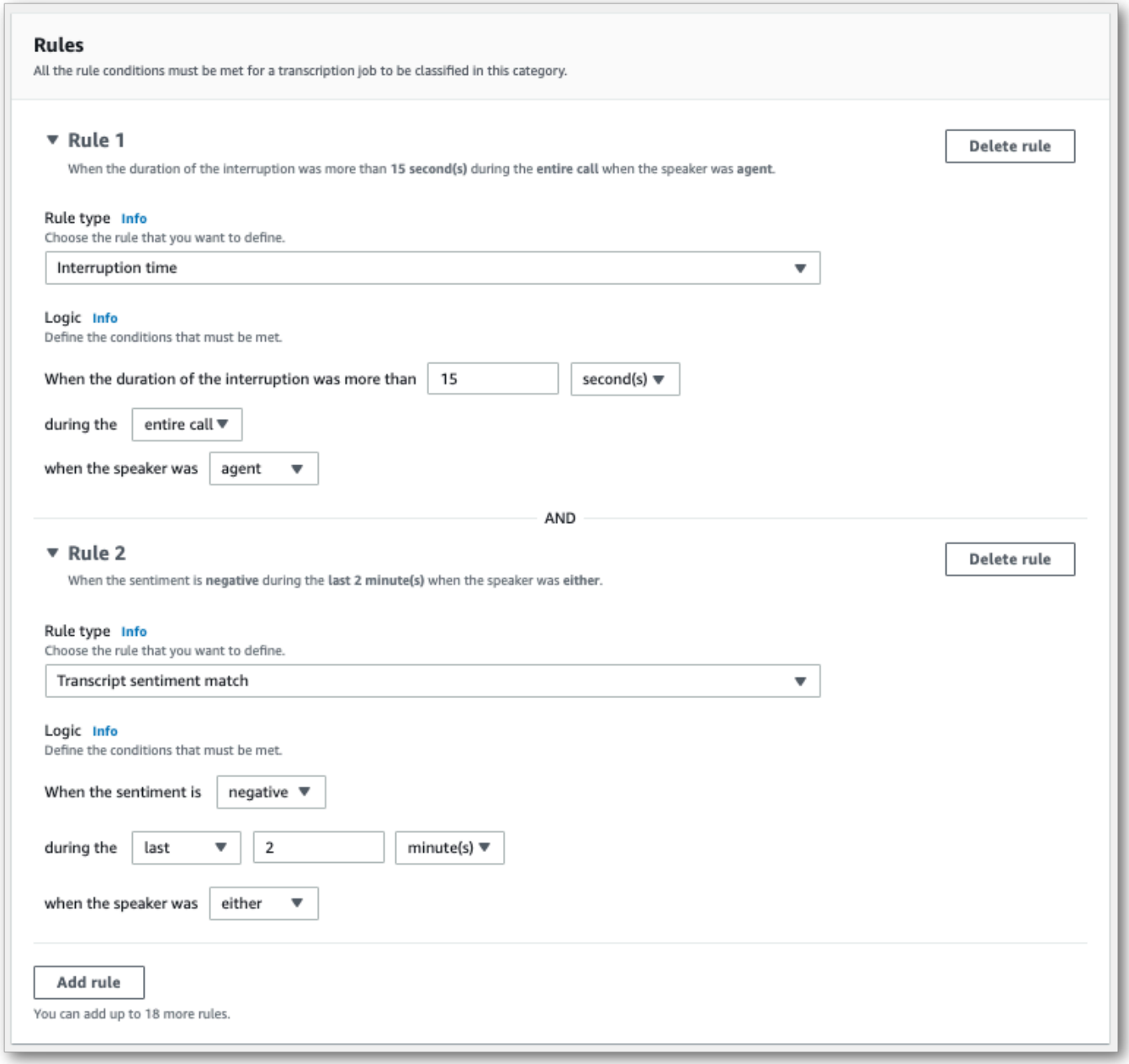
5. 如果要创建自定义类别：请选择从头开始创建。



6. 使用下拉菜单向您的类别添加规则。您最多可以为每个类别添加 20 条规则。



7. 以下是包含两条规则的类别示例：座席在通话期间打断客户超过 15 秒，以及客户或座席在通话的最后两分钟内感受到负面情绪。



8. 向类别添加完规则后，选择创建类别。

AWS CLI

此示例使用create-call-analytics-category命令。有关更多信息，请参阅 CreateCallAnalyticsCategory、CategoryProperties 和 Rule。

下面的示例创建了使用以下规则的类别：

- 客户在最初的 60000 毫秒被中断了。这些中断的持续时间至少持续了 10000 毫秒。

- 有一段沉默时间至少持续了 20000 毫秒，通话进行 10% 到通话进行 80% 之间。
- 该座席在通话中的某个时刻有负面情绪。
- 在通话的前 10000 毫秒内没有使用“欢迎”或“您好”这两个词。

此示例使用create-call-analytics-category命令和请求正文，该请求正文将多条规则添加到您的类别中。

```
aws transcribe create-call-analytics-category \
--cli-input-json file://filepath/my-first-analytics-category.json
```

my-first-analytics-category.json 文件包含以下请求正文。

```
{
  "CategoryName": "my-new-category",
  "InputType": "POST_CALL",
  "Rules": [
        {
            "InterruptionFilter": {
                "AbsoluteTimeRange": {
                    "First": 60000
                },
                "Negate": false,
                "ParticipantRole": "CUSTOMER",
                "Threshold": 10000
            }
        },
        {
            "NonTalkTimeFilter": {
                "Negate": false,
                "RelativeTimeRange": {
                    "EndPercentage": 80,
                    "StartPercentage": 10
                },
                "Threshold": 20000
            }
        },
        {
            "SentimentFilter": {
                "ParticipantRole": "AGENT",
                "Sentiments": [
                    "NEGATIVE"
                ]
```

```
            }
        },
        {
            "TranscriptFilter": {
                "Negate": true,
                "AbsoluteTimeRange": {
                    "First": 10000
                },
                "Targets": [
                    "welcome",
                    "hello"
                ],
                "TranscriptFilterType": "EXACT"
            }
        }
    ]
}
```

AWS SDK for Python (Boto3)

此示例使用 create_call_an alytics_category 方法的CategoryName和Rules参数创建类别。 AWS SDK for Python (Boto3) 有关更多信息，请参阅 CreateCallAnalyticsCategory、CategoryProperties 和 Rule。

有关使用 AWS 软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本章。使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS

下面的示例创建了使用以下规则的类别：

- 客户在最初的 60000 毫秒被中断了。这些中断的持续时间至少持续了 10000 毫秒。
- 有一段沉默时间至少持续了 20000 毫秒，通话进行 10% 到通话进行 80% 之间。
- 该座席在通话中的某个时刻有负面情绪。
- 在通话的前 10000 毫秒内没有使用“欢迎”或“您好”这两个词。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
category_name = "my-new-category"
transcribe.create_call_analytics_category(
    CategoryName = category_name,
```

```
    InputType = POST_CALL,
    Rules = [
        {
            'InterruptionFilter': {
                'AbsoluteTimeRange': {
                    'First': 60000
                },
                'Negate': False,
                'ParticipantRole': 'CUSTOMER',
                'Threshold': 10000
            }
        },
        {
            'NonTalkTimeFilter': {
                'Negate': False,
                'RelativeTimeRange': {
                    'EndPercentage': 80,
                    'StartPercentage': 10
                },
                'Threshold': 20000
            }
        },
        {
            'SentimentFilter': {
                'ParticipantRole': 'AGENT',
                'Sentiments': [
                    'NEGATIVE'
                ]
            }
        },
        {
            'TranscriptFilter': {
                'Negate': True,
                'AbsoluteTimeRange': {
                    'First': 10000
                },
                'Targets': [
                    'welcome',
                    'hello'
                ],
                'TranscriptFilterType': 'EXACT'
            }
        }
    ]
```

```
)

result = transcribe.get_call_analytics_category(CategoryName = category_name)
print(result)
```

通话后分析类别的规则标准

本节概述了您可以使用 CreateCallAnalyticsCategory API 操作创建的自定义 POST_CALL 规则的类型。

中断匹配

使用中断（InterruptionFilter 数据类型）的规则旨在匹配：

- 座席打断客户的情况
- 客户打断座席的情况
- 任何参与者打断对方的情况
- 没有中断

以下是 InterruptionFilter 可用参数的示例：

```
"InterruptionFilter": {
    "AbsoluteTimeRange": {
       Specify the time frame, in milliseconds, when the match should occur
    },
    "RelativeTimeRange": {
       Specify the time frame, in percentage, when the match should occur
    },
    "Negate": Specify if you want to match the presence or absence of interruptions,
    "ParticipantRole": Specify if you want to match speech from the agent, the
 customer, or both,
    "Threshold": Specify a threshold for the amount of time, in seconds, interruptions
 occurred during the call
},
```

有关这些参数以及与每个参数关联的有效值的更多信息，请参阅 CreateCallAnalyticsCategory 和 InterruptionFilter。

关键字匹配

使用关键字（TranscriptFilter 数据类型）的规则旨在匹配：

- 座席、客户或两者都说了的自定义单词或短语
- 座席、客户或两者都没说的自定义单词或短语
- 在特定时间范围内出现的自定义单词或短语

以下是 TranscriptFilter 可用参数的示例：

```
"TranscriptFilter": {
    "AbsoluteTimeRange": {
       Specify the time frame, in milliseconds, when the match should occur
    },
    "RelativeTimeRange": {
       Specify the time frame, in percentage, when the match should occur
    },
    "Negate": Specify if you want to match the presence or absence of your custom
 keywords,
    "ParticipantRole": Specify if you want to match speech from the agent, the
 customer, or both,
    "Targets": [ The custom words and phrases you want to match ],
    "TranscriptFilterType": Use this parameter to specify an exact match for the
 specified targets
}
```

有关这些参数以及与每个参数关联的有效值的更多信息，请参阅 CreateCallAnalyticsCategory 和 TranscriptFilter。

非通话时间匹配

使用非通话时间（NonTalkTimeFilter 数据类型）的规则旨在匹配：

- 在整个通话过程中，在指定时段存在沉默的情况
- 在整个通话过程中，在指定时段有语音的情况

以下是 NonTalkTimeFilter 可用参数的示例：

```
"NonTalkTimeFilter": {
    "AbsoluteTimeRange": {
```

```
 Specify the time frame, in milliseconds, when the match should occur
 },
    "RelativeTimeRange": {
 Specify the time frame, in percentage, when the match should occur
 },
    "Negate": Specify if you want to match the presence or absence of speech,
    "Threshold": Specify a threshold for the amount of time, in seconds, silence (or
 speech) occurred during the call
},
```

有关这些参数以及与每个参数关联的有效值的更多信息，请参阅 CreateCallAnalyticsCategory 和 NonTalkTimeFilter。

情绪匹配

使用情绪（SentimentFilter 数据类型）的规则旨在匹配：

- 客户、座席或两者在通话的指定时刻是否存在正面情绪的情况
- 客户、座席或两者在通话中的指定时刻是否存在负面情绪的情况
- 客户、座席或两者在通话的指定时刻是否存在中性情绪的情况
- 客户、座席或两者在通话的指定时刻是否正负面情绪兼有的情况

以下是 SentimentFilter 可用参数的示例：

```
"SentimentFilter": {
    "AbsoluteTimeRange": {
    Specify the time frame, in milliseconds, when the match should occur
    },
    "RelativeTimeRange": {
    Specify the time frame, in percentage, when the match should occur
    },
    "Negate": Specify if you want to match the presence or absence of your chosen
 sentiment,
    "ParticipantRole": Specify if you want to match speech from the agent, the
 customer, or both,
    "Sentiments": [ The sentiments you want to match ]
},
```

有关这些参数以及与每个参数关联的有效值的更多信息，请参阅 CreateCallAnalyticsCategory 和 SentimentFilter。

启动通话后分析转录

在开始通话后分析转录之前，您必须创建 Amazon Transcribe 要在音频中匹配的所有类别。

> **Note**
>
> 呼叫分析转录无法追溯性地与新类别匹配。只有您在启动呼叫分析转录之前创建的类别才能应用于该转录输出。

如果您创建了一个或多个类别，并且您的音频与至少一个类别中的所有规则匹配， Amazon Transcribe 会使用匹配的类别来标记您的输出。如果您选择不使用类别，或者您的音频与类别中指定的规则不匹配，则不会标记您的转录。

要启动通话后分析转录，您可以使用 AWS Management Console、AWS CLI 或 AWS SDK；有关示例，请参阅以下内容：

AWS Management Console

使用以下过程启动通话后分析作业。符合类别定义的所有特点的通话将使用该类别进行标记。

1. 在导航窗格中的 Amazon Transcribe 呼叫分析下，选择呼叫分析作业。
2. 请选择创建作业。

Configure job - *optional* Info

Content removal

Content removal conceals information in the resulting transcript from your source audio file. Amazon Transcribe changes items in the transcript and does not modify the source audio.

PII redaction Info
Label the type of PII and also mask the content with the PII entity type in the transcription output. For example, (123) 456-7890 will be masked as [PHONE].

Vocabulary filtering Info
Vocabulary filtering can remove, mask or tag specified words in the final transcript.

Customization

Custom vocabulary Info
A custom vocabulary improves the accuracy of recognizing words and phrases specific to your use case.

Summarization

Generative call summarization - *preview* Info
Generative call summarization provides a summary of the transcript, including important components of the conversation.

Categories

Create categories to classify calls. For example, you can create a category for all cancellation requests. When you run an analytics job, Amazon Transcribe applies that category to all calls that request cancellation.

Call analytics categories (2) Info

Search

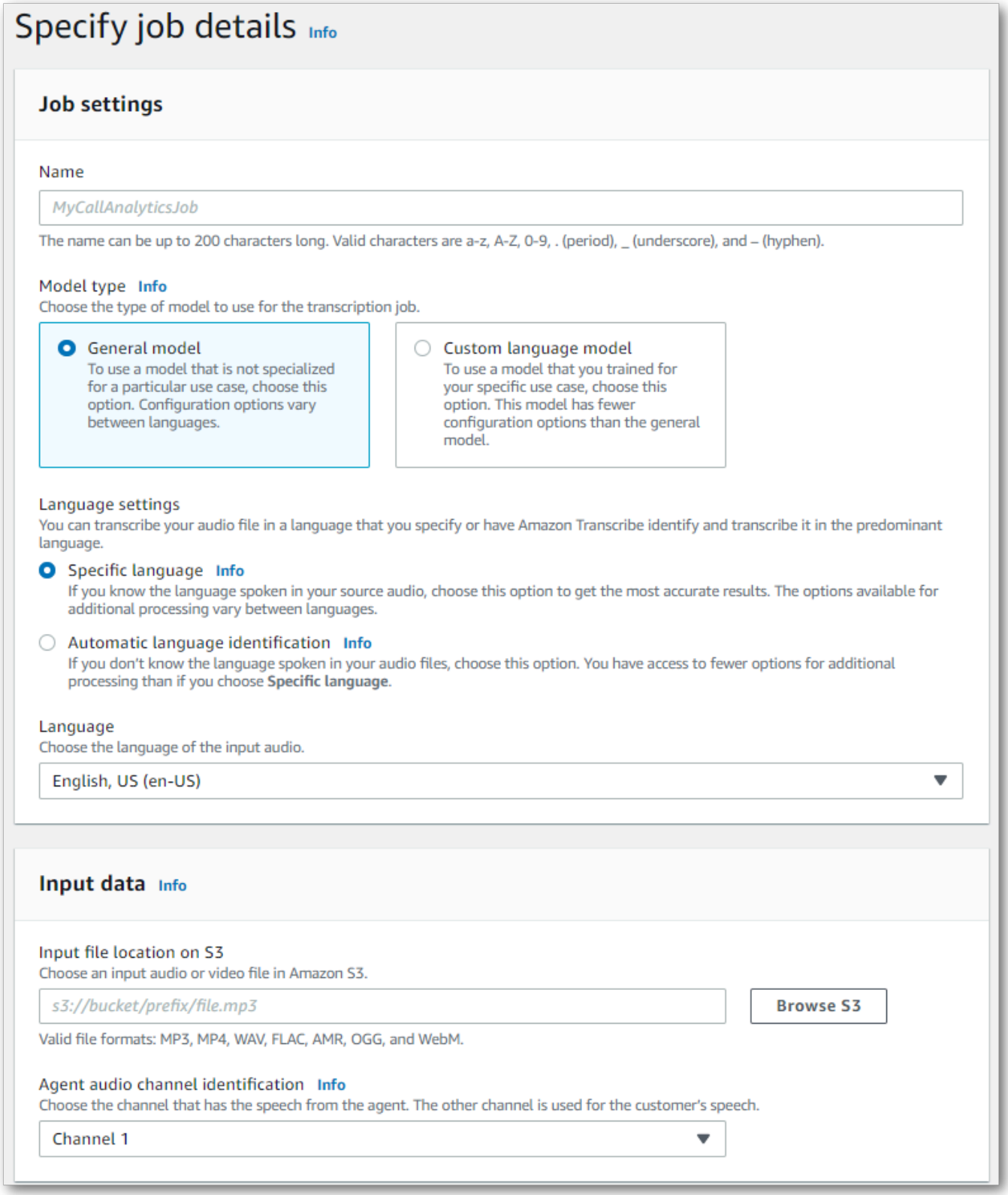
1

	Name	Type	Created	Modified
○	speak-over	POST_CALL	October 17 2023, 15:47 (UTC-07:00)	October 17 2023, 15:47 (UTC-07:00)
○	negative-ending	POST_CALL	October 17 2023, 15:46 (UTC-07:00)	October 17 2023, 15:46 (UTC-07:00)

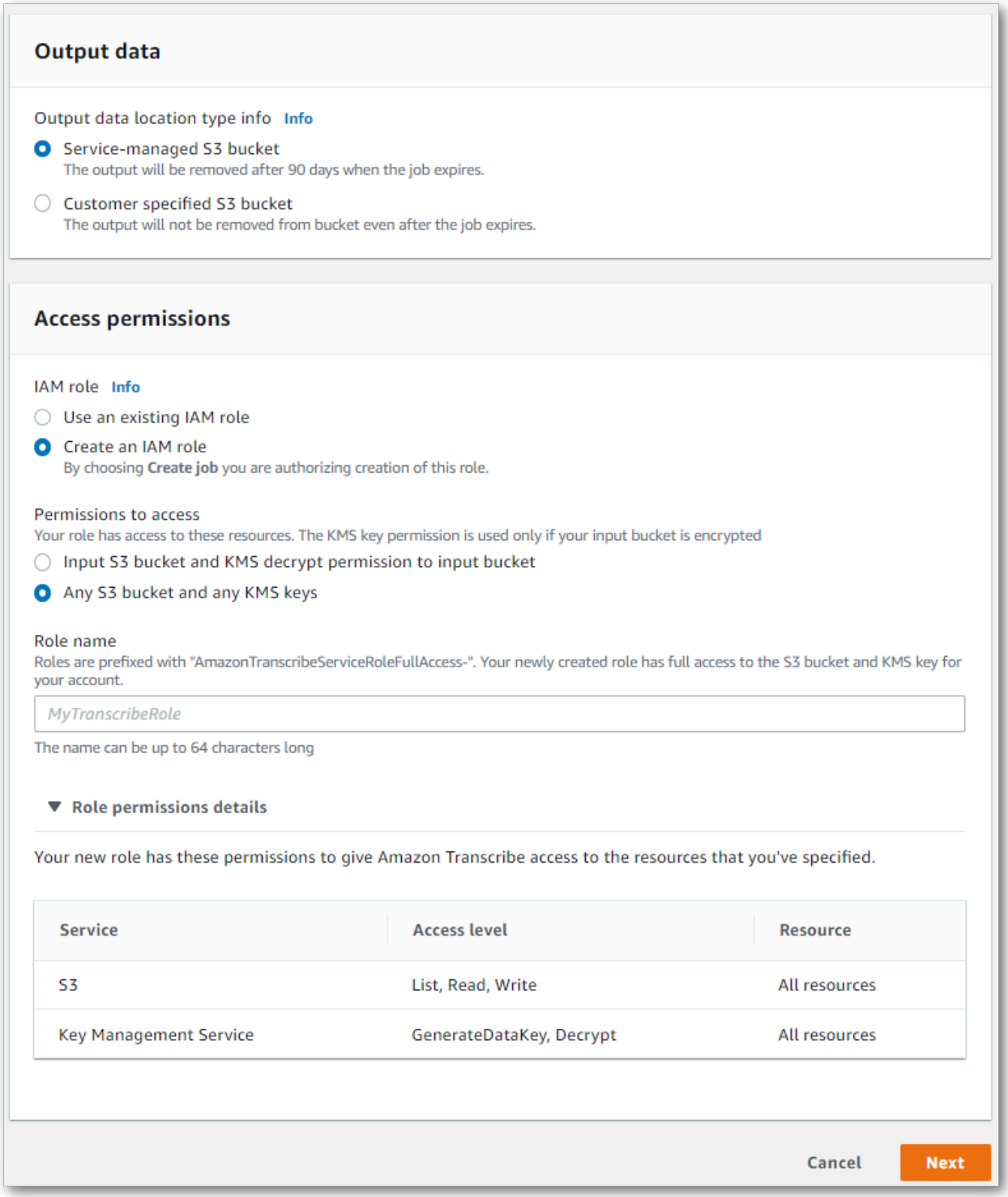
If the above categories aren't relevant to your use case, you can create a new category. Create a new category.

Cancel Previous Create job

3. 在指定作业详细信息页面上，提供有关您的呼叫分析作业的信息，包括输入数据的位置。



指定输出数据的所需 Amazon S3 位置以及要使用的 IAM 角色。



4. 选择下一步。
5. 对于配置作业，请打开要包含在呼叫分析作业中的所有可选特征。如果您之前创建了类别，则它们会显示在类别面板中，并自动应用于您的呼叫分析作业。

Configure job - *optional* Info

Content removal

Content removal conceals information in the resulting transcript from your source audio file. Amazon Transcribe changes items in the transcript and does not modify the source audio.

PII redaction Info
Label the type of PII and also mask the content with the PII entity type in the transcription output. For example, (123) 456-7890 will be masked as [PHONE].

Vocabulary filtering Info
Vocabulary filtering can remove, mask or tag specified words in the final transcript.

Customization

Custom vocabulary Info
A custom vocabulary improves the accuracy of recognizing words and phrases specific to your use case.

Summarization

Generative call summarization - *preview* Info
Generative call summarization provides a summary of the transcript, including important components of the conversation.

Categories

Create categories to classify calls. For example, you can create a category for all cancellation requests. When you run an analytics job, Amazon Transcribe applies that category to all calls that request cancellation.

Call analytics categories (2) Info

Search

1

	Name	Type	Created	Modified
○	speak-over	POST_CALL	October 17 2023, 15:47 (UTC-07:00)	October 17 2023, 15:47 (UTC-07:00)
○	negative-ending	POST_CALL	October 17 2023, 15:46 (UTC-07:00)	October 17 2023, 15:46 (UTC-07:00)

If the above categories aren't relevant to your use case, you can create a new category. Create a new category.

Cancel Previous Create job

6. 请选择创建作业。

AWS CLI

此示例使用start-call-analytics-job命令和channel-definitions参数。有关更多信息，请参阅 StartCallAnalyticsJob 和 ChannelDefinition。

```
aws transcribe start-call-analytics-job \
--region us-west-2 \
--call-analytics-job-name my-first-call-analytics-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-location s3://DOC-EXAMPLE-BUCKET/my-output-files/ \
--data-access-role-arn arn:aws:iam::111122223333:role/ExampleRole \
--channel-definitions ChannelId=0,ParticipantRole=AGENT
 ChannelId=1,ParticipantRole=CUSTOMER
```

以下是另一个使用start-call-analytics-job命令的示例，以及为该任务启用 Call Analytics 的请求正文。

```
aws transcribe start-call-analytics-job \
--region us-west-2 \
--cli-input-json file://filepath/my-call-analytics-job.json
```

my-call-analytics-job.json 文件包含以下请求正文。

```
{
      "CallAnalyticsJobName": "my-first-call-analytics-job",
      "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole",
      "Media": {
          "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
      },
      "OutputLocation": "s3://DOC-EXAMPLE-BUCKET/my-output-files/",
      "ChannelDefinitions": [
          {
              "ChannelId": 0,
              "ParticipantRole": "AGENT"
          },
          {
              "ChannelId": 1,
              "ParticipantRole": "CUSTOMER"
          }
      ]
```

```
}
```

AWS SDK for Python (Boto3)

此示例使用 start_call_analytics _job 方法启动呼叫分析作业。 AWS SDK for Python (Boto3) 有关更多信息，请参阅 StartCallAnalyticsJob 和 ChannelDefinition。

有关使用 AWS 软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本章。使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-call-analytics-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
output_location = "s3://DOC-EXAMPLE-BUCKET/my-output-files/"
data_access_role = "arn:aws:iam::111122223333:role/ExampleRole"
transcribe.start_call_analytics_job(
     CallAnalyticsJobName = job_name,
     Media = {
        'MediaFileUri': job_uri
     },
     DataAccessRoleArn = data_access_role,
     OutputLocation = output_location,
     ChannelDefinitions = [
        {
            'ChannelId': 0,
            'ParticipantRole': 'AGENT'
        },
        {
            'ChannelId': 1,
            'ParticipantRole': 'CUSTOMER'
        }
     ]
)

 while True:
   status = transcribe.get_call_analytics_job(CallAnalyticsJobName = job_name)
   if status['CallAnalyticsJob']['CallAnalyticsJobStatus'] in ['COMPLETED', 'FAILED']:
     break
   print("Not ready yet...")
   time.sleep(5)
```

```
print(status)
```

通话后分析输出

通话后分析记录按细分的 turn-by-turn 格式显示。它们包括通话分类、通话特点（音量分数、中断、非通话时间、通话速度）、通话摘要（问题、结果和操作项目）、编辑和情绪。此外，转录末尾还提供了对话特点摘要。

为了提高准确性并根据您的用例进一步自定义您的转录（例如包括特定行业的术语），请在您的呼叫分析请求中添加自定义词汇表或自定义语言模型。要屏蔽、删除或标记不希望出现在转录结果中的单词（例如脏话），请添加词汇表过滤。如果您不确定要传递给媒体文件的语言代码，您可以启用批量语言识别以自动识别媒体文件中的语言。

以下各节显示了见解级的 JSON 输出示例。有关编译后的输出，请参见已编译的通话后分析输出。

通话分类

以下是转录输出中的类别匹配介绍。此示例显示从 40040 毫秒时间戳到 42460 毫秒时间戳的音频与“正面解决”类别相匹配。在这种情况下，自定义的“正面解决”类别需要在语音最后几秒钟有正面的情绪。

```
"Categories": {
    "MatchedDetails": {
        "positive-resolution": {
            "PointsOfInterest": [
                {
                    "BeginOffsetMillis":  40040,
                    "EndOffsetMillis":  42460
                }
            ]
        }
    },
    "MatchedCategories": [
        " positive-resolution"
    ]
},
```

通话特点

以下是转录输出中的通话特点。请注意，针对每个对话轮次都提供了音量分数，而所有其它特点则在转录的末尾提供。

```
"LoudnessScores": [
    87.54,
    88.74,
    90.16,
    86.36,
    85.56,
    85.52,
    81.79,
    87.74,
    89.82
],

...

"ConversationCharacteristics": {
    "NonTalkTime": {
        "Instances": [],
        "TotalTimeMillis": 0
    },
    "Interruptions": {
        "TotalCount": 2,
        "TotalTimeMillis": 10700,
        "InterruptionsByInterrupter": {
            "AGENT": [
                {
                    "BeginOffsetMillis": 26040,
                    "DurationMillis": 5510,
                    "EndOffsetMillis": 31550
                }
            ],
            "CUSTOMER": [
                {
                    "BeginOffsetMillis": 770,
                    "DurationMillis": 5190,
                    "EndOffsetMillis": 5960
                }
            ]
        }
    },
    "TotalConversationDurationMillis": 42460,

    ...
```

```
    "TalkSpeed": {
        "DetailsByParticipant": {
            "AGENT": {
                "AverageWordsPerMinute": 150
            },
            "CUSTOMER": {
                "AverageWordsPerMinute": 167
            }
        }
    },
    "TalkTime": {
        "DetailsByParticipant": {
            "AGENT": {
                "TotalTimeMillis": 32750
            },
            "CUSTOMER": {
                "TotalTimeMillis": 18010
            }
        },
        "TotalTimeMillis": 50760
    }
},
```

问题、操作项目和后续步骤

- 在以下示例中，问题被确定为从字符 7 开始，到字符 51 结束，这指的是文本的这一部分：“我想取消我的食谱订阅”。

```
"Content": "Well, I would like to cancel my recipe subscription.",

"IssuesDetected": [
    {
        "CharacterOffsets": {
            "Begin": 7,
            "End": 51
        }
    }
],
```

- 在以下示例中，结果被确定为从字符 12 开始，到字符 78 结束，这指的是文本的这一部分：“我对您的账户进行了所有更改，现在已应用此折扣”。

```
"Content": "Wonderful. I made all changes to your account and now this discount is
 applied, please check.",

"OutcomesDetected": [
    {
        "CharacterOffsets": {
            "Begin": 12,
            "End": 78
        }
    }
],
```

- 在以下示例中，操作项被确定为从字符 0 开始，到字符 103 结束，这指的是文本的这一部分：“我今天将向您发送一封包含所有详细信息的电子邮件，下周我会给您回电进行跟进”。

```
"Content": "I will send an email with all the details to you today, and I will call
 you back next week to follow up. Have a wonderful evening.",

"ActionItemsDetected": [
    {
        "CharacterOffsets": {
            "Begin": 0,
            "End": 103
        }
    }
],
```

生成式通话摘要

以下是转录输出中的生成式通话摘要：

```
"ContactSummary": {
    "AutoGenerated": {
        "OverallSummary": {
            "Content": "A customer wanted to check to see if we had a bag allowance. We
 told them that we didn't have it, but we could add the bag from Canada to Calgary and
 then do the one coming back as well."
        }
    }
}
```

情绪分析

以下是转录输出中的情绪分析。

- 定性 turn-by-turn 情绪值：

```
"Content": "That's very sad to hear. Can I offer you a 50% discount to have you stay
 with us?",

...

"BeginOffsetMillis": 12180,
"EndOffsetMillis": 16960,
"Sentiment": "NEGATIVE",
"ParticipantRole": "AGENT"

...

"Content": "That is a very generous offer. And I accept.",

...

"BeginOffsetMillis": 17140,
"EndOffsetMillis": 19860,
"Sentiment": "POSITIVE",
"ParticipantRole": "CUSTOMER"
```

- 整个通话的定量情绪值：

```
"Sentiment": {
    "OverallSentiment": {
        "AGENT": 2.5,
        "CUSTOMER": 2.1
    },
```

- 每位参与者和每个通话四等份之一的定量情绪值：

```
"SentimentByPeriod": {
    "QUARTER": {
        "AGENT": [
            {
                "Score": 0.0,
                "BeginOffsetMillis": 0,
```

```
                "EndOffsetMillis": 9862
            },
            {
                "Score": -5.0,
                "BeginOffsetMillis": 9862,
                "EndOffsetMillis": 19725
            },
            {
                "Score": 5.0,
                "BeginOffsetMillis": 19725,
                "EndOffsetMillis": 29587
            },
            {
                "Score": 5.0,
                "BeginOffsetMillis": 29587,
                "EndOffsetMillis": 39450
            }
        ],
        "CUSTOMER": [
            {
                "Score": -2.5,
                "BeginOffsetMillis": 0,
                "EndOffsetMillis": 10615
            },
            {
                "Score": 5.0,
                "BeginOffsetMillis": 10615,
                "EndOffsetMillis": 21230
            },
            {
                "Score": 2.5,
                "BeginOffsetMillis": 21230,
                "EndOffsetMillis": 31845
            },
            {
                "Score": 5.0,
                "BeginOffsetMillis": 31845,
                "EndOffsetMillis": 42460
            }
        ]
    }
}
```

PII 编辑

以下是转录输出中的 PII 编辑。

```
"Content": "[PII], my name is [PII], how can I help?",
"Redaction": [{
    "Confidence": "0.9998",
    "Type": "NAME",
    "Category": "PII"
}]
```

有关更多信息，请参阅在批处理作业中编辑 PII。

语言识别

以下是转录输出中的语言识别（如果启用了该功能）。

```
"LanguageIdentification": [{
  "Code": "en-US",
  "Score": "0.8299"
}, {
  "Code": "en-NZ",
  "Score": "0.0728"
}, {
  "Code": "zh-TW",
  "Score": "0.0695"
}, {
  "Code": "th-TH",
  "Score": "0.0156"
}, {
  "Code": "en-ZA",
  "Score": "0.0121"
}]
```

在上面的输出示例中，语言识别使用置信度分数填充语言代码。将选择分数最高的结果以作为转录的语言代码。有关模式详细信息，请参阅识别媒体中的主要语言。

已编译的通话后分析输出

为简洁起见，在以下转录输出中，某些内容被替换为省略号。

```
{
    "JobStatus": "COMPLETED",
```

```
    "LanguageCode": "en-US",
    "Transcript": [
        {
            "LoudnessScores": [
                78.63,
                78.37,
                77.98,
                74.18
            ],
            "Content": "[PII], my name is [PII], how can I help?",

            ...

             "Content": "Well, I would like to cancel my recipe subscription.",
             "IssuesDetected": [
                 {
                     "CharacterOffsets": {
                         "Begin": 7,
                         "End": 51
                     }
                 }
             ],

            ...

            "Content": "That's very sad to hear. Can I offer you a 50% discount to have
you stay with us?",
            "Items": [
            ...
             ],
            "Id": "649afe93-1e59-4ae9-a3ba-a0a613868f5d",
            "BeginOffsetMillis": 12180,
            "EndOffsetMillis": 16960,
            "Sentiment": "NEGATIVE",
            "ParticipantRole": "AGENT"
        },
        {
            "LoudnessScores": [
                    80.22,
                    79.48,
                    82.81
            ],
            "Content": "That is a very generous offer. And I accept.",
            "Items": [
```

```
            ...
            ],
            "Id": "f9266cba-34df-4ca8-9cea-4f62a52a7981",
            "BeginOffsetMillis": 17140,
            "EndOffsetMillis": 19860,
            "Sentiment": "POSITIVE",
            "ParticipantRole": "CUSTOMER"
        },
        {

    ...

            "Content": "Wonderful. I made all changes to your account and now this
discount is applied, please check.",
            "OutcomesDetected": [
                {
                    "CharacterOffsets": {
                        "Begin": 12,
                        "End": 78
                    }
                }
            ],

            ...

            "Content": "I will send an email with all the details to you today, and I
will call you back next week to follow up. Have a wonderful evening.",
            "Items": [
            ...
            ],
            "Id": "78cd0923-cafd-44a5-a66e-09515796572f",
            "BeginOffsetMillis": 31800,
            "EndOffsetMillis": 39450,
            "Sentiment": "POSITIVE",
            "ParticipantRole": "AGENT"
        },
        {
           "LoudnessScores": [
               78.54,
               68.76,
               67.76
           ],
           "Content": "Thank you very much, sir. Goodbye.",
           "Items": [
```

```
            ...
            ],
            "Id": "5c5e6be0-8349-4767-8447-986f995af7c3",
            "BeginOffsetMillis": 40040,
            "EndOffsetMillis": 42460,
            "Sentiment": "POSITIVE",
            "ParticipantRole": "CUSTOMER"
        }
    ],

    ...

    "Categories": {
         "MatchedDetails": {
             "positive-resolution": {
                 "PointsOfInterest": [
                     {
                         "BeginOffsetMillis": 40040,
                         "EndOffsetMillis": 42460
                     }
                 ]
             }
         },
         "MatchedCategories": [
             "positive-resolution"
         ]
     },

     ...

     "ConversationCharacteristics": {
         "NonTalkTime": {
             "Instances": [],
             "TotalTimeMillis": 0
         },
         "Interruptions": {
             "TotalCount": 2,
             "TotalTimeMillis": 10700,
             "InterruptionsByInterrupter": {
                 "AGENT": [
                     {
                         "BeginOffsetMillis": 26040,
                         "DurationMillis": 5510,
                         "EndOffsetMillis": 31550
```

```
                }
            ],
            "CUSTOMER": [
                {
                    "BeginOffsetMillis": 770,
                    "DurationMillis": 5190,
                    "EndOffsetMillis": 5960
                }
            ]
        }
    },
    "TotalConversationDurationMillis": 42460,
    "Sentiment": {
        "OverallSentiment": {
            "AGENT": 2.5,
            "CUSTOMER": 2.1
        },
        "SentimentByPeriod": {
            "QUARTER": {
                "AGENT": [
                    {
                        "Score": 0.0,
                        "BeginOffsetMillis": 0,
                        "EndOffsetMillis": 9862
                    },
                    {
                        "Score": -5.0,
                        "BeginOffsetMillis": 9862,
                        "EndOffsetMillis": 19725
                    },
                    {
                        "Score": 5.0,
                        "BeginOffsetMillis": 19725,
                        "EndOffsetMillis": 29587
                    },
                    {
                        "Score": 5.0,
                        "BeginOffsetMillis": 29587,
                        "EndOffsetMillis": 39450
                    }
                ],
                "CUSTOMER": [
                    {
                        "Score": -2.5,
```

```
                        "BeginOffsetMillis": 0,
                        "EndOffsetMillis": 10615
                    },
                    {
                        "Score": 5.0,
                        "BeginOffsetMillis": 10615,
                        "EndOffsetMillis": 21230
                    },
                    {
                        "Score": 2.5,
                        "BeginOffsetMillis": 21230,
                        "EndOffsetMillis": 31845
                    },
                    {
                        "Score": 5.0,
                        "BeginOffsetMillis": 31845,
                        "EndOffsetMillis": 42460
                    }
                ]
            }
        }
    },
    "TalkSpeed": {
        "DetailsByParticipant": {
            "AGENT": {
                "AverageWordsPerMinute": 150
            },
            "CUSTOMER": {
                "AverageWordsPerMinute": 167
            }
        }
    },
    "TalkTime": {
        "DetailsByParticipant": {
            "AGENT": {
                "TotalTimeMillis": 32750
            },
            "CUSTOMER": {
                "TotalTimeMillis": 18010
            }
        },
        "TotalTimeMillis": 50760
    }
```

```
    },
```

启用生成式通话摘要

> **Note**
>
> 生成式通话摘要功能包含在 Transcribe 通话分析 - 通话后分析预览版中，并且可能会发生变化。该功能作为 AWS 服务条款中定义的预览服务提供。

> **Note**
>
> 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于生成式人工智能支持的通话后摘要是基于 Amazon Bedrock 构建的，因此，用户可以充分利用 Amazon Bedrock 中实施的控制措施以安全且负责任地使用人工智能 (AI)。

要在通话后分析作业中使用生成式通话摘要，请参阅以下示例：

AWS Management Console

在“摘要”面板中，启用生成式通话摘要以在输出中收到摘要。

Configure job - *optional* Info

Content removal

Content removal conceals information in the resulting transcript from your source audio file. Amazon Transcribe changes items in the transcript and does not modify the source audio.

PII redaction Info
Label the type of PII and also mask the content with the PII entity type in the transcription output. For example, (123) 456-7890 will be masked as [PHONE].

Vocabulary filtering Info
Vocabulary filtering can remove, mask or tag specified words in the final transcript.

Customization

Custom vocabulary Info
A custom vocabulary improves the accuracy of recognizing words and phrases specific to your use case.

Summarization

Generative call summarization - *preview* Info
Generative call summarization provides a summary of the transcript, including important components of the conversation.

Categories

Create categories to classify calls. For example, you can create a category for all cancellation requests. When you run an analytics job, Amazon Transcribe applies that category to all calls that request cancellation.

Call analytics categories (2) Info

Search

1

	Name	Type	Created	Modified
○	speak-over	POST_CALL	October 17 2023, 15:47 (UTC-07:00)	October 17 2023, 15:47 (UTC-07:00)
○	negative-ending	POST_CALL	October 17 2023, 15:46 (UTC-07:00)	October 17 2023, 15:46 (UTC-07:00)

If the above categories aren't relevant to your use case, you can create a new category. Create a new category.

AWS CLI

此示例使用带有Summarization子Settings参数的start-call-analytics-job命令和参数。有关更多信息，请参阅 StartCallAnalyticsJob。

```
aws transcribe start-call-analytics-job \
--region us-west-2 \
--call-analytics-job-name my-first-call-analytics-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-location s3://DOC-EXAMPLE-BUCKET/my-output-files/ \
--data-access-role-arn arn:aws:iam::111122223333:role/ExampleRole \
--channel-definitions ChannelId=0,ParticipantRole=AGENT
 ChannelId=1,ParticipantRole=CUSTOMER
--settings '{"Summarization":{"GenerateAbstractiveSummary":true}}'
```

以下是另一个使用start-call-analytics-job命令的示例，以及支持该任务汇总的请求正文。

```
aws transcribe start-call-analytics-job \
--region us-west-2 \
--cli-input-json file://filepath/my-call-analytics-job.json
```

my-call-analytics-job.json 文件包含以下请求正文。

```
{
  "CallAnalyticsJobName": "my-first-call-analytics-job",
  "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
  },
  "OutputLocation": "s3://DOC-EXAMPLE-BUCKET/my-output-files/",
  "ChannelDefinitions": [
    {
      "ChannelId": 0,
      "ParticipantRole": "AGENT"
    },
    {
      "ChannelId": 1,
      "ParticipantRole": "CUSTOMER"
```

```
    }
  ],
  "Settings": {
    "Summarization":{
      "GenerateAbstractiveSummary": true
    }
  }
}
```

AWS SDK for Python (Boto3)

此示例使用 start_call_analytics _job 方法启动启用汇总功能的呼叫分析。 AWS SDK for Python (Boto3) 有关更多信息，请参阅 StartCallAnalyticsJob。

有关使用 AWS 软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本章。使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS

```
from __future__ import print_function
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-call-analytics-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
output_location = "s3://DOC-EXAMPLE-BUCKET/my-output-files/"
data_access_role = "arn:aws:iam::111122223333:role/ExampleRole"
transcribe.start_call_analytics_job(
  CallAnalyticsJobName = job_name,
  Media = {
    'MediaFileUri': job_uri
  },
  DataAccessRoleArn = data_access_role,
  OutputLocation = output_location,
  ChannelDefinitions = [
    {
      'ChannelId': 0,
      'ParticipantRole': 'AGENT'
    },
    {
      'ChannelId': 1,
      'ParticipantRole': 'CUSTOMER'
    }
```

```
  ],
  Settings = {
    "Summarization":
      {
        "GenerateAbstractiveSummary": true
      }
  }
)

while True:
  status = transcribe.get_call_analytics_job(CallAnalyticsJobName = job_name)
  if status['CallAnalyticsJob']['CallAnalyticsJobStatus'] in ['COMPLETED', 'FAILED']:
    break
  print("Not ready yet...")
  time.sleep(5)
print(status)
```

实时通话分析

实时呼叫分析提供实时见解，可用于解决问题和缓解问题升级。

实时呼叫分析可获得以下见解：

- 类别事件，使用规则标记特定关键字和短语；类别事件可用于创建实时警报
- 问题检测，可识别每个音频片段中出现的问题
- 文本转录中的 PII（敏感数据）识别
- 对您的文本转录进行的 PII（敏感数据）编辑
- 每个语音片段的情绪分析

除了实时通话分析外，Amazon Transcribe 还可以对您的媒体流进行通话后分析。您可以使用 `PostCallAnalyticsSettings` 参数在实时呼叫分析请求中包含通话后分析。

实时见解

本节详细介绍了可用于实时呼叫分析转录的见解。

类别事件

使用类别事件，您可以根据确切的关键字或短语来匹配您的转录。例如，如果您为“我想和经理说话”这句话设置了过滤 Amazon Transcribe 器，则会筛选出该短语的确切内容。

以下是一个输出示例。

有关创建实时呼叫分析类别的更多信息，请参阅为实时转录创建类别。

> **Tip**
>
> 类别事件允许您设置实时警报；有关更多信息，请参阅为类别匹配创建实时警报。

问题检测

问题检测提供了每个音频片段中检测到的问题的简要摘要。使用问题检测特征，您可以：

- 减少在通话期间和通话后手动做笔记的需求
- 提高座席效率，使他们能够更快地响应客户

> **Note**
>
> 问题检测支持以下地方的英语方言：澳大利亚 (en-AU)、英国 (en-GB) 和美国 (en-US)。

问题检测特征适用于所有行业和业务领域，并且基于上下文。它可以正常工作 out-of-the-box ，因此不支持自定义，例如模型训练或自定义类别。

通过实时呼叫分析对每个完整的音频片段进行问题检测。

以下是一个输出示例。

PII（敏感数据）识别

敏感数据识别会在文本转录中标记个人身份信息 (PII)。此参数对于保护客户信息很有用。

> **Note**
>
> PII 识别支持以下地方的英语方言：澳大利亚 (en-AU)、英国 (en-GB) 和美国 (en-US)。

通过实时呼叫分析对每个完整的音频片段进行 PII 识别。

要查看使用此功能识别的 PII 列表，或要了解有关使用识别个人身份的更多信息 Amazon Transcribe，请参阅。个人身份信息个人身份信息个人身份信息个人身份信息个人身份信息

以下是一个输出示例。

PII（敏感数据）编辑

敏感数据编辑将文本转录中的个人身份信息 (PII) 替换为 PII 的类型（例如 [NAME]）。此参数对于保护客户信息很有用。

> **Note**
>
> PII 编辑支持以下地方的英语方言：澳大利亚 (en-AU)、英国 (en-GB) 和美国 (en-US)。

通过实时呼叫分析对每个完整的音频片段进行 PII 编辑。

要查看使用此特征进行了编辑的 PII 列表，或要了解有关使用 Amazon Transcribe进行编辑的更多信息，请参阅 个人身份信息个人身份信息个人身份信息个人身份信息个人身份信息。

以下是一个输出示例。

情绪分析

情绪分析可以评估客户和座席在整个通话过程中的感受。此指标为每个语音片段提供，并以定性值（positive、neutral、mixed、或 negative）表示。

使用此参数，您可以定性地评估每个通话参与者的总体情绪以及每个参与者在每个语音片段中的情绪。该指标可以帮助确定您的座席是否能够在通话结束时让心烦意乱的客户感到满意。

通过实时呼叫分析对每个完整的音频片段进行情绪分析。

情感分析有效 out-of-the-box ，因此不支持自定义，例如模型训练或自定义类别。

以下是一个输出示例。

为实时转录创建类别

实时呼叫分析支持创建自定义类别，使您能够量身定制转录分析，以便于极大程度地满足您的特定业务需求。

您可以根据需要创建任意数量的类别，以囊括一系列不同的场景。对于您创建的每个类别，必须创建 1 到 20 条规则。实时呼叫分析转录仅支持使用 TranscriptFilter（关键字匹配）的规则。有关在 CreateCallAnalyticsCategory 操作中使用规则的更多详细信息，请参阅实时呼叫分析类别的规则标准一节。

如果您的媒体内容符合您在给定类别中指定的所有规则，则 Amazon Transcribe 会使用该类别标记您的输出。有关采用 JSON 输出格式的类别匹配示例，请参阅类别事件输出。

以下是一些关于如何使用自定义类别的示例：

- 通过标记和跟踪特定的一组关键字，找出需要立即关注的问题
- 监控合规性，例如座席说出（或省略）特定短语
- 实时标记特定的单词和短语；然后，您可以设置类别匹配以设置即时警报。例如，如果您为提出“要与经理交谈”的客户创建了实时呼叫分析类别，则可以为该实时类别匹配设置事件警报，通知值班经理。

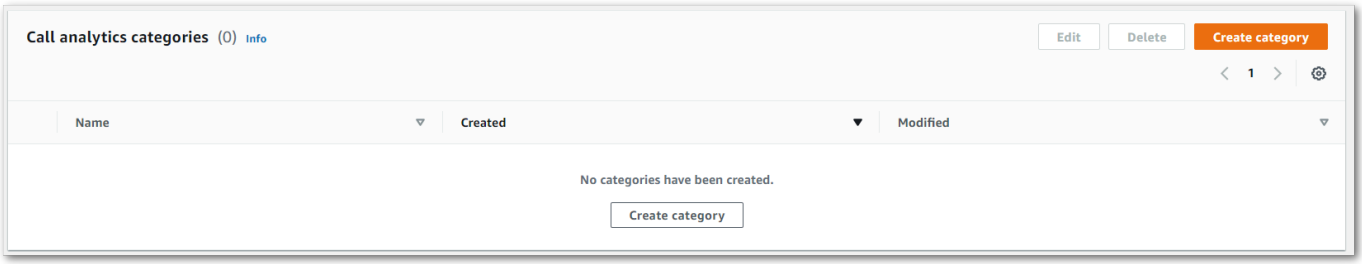
通话后类别与实时类别

创建新类别时，您可以指定是要将其创建为通话后类别 (POST_CALL)，还是创建为实时类别 (REAL_TIME)。如果您没有指定选项，则默认情况下，您的类别将创建为通话后类别。实时类别匹配可用于创建实时警报。有关更多信息，请参阅 为类别匹配创建实时警报。

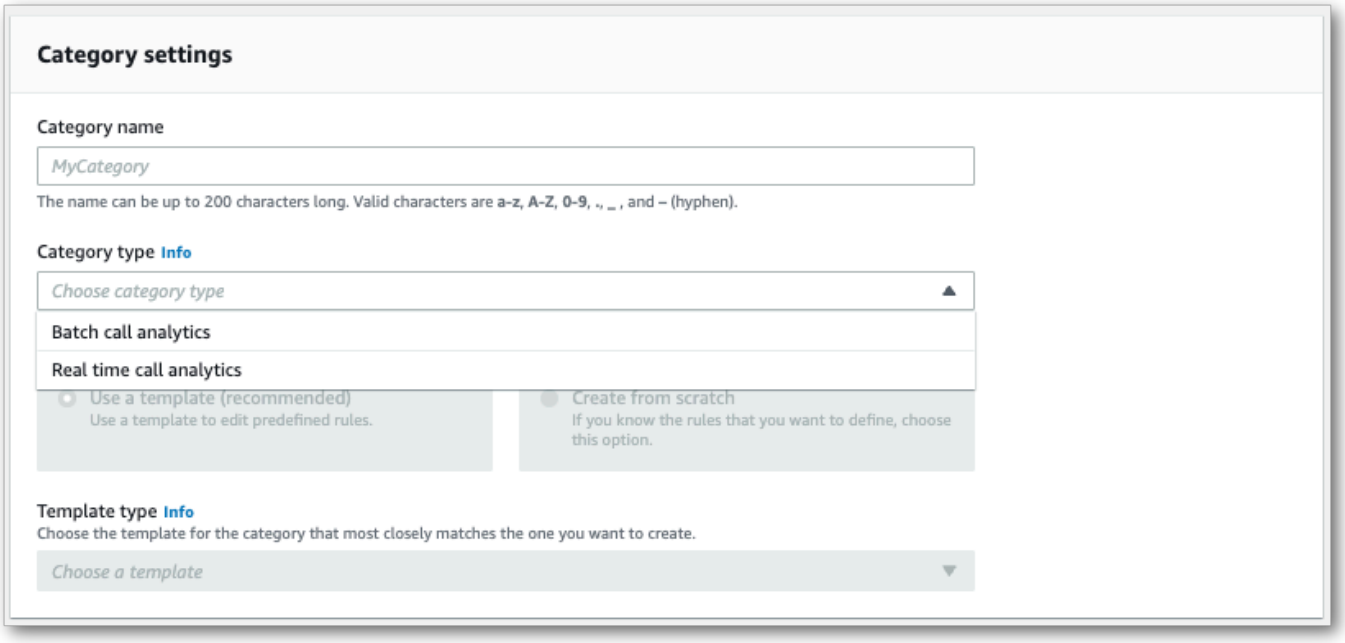
要为实时呼叫分析创建新类别，您可以使用 AWS Management Console、AWS CLI 或 AWS SDK；有关示例，请参阅以下内容：

AWS Management Console

1. 在导航窗格中 Amazon Transcribe，选择Amazon Transcribe 呼叫分析。
2. 选择呼叫分析类别，之后您将进入呼叫分析类别页面。选择创建类别按钮。

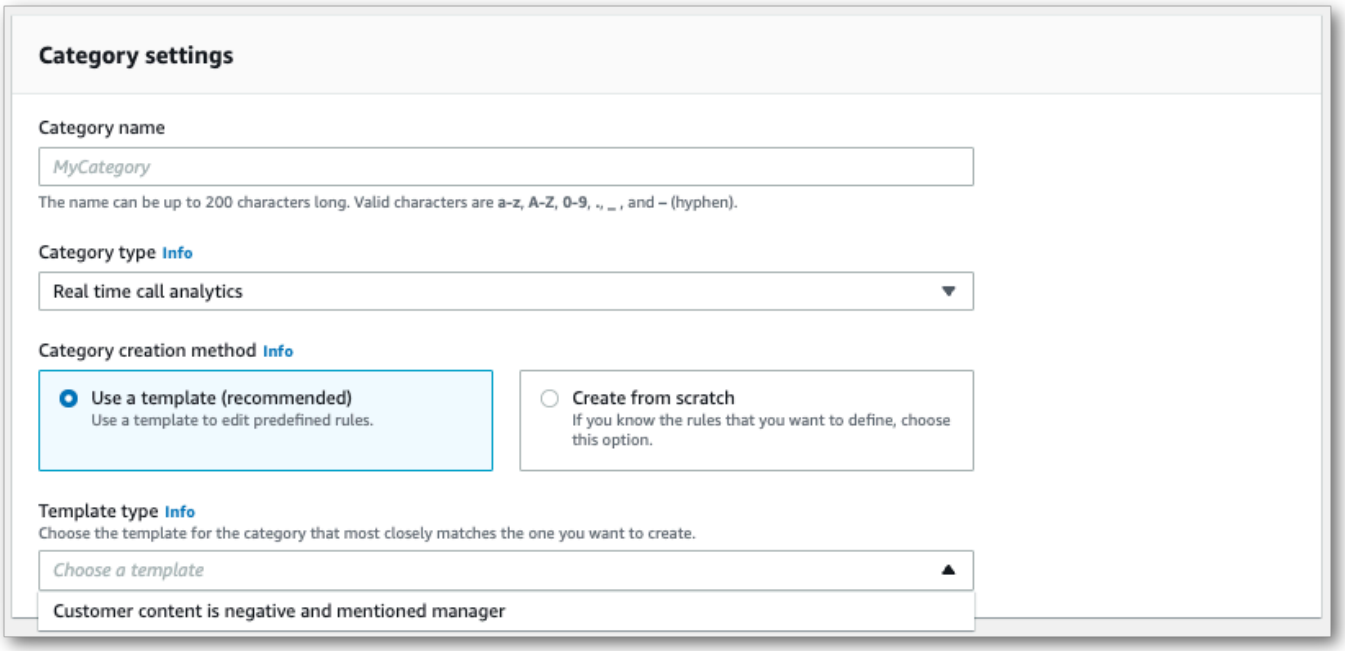


3. 您现在已进入创建类别页面。输入类别的名称，然后在类别类型下拉菜单中选择“实时呼叫分析”。

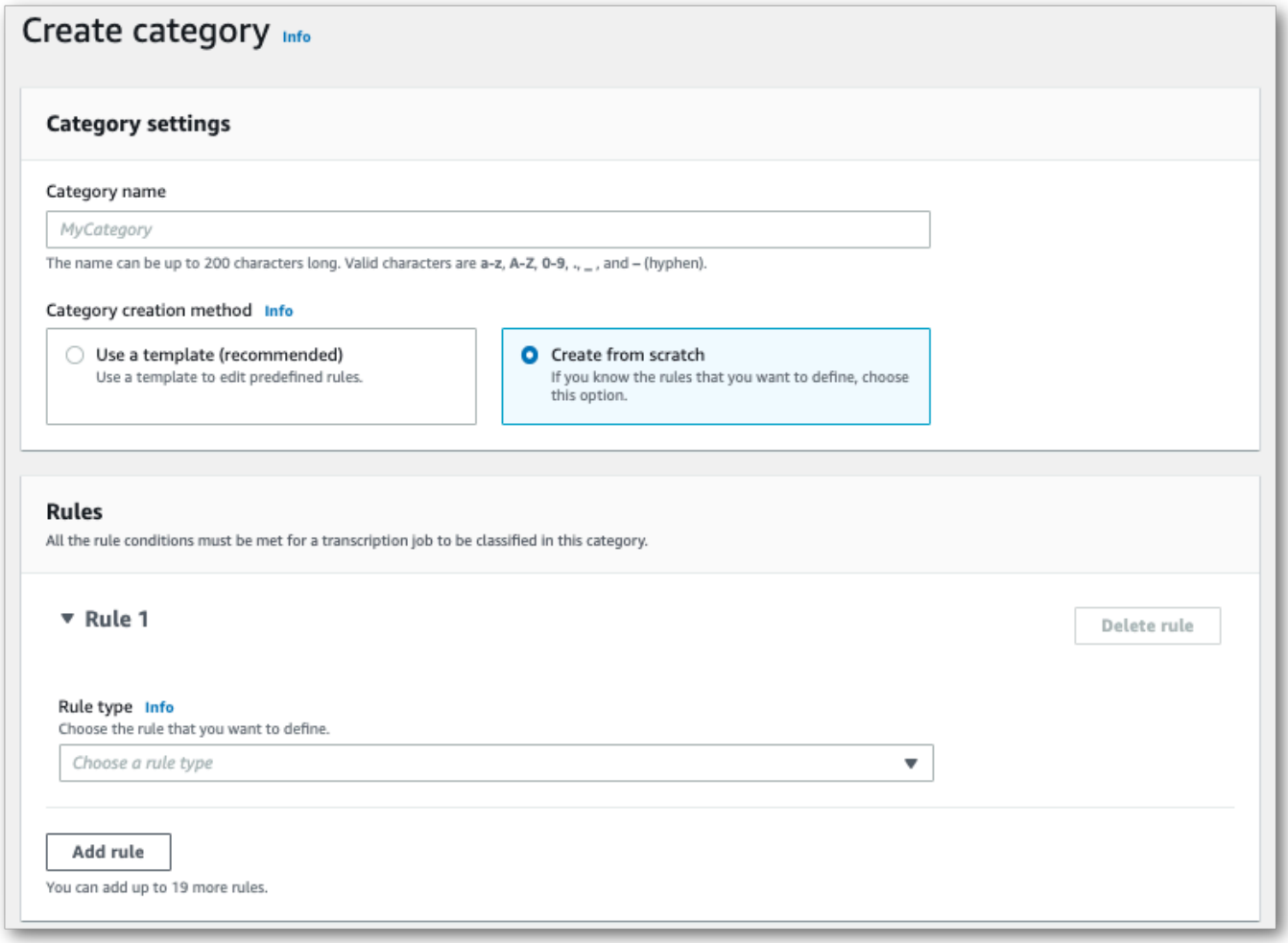


4. 您可以选择一个模板来创建您的类别，也可以从头开始制作一个模板。

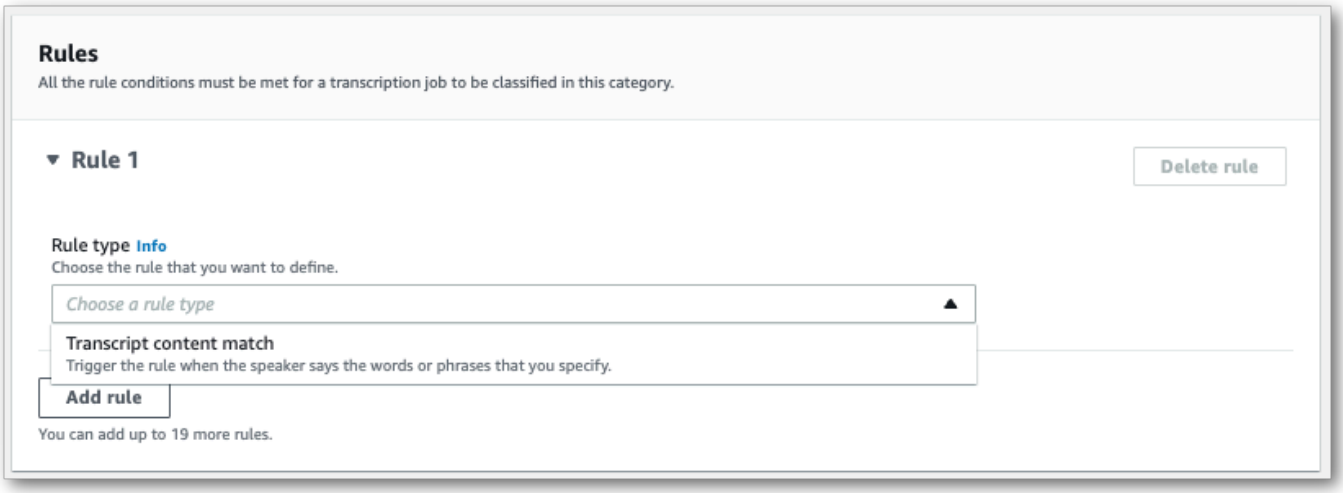
   如果使用模板：选择使用模板(建议)，选择所需的模板，然后选择创建类别。



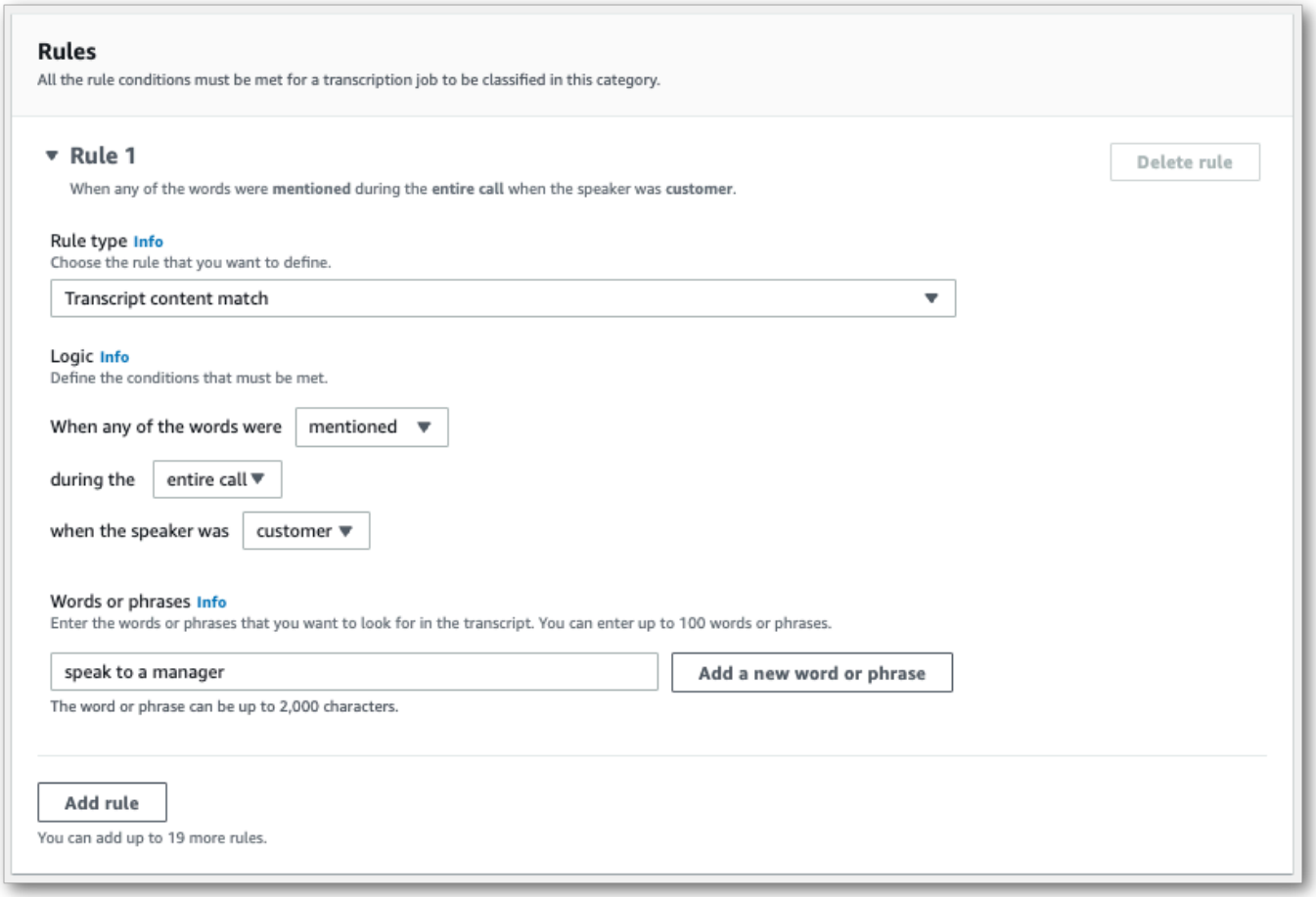
5. 如果要创建自定义类别：请选择从头开始创建。



6. 使用下拉菜单向您的类别添加规则。您最多可以为每个类别添加 20 条规则。对于实时呼叫分析转录，您只能包含涉及转录内容匹配的规则。任何匹配项都会实时标记。



7. 此处介绍了使用以下规则的类别示例：客户在通话中的任何时候提出“要与经理交谈”。



8. 向类别添加完规则后，选择创建类别。

AWS CLI

此示例使用create-call-analytics-category命令。有关更多信息，请参阅 CreateCallAnalyticsCategory、CategoryProperties 和 Rule。

下面的示例创建了使用以下规则的类别：

- 客户在通话中的任何时候提到“要与经理交谈”这句话。

此示例使用create-call-analytics-category命令和请求正文将规则添加到您的类别中。

```
aws transcribe create-call-analytics-category \
--cli-input-json file://filepath/my-first-analytics-category.json
```

my-first-analytics-category.json 文件包含以下请求正文。

```
{
  "CategoryName": "my-new-real-time-category",
  "InputType": "REAL_TIME",
  "Rules": [
        {
            "TranscriptFilter": {
                "Negate": false,
                "Targets": [
                    "speak to the manager"
                ],
                "TranscriptFilterType": "EXACT"
            }
        }
    ]
}
```

AWS SDK for Python (Boto3)

此示例使用 create_call_an alytics_category 方法的CategoryName和Rules参数创建类别。 AWS SDK for Python (Boto3) 有关更多信息，请参阅 CreateCallAnalyticsCategory、CategoryProperties 和 Rule。

有关使用 AWS 软件开发工具包的其他示例，包括特定功能、场景和跨服务示例，请参阅本章。使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS

下面的示例创建了使用以下规则的类别：

- 客户在通话中的任何时候提到“要与经理交谈”这句话。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
category_name = "my-new-real-time-category"
transcribe.create_call_analytics_category(
    CategoryName = category_name,
    InputType = "REAL_TIME",
    Rules = [
        {
            'TranscriptFilter': {
                'Negate': False,
```

```
                'Targets': [
                    'speak to the manager'
                ],
                'TranscriptFilterType': 'EXACT'
            }
        }
    ]
)

result = transcribe.get_call_analytics_category(CategoryName = category_name)
print(result)
```

实时呼叫分析类别的规则标准

本节概述了您可以使用 CreateCallAnalyticsCategory API 操作创建的自定义 REAL_TIME 规则的类型。

问题检测会自动进行，因此您无需创建任何规则或类别来标记问题。

请注意，实时呼叫分析转录仅支持关键字匹配。如果要创建包含中断、沉默或情绪的类别，请参阅通话后分析类别的规则标准。

关键字匹配

使用关键字（TranscriptFilter 数据类型）的规则旨在匹配：

- 座席、客户或两者都说了的自定义单词或短语
- 座席、客户或两者都没说的自定义单词或短语
- 在特定时间范围内出现的自定义单词或短语

以下是 TranscriptFilter 可用参数的示例：

```
"TranscriptFilter": {
    "AbsoluteTimeRange": {
       Specify the time frame, in milliseconds, when the match should occur
    },
    "RelativeTimeRange": {
       Specify the time frame, in percentage, when the match should occur
    },
    "Negate": Specify if you want to match the presence or absence of your custom
 keywords,
```

```
    "ParticipantRole": Specify if you want to match speech from the agent, the
 customer, or both,
    "Targets": [ The custom words and phrases you want to match ],
    "TranscriptFilterType": Use this parameter to specify an exact match for the
 specified targets
}
```

有关这些参数以及与每个参数关联的有效值的更多信息，请参阅 CreateCallAnalyticsCategory 和 TranscriptFilter。

通过实时转录进行通话后分析

通话后分析是一项可选特征，可用于实时呼叫分析转录。除了标准的实时分析见解外，通话后分析还为您提供以下信息：

- 操作项目：列出在通话中识别的所有操作项目。
- 中断：衡量一个参与者是否以及何时打断了另一个参与者的话
- 问题：提供通话中发现的问题
- 音量：衡量每位参与者的说话音量
- 非通话时间：衡量不含语音的时间段
- 结果：提供在通话中识别的结果或解决方案
- 通话速度：衡量两个参与者的说话速度
- 通话时间：衡量每个参与者在通话期间的说话时长，以毫秒为单位

启用后，来自音频流的呼叫后分析会生成类似于音频文件中的通话后分析的脚本，并将其存储在中指定的 Amazon S3 存储桶中。OutputLocation此外，通话后分析会记录您的音频流，并将其作为音频文件（WAV格式）保存在同一个 Amazon S3 存储桶中。如果您启用密文，则经过编辑的笔录和经过编辑的音频文件也会存储在指定的存储桶中。 Amazon S3 对音频流启用通话后分析会生成两到四个文件，如下所述：

- 如果未启用编辑功能，则您的输出文件为：
  1. 未经编辑的转录
  2. 未经编辑的音频文件
- 如果在没有未编辑选项 (redacted) 的情况下启用了编辑功能，则您的输出文件为：
  1. 经过编辑的转录
  2. 经过编辑的音频文件

- 如果在有未编辑选项 (redacted_and_unredacted) 的情况下启用了编辑功能，则您的输出文件为：
  1. 经过编辑的转录
  2. 经过编辑的音频文件
  3. 未经编辑的转录
  4. 未经编辑的音频文件

请注意，如果您在请求中启用了通话后分析 (PostCallAnalyticsSettings)，并且您使用的是 FLAC 或 OPUS-OGG 媒体，则不会在转录中体现 loudnessScore，也不会创建音频流的音频录音。

有关为音频流通话后分析提供的见解的更多信息，请参阅话通后分析见解部分。

> **Tip**
>
> 如果您在实时呼叫分析请求中启用通话后分析，则您的所有 POST_CALL 和 REAL-TIME 类别都将应用于您的通话后分析转录。

启用通话后分析

要启用通话后分析，您必须在实时呼叫分析请求中包含 PostCallAnalyticsSettings 参数。启用 PostCallAnalyticsSettings 时必须包括以下参数：

- OutputLocation: 您要 Amazon S3 存储通话后记录的存储桶。
- DataAccessRoleArn：有权访问指定 Amazon S3 存储桶的 Amazon S3 角色的 Amazon 资源名称 (ARN)。请注意，您还必须使用适用于实时分析的信任策略。

如果您想要一份经过编辑的转录，可以在请求中包含 ContentRedactionOutput 或 ContentRedactionType。有关这些参数的更多信息，请参阅 API 参考中的 StartCallAnalyticsStreamTranscription。

要在启用呼叫后分析的情况下开始实时呼叫分析转录，您可以使用 AWS Management Console（仅限演示）、HTTP/ 2 或。WebSockets有关示例，请参阅启动实时呼叫分析转录。

> **Important**
>
> 目前，AWS Management Console 唯一提供带有预加载音频示例的实时呼叫分析演示。如果您想使用自己的音频，则必须使用 API（HTTP/2 或 SDK）。 WebSockets

通话后分析输出示例

通话后记录按分段的 turn-by-turn 格式显示。它们包括通话特点、情绪、通话摘要、问题检测和（可选）PII 编辑。如果您的任何通话后类别与音频内容匹配，则这些类别也会出现在您的输出中。

为了提高准确性并根据您的用例进一步自定义您的转录（例如包括特定行业的术语），请在您的呼叫分析请求中添加自定义词汇表或自定义语言模型。要屏蔽、删除或标记不希望出现在转录结果中的单词（例如脏话），请添加词汇表过滤。

以下是编译后的通话后分析输出示例：

```
{
    "JobStatus": "COMPLETED",
    "LanguageCode": "en-US",
    "AccountId": "1234567890",
    "Channel": "VOICE",
    "Participants": [{
        "ParticipantRole": "AGENT"
    },
    {
        "ParticipantRole": "CUSTOMER"
    }],
    "SessionId": "12a3b45c-de6f-78g9-0123-45h6ab78c901",
    "ContentMetadata": {
        "Output": "Raw"
    }
    "Transcript": [{
        "LoudnessScores": [
            78.63,
            78.37,
            77.98,
            74.18
        ],
        "Content": "[PII], my name is [PII], how can I help?",

            ...
```

```
       "Content": "Well, I would like to cancel my recipe subscription.",
           "IssuesDetected": [{
               "CharacterOffsets": {
                   "Begin": 7,
                   "End": 51
               }
           }],

           ...

       "Content": "That's very sad to hear. Can I offer you a 50% discount to have you
stay with us?",
       "Id": "649afe93-1e59-4ae9-a3ba-a0a613868f5d",
       "BeginOffsetMillis": 12180,
       "EndOffsetMillis": 16960,
       "Sentiment": "NEGATIVE",
       "ParticipantRole": "AGENT"
   },
   {
       "LoudnessScores": [
           80.22,
           79.48,
           82.81
       ],
       "Content": "That is a very generous offer. And I accept.",
       "Id": "f9266cba-34df-4ca8-9cea-4f62a52a7981",
       "BeginOffsetMillis": 17140,
       "EndOffsetMillis": 19860,
       "Sentiment": "POSITIVE",
       "ParticipantRole": "CUSTOMER"
   },
           ...

       "Content": "Wonderful. I made all changes to your account and now this discount
is applied, please check.",
       "OutcomesDetected": [{
       "CharacterOffsets": {
           "Begin": 12,
           "End": 78
       }
       }],

           ...
```

```
        "Content": "I will send an email with all the details to you today, and I will
call you back next week to follow up. Have a wonderful evening.",
        "Id": "78cd0923-cafd-44a5-a66e-09515796572f",
        "BeginOffsetMillis": 31800,
        "EndOffsetMillis": 39450,
        "Sentiment": "POSITIVE",
        "ParticipantRole": "AGENT"
    },
    {
        "LoudnessScores": [
            78.54,
            68.76,
            67.76
        ],
        "Content": "Thank you very much, sir. Goodbye.",
        "Id": "5c5e6be0-8349-4767-8447-986f995af7c3",
        "BeginOffsetMillis": 40040,
        "EndOffsetMillis": 42460,
        "Sentiment": "POSITIVE",
        "ParticipantRole": "CUSTOMER"
    }
    ],

    ...

    "Categories": {
        "MatchedDetails": {
            "positive-resolution": {
                "PointsOfInterest": [{
                    "BeginOffsetMillis": 40040,
                    "EndOffsetMillis": 42460
                }]
            }
        },
        "MatchedCategories": [
            "positive-resolution"
        ]
    },

    ...

    "ConversationCharacteristics": {
        "NonTalkTime": {
```

```
            "Instances": [],
            "TotalTimeMillis": 0
        },
        "Interruptions": {
            "TotalCount": 2,
            "TotalTimeMillis": 10700,
            "InterruptionsByInterrupter": {
                "AGENT": [{
                    "BeginOffsetMillis": 26040,
                    "DurationMillis": 5510,
                    "EndOffsetMillis": 31550
                }],
                "CUSTOMER": [{
                    "BeginOffsetMillis": 770,
                    "DurationMillis": 5190,
                    "EndOffsetMillis": 5960
                }]
            }
        },
        "TotalConversationDurationMillis": 42460,
        "Sentiment": {
            "OverallSentiment": {
                "AGENT": 2.5,
                "CUSTOMER": 2.1
            },
            "SentimentByPeriod": {
                "QUARTER": {
                    "AGENT": [{
                        "Score": 0.0,
                        "BeginOffsetMillis": 0,
                        "EndOffsetMillis": 9862
                    },
                    {
                        "Score": -5.0,
                        "BeginOffsetMillis": 9862,
                        "EndOffsetMillis": 19725
                    },
                    {
                        "Score": 5.0,
                        "BeginOffsetMillis": 19725,
                        "EndOffsetMillis": 29587
                    },
                    {
                       "Score": 5.0,
```

```
                    "BeginOffsetMillis": 29587,
                    "EndOffsetMillis": 39450
                }
                ],
                "CUSTOMER": [{
                    "Score": -2.5,
                    "BeginOffsetMillis": 0,
                    "EndOffsetMillis": 10615
                },
                {
                    "Score": 5.0,
                    "BeginOffsetMillis": 10615,
                    "EndOffsetMillis": 21230
                },
                {
                    "Score": 2.5,
                    "BeginOffsetMillis": 21230,
                    "EndOffsetMillis": 31845
                },
                {
                    "Score": 5.0,
                    "BeginOffsetMillis": 31845,
                    "EndOffsetMillis": 42460
                }
                ]
            }
        }
    },
    "TalkSpeed": {
        "DetailsByParticipant": {
            "AGENT": {
                "AverageWordsPerMinute": 150
            },
            "CUSTOMER": {
                "AverageWordsPerMinute": 167
            }
        }
    },
    "TalkTime": {
        "DetailsByParticipant": {
            "AGENT": {
                "TotalTimeMillis": 32750
            },
            "CUSTOMER": {
```

```
                "TotalTimeMillis": 18010
            }
        },
        "TotalTimeMillis": 50760
    }
},
```

启动实时呼叫分析转录

在开始实时 Call Analytics 转录之前，您必须创建 Amazon Transcribe 要在通话中匹配的所有类别。

> **Note**
>
> 呼叫分析转录无法追溯性地与新类别匹配。只有您在启动呼叫分析转录之前创建的类别才能应用于该转录输出。

如果您创建了一个或多个类别，并且您的音频与至少一个类别中的所有规则匹配， Amazon Transcribe 会使用匹配的类别来标记您的输出。如果您选择不使用类别，或者您的音频与类别中指定的规则不匹配，则不会标记您的转录。

要在实时呼叫分析转录中包含通话后分析，您必须使用 `OutputLocation` 参数在请求中提供一个 Amazon S3 存储桶。您还必须包括对指定存储桶具有写入权限的 `DataAccessRoleArn`。在您的实时呼叫分析流式会话完成后，系统会生成一份单独的转录并将其存储在指定的存储桶中。

借助实时呼叫分析，您还可以选择创建实时类别警报；有关说明，请参阅为类别匹配创建实时警报。

要开始实时 Call Analytics 转录，你可以使用AWS Management Console、HTTP/2 或 WebSockets；查看以下示例：

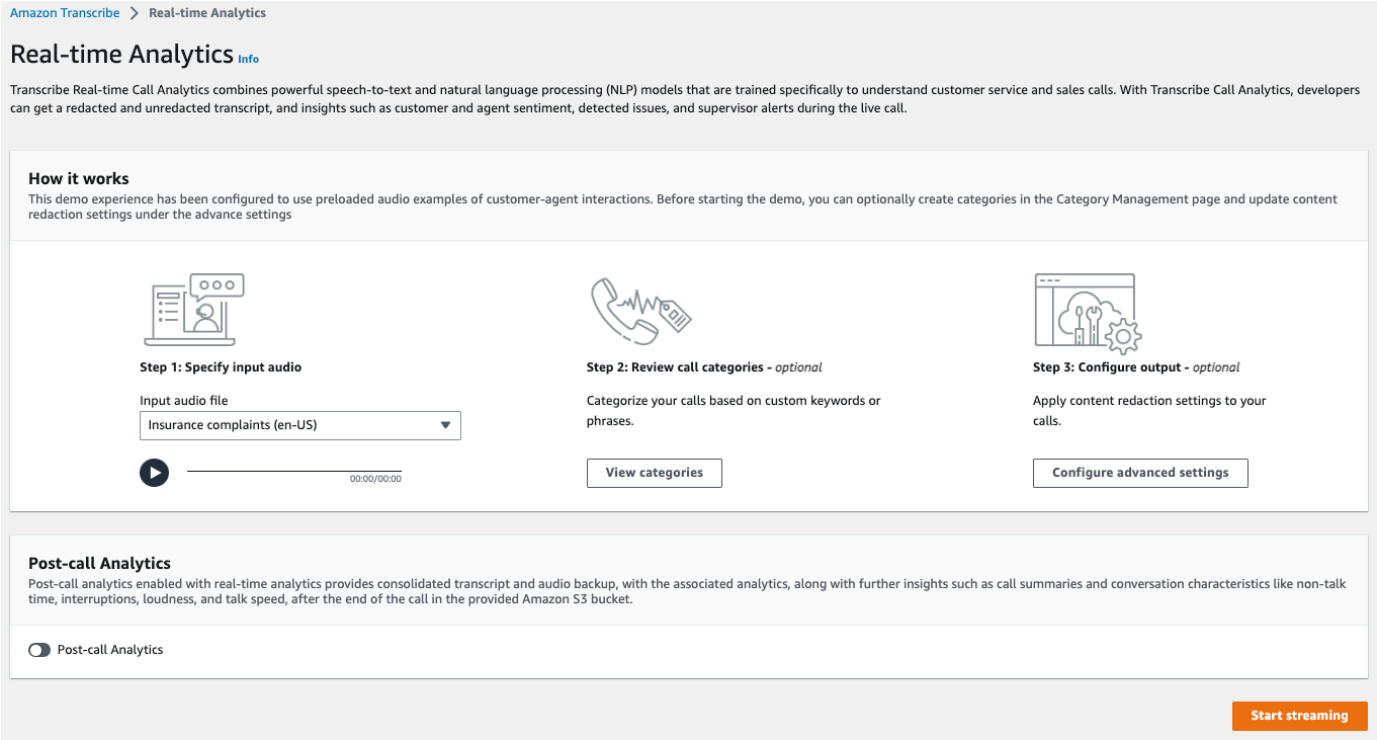
> **Important**
>
> 目前， AWS Management Console 唯一提供带有预加载音频示例的实时呼叫分析演示。如果您想使用自己的音频，则必须使用 API（HTTP/2 或 SDK）。 WebSockets

AWS Management Console

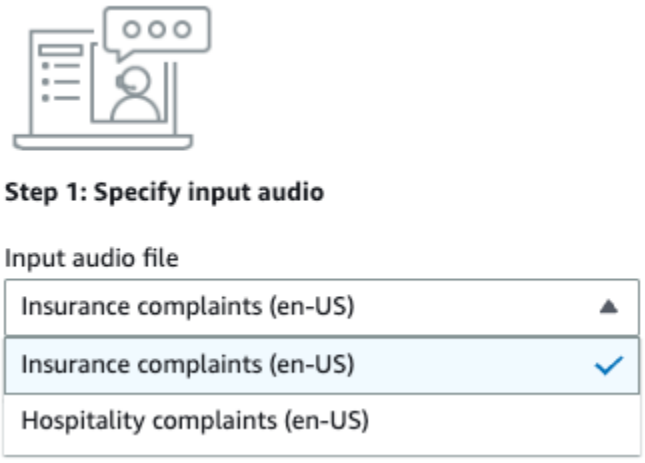
请按以下过程启动呼叫分析请求。符合类别定义的所有特点的通话将使用该类别进行标记。

> **Note**
>
> AWS Management Console中只有演示可用。要启动自定义实时分析转录，必须使用 API。

1. 在导航窗格的 Amazon Transcribe 呼叫分析下，选择分析实时呼叫。

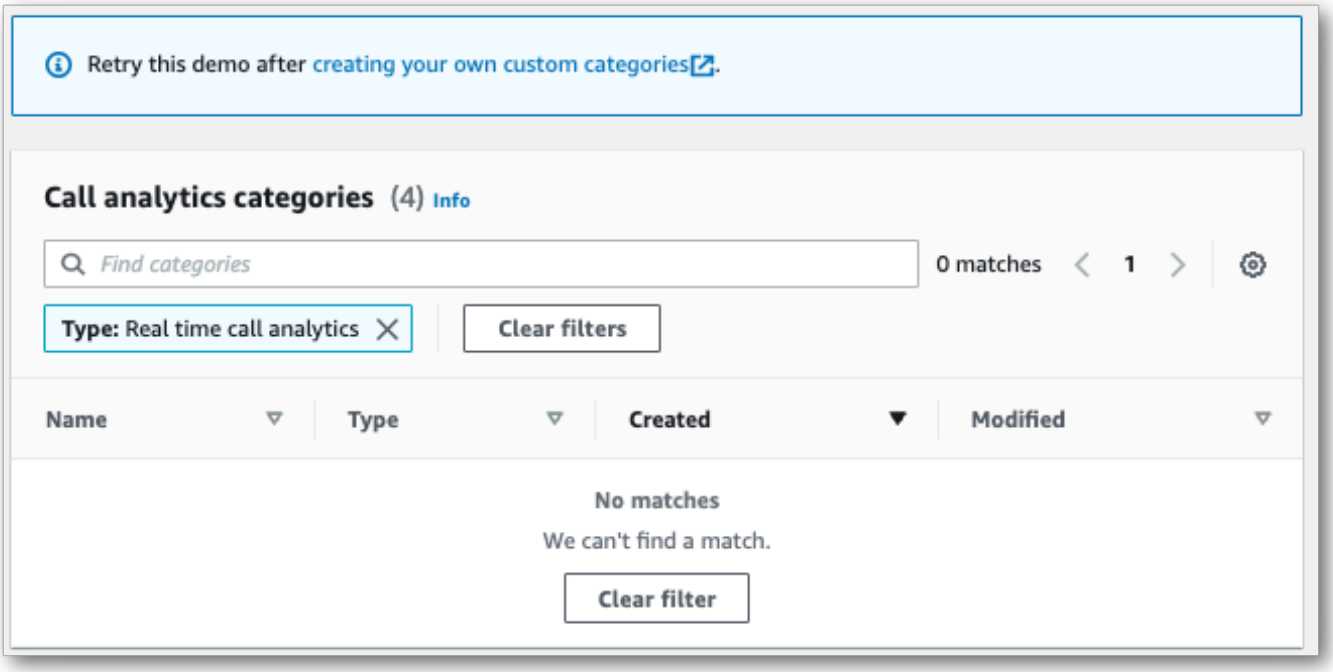


2. 对于步骤 1: 指定输入音频，请从下拉菜单中选择一个演示测试文件。



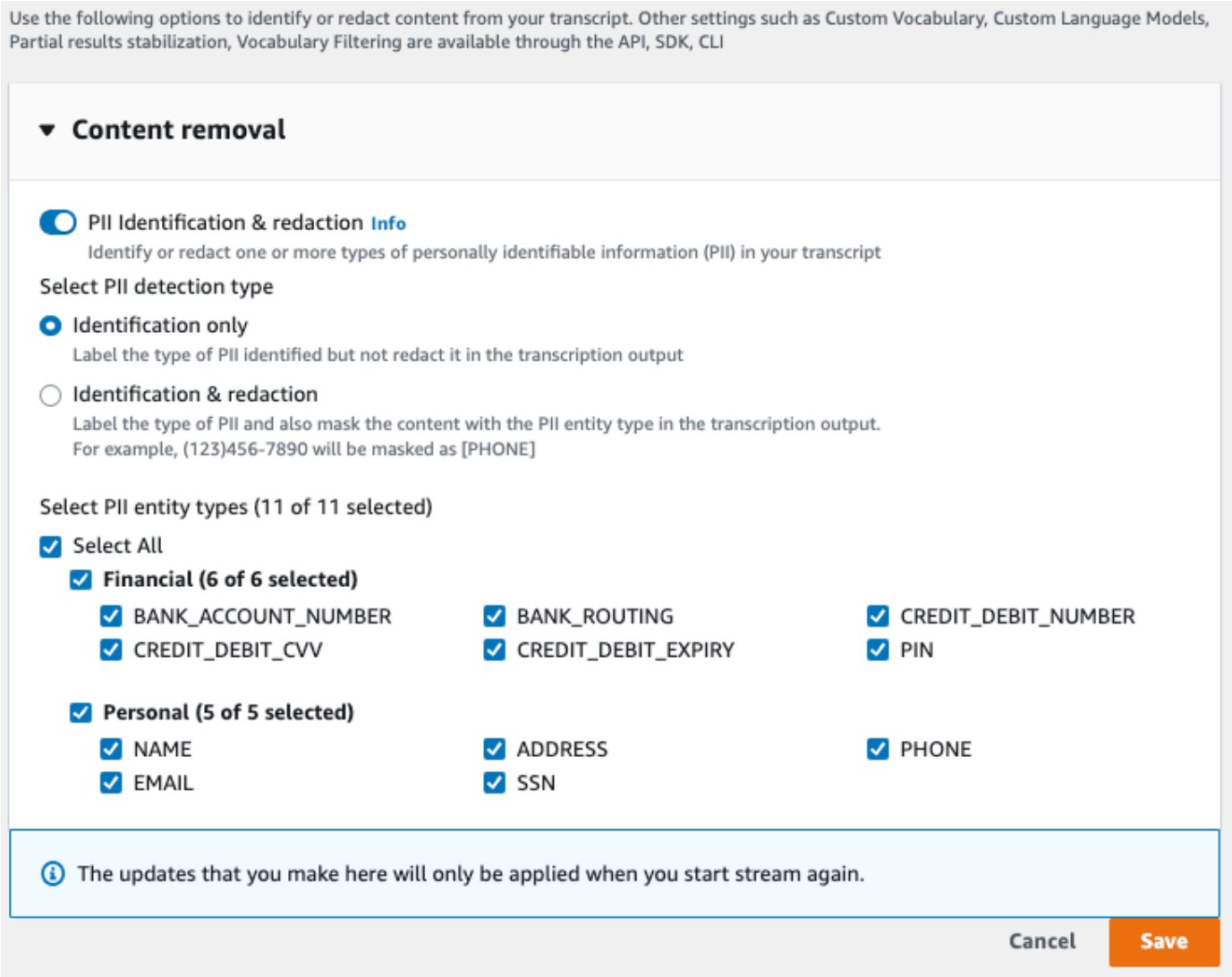
3. 对于第 2 步: 查看通话类别，您可以选择查看之前创建的实时呼叫分析类别。所有实时呼叫分析类别都应用于您的转录。

选择查看类别后将打开一个新窗格，其中显示您现有的实时呼叫分析类别，并提供创建新类别的链接。



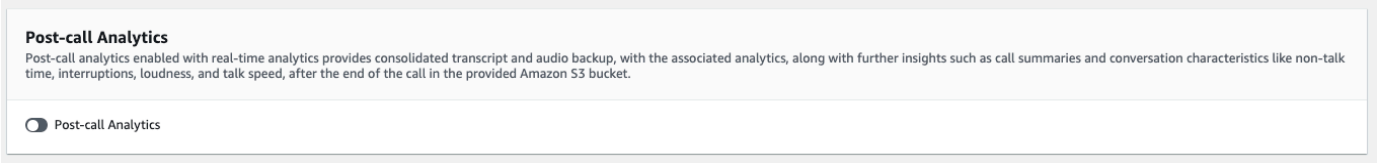
4. 对于步骤 3: 配置输入和输出，您可以选择应用其它设置。

   选择配置高级设置将打开一个新窗格，您可以在其中指定内容编辑设置。

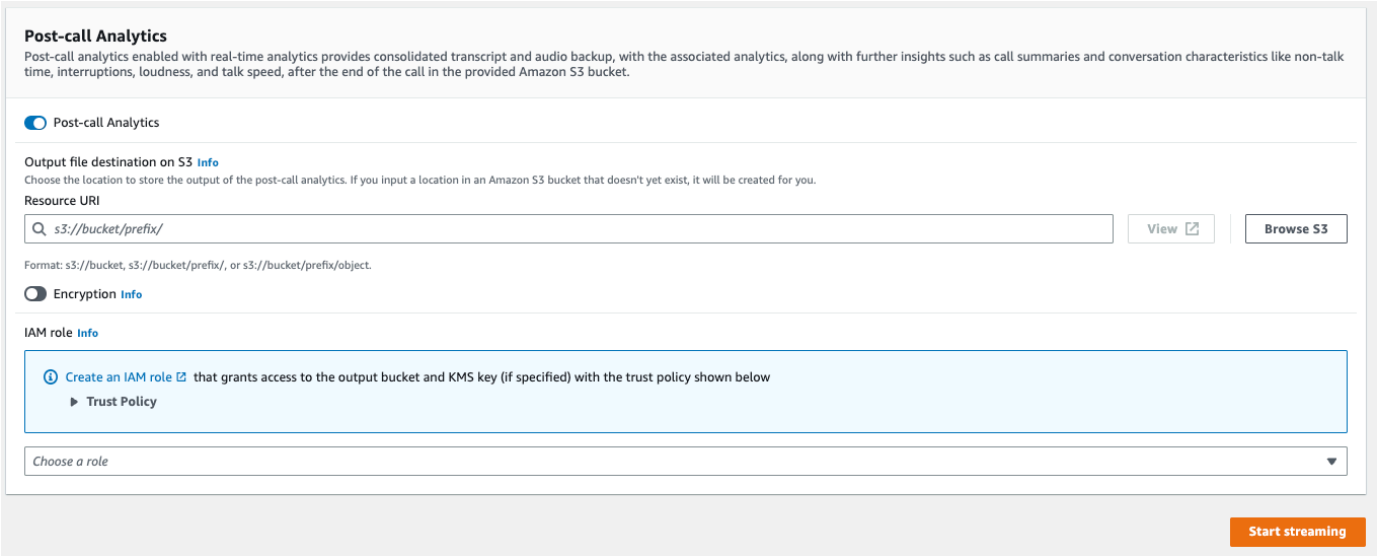


完成所有选择后，选择保存返回主页。

5. 要应用其它分析，您可以开启通话后分析。这为您提供了与通话后分析转录相同的分析，包括中断、音量、非通话时间、通话速度、通话时间、问题、操作项目和结果。通话后分析输出与您的实时呼叫分析转录存储在不同的文件中。



如果您应用电话后分析，则必须指定 Amazon S3 输出文件目标和 IAM 角色。您可以选择对输出进行加密。



6. 选择 Start streaming（开始流式传输）。

HTTP/2 音频流

此示例创建了一个启用了呼叫分析的 HTTP/2 请求。有关使用 HTTP/2 流式传输的更多信息 Amazon Transcribe，请参阅。设置 HTTP/2 音频流有关特定于的参数和标题的更多详细信息 Amazon Transcribe，请参阅StartCallAnalyticsStreamTranscription。

此示例包括通话后分析。如果您不想进行通话后分析，请从请求中删除 PostCallAnalyticsSettings 部分。

请注意，以下示例中显示的配置事件需要作为流中的第一个事件传送。

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
X-Amz-Target: com.amazonaws.transcribe.Transcribe.StartCallAnalyticsStreamTranscription
Content-Type: application/vnd.amazon.eventstream
X-Amz-Content-Sha256: string
X-Amz-Date: 20220208T235959Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key/20220208/us-west-2/transcribe/
aws4_request, SignedHeaders=content-type;host;x-amz-content-sha256;x-amz-date;x-amz-
target;x-amz-security-token, Signature=string
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
transfer-encoding: chunked

{
```

```
    "AudioStream": {
        "AudioEvent": {
            "AudioChunk": blob
        },
        "ConfigurationEvent": {
            "ChannelDefinitions": [
                {
                    "ChannelId": 0,
                    "ParticipantRole": "AGENT"
                },
                {
                     "ChannelId": 1,
                    "ParticipantRole": "CUSTOMER"
                }
            ],
            "PostCallAnalyticsSettings": {
                "OutputLocation": "s3://DOC-EXAMPLE-BUCKET/my-output-files/",
                "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole"
            }
        }
    }
}
```

参数定义可在 API 参考中找到；所有 AWS API 操作的通用参数列在常用参数部分中。

WebSocket 直播

此示例创建了一个在 WebSocket 直播中使用 Call Analytics 的预签名网址。为了便于阅读，已增加了换行符。有关将 WebSocket 直播与配合使用的更多信息 Amazon Transcribe，请参阅设置直 WebSocket 播。有关参数的更多详细信息，请参阅 StartCallAnalyticsStreamTranscription。

此示例包括通话后分析。如果您不想进行通话后分析，请从请求中删除 PostCallAnalyticsSettings 部分。

请注意，以下示例中显示的配置事件需要作为流中的第一个事件传送。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/call-analytics-stream-
transcription-websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
```

```
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&language-code=en-US
&media-encoding=flac
&sample-rate=16000

{
    "AudioStream": {
        "AudioEvent": {
            "AudioChunk": blob
        },
        "ConfigurationEvent": {
            "ChannelDefinitions": [
                {
                    "ChannelId": 0,
                    "ParticipantRole": "AGENT"
                },
                {
                     "ChannelId": 1,
                    "ParticipantRole": "CUSTOMER"
                }
            ],
            "PostCallAnalyticsSettings": {
                "OutputLocation": "s3://DOC-EXAMPLE-BUCKET/my-output-files/",
                "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole"
            }
        }
    }
}
```

参数定义可在 API 参考中找到；所有 AWS API 操作的通用参数列在常用参数部分中。

> **Tip**
>
> 上面的 HTTP/2 和 WebSocket 示例包括通话后分析。如果您不想进行通话后分析，请从请求中删除 PostCallAnalyticsSettings 部分。
> 如果启用 PostCallAnalyticsSettings，则必须将配置事件作为第一个事件发送。您的配置事件包括 ChannelDenifitions 和 PostStreamAnalyticsSettings 的设置，如前面的示例所示。

二进制数据通过 content-type application/octet-stream 以二进制消息的形式传送，配置事件通过 content-type application/json 以文本消息的形式传送。有关更多信息，请参阅 设置流式转录。

为类别匹配创建实时警报

要设置实时警报，必须先创建一个带有 REAL_TIME 标志的 TranscriptFilterType 类别。此标志允许将您的类别应用于实时呼叫分析转录。

有关创建新类别的说明，请参阅 为实时转录创建类别。

当您开始实时呼叫分析转录时，所有带有 REAL_TIME 标记的类别都会自动应用于片段级转录输出。如果出现 TranscriptFilterType 匹配，则会显示在转录的 CategoryEvent 部分下。然后，您可以使用此参数及其子参数 MatchedCategories 和 MatchedDetails，来设置自定义实时警报。

以下是 CategoryEvent 匹配的实时呼叫分析转录输出示例：

```
"CategoryEvent": {
    "MatchedCategories": [ "shipping-complaint" ],
    "MatchedDetails": {
        "my package never arrived" : {
            "TimestampRanges": [
                {
                    "BeginOffsetMillis": 19010,
                    "EndOffsetMillis": 22690
                }
            ]
        }
    }
},
```

前面的示例表示与“我的包裹一直未送达”这句话完全匹配的文本，其代表了“配送投诉”类别中的一条规则。

您可以将实时警报设置为包含所列参数的任意组合。例如，您可以将警报设置为仅包含匹配的短语 (MatchedDetails) 或仅包含类别名称 (MatchedCategories)。或者，您可以将警报设置为包含所有参数。

如何设置实时警报取决于组织的界面和所需的警报类型。例如，您可以将 CategoryEvent 匹配项设置为发送弹出式通知、电子邮件、短信或系统可以接受的任何其它警报。

实时呼叫分析输出

实时呼叫分析记录按区段的 turn-by-turn 格式显示。它们包括类别事件、问题检测、情绪以及 PII 识别和编辑。类别事件允许您设置实时警报；有关更多信息，请参阅为类别匹配创建实时警报。

为了提高准确性并根据您的用例进一步自定义您的转录（例如包括特定行业的术语），请在您的呼叫分析请求中添加自定义词汇表或自定义语言模型。要屏蔽、删除或标记不希望出现在转录结果中的单词（例如脏话），请添加词汇表过滤。

以下各节显示了实时呼叫分析转录的 JSON 输出示例。

类别事件

以下是转录输出中的类别匹配介绍。此示例显示从 19010 毫秒时间戳到 22690 毫秒时间戳的音频与“网络投诉”类别相匹配。在这种情况下，自定义的“网络投诉”类别要求客户提到“网络问题”（字词完全匹配）。

```
"CategoryEvent": {
    "MatchedCategories": [
        "network-complaint"
    ],
    "MatchedDetails": {
        "network issues" : {
            "TimestampRanges": [
                {
                    "BeginOffsetMillis": 9299375,
                    "EndOffsetMillis": 7899375
                }
            ]
        }
    }
},
```

问题检测

以下是转录输出中的问题检测匹配介绍。此示例显示，从字符 26 到字符 62 的文本描述了一个问题。

```
"UtteranceEvent": {
    ...
    "Transcript": "Wang Xiulan I'm tired of the network issues my phone is having.",
    ...
    "IssuesDetected": [
```

```
        {
            "CharacterOffsets": {
                "BeginOffsetChar": 26,
                "EndOffsetChar": 62
            }
        }
    ]
},
```

情绪

以下是转录输出中的情绪分析介绍。

```
"UtteranceEvent": {
    ...
    "Sentiment": "NEGATIVE",
    "Items": [{
        ...
```

PII 识别

以下是转录输出中 PII 识别介绍。

```
"Entities": [
    {
        "Content": "Wang Xiulan",
        "Category": "PII",
        "Type": "NAME",
        "BeginOffsetMillis": 7999375,
        "EndOffsetMillis": 199375,
        "Confidence": 0.9989
    }
],
```

PII 编辑

以下是转录输出中的 PII 编辑介绍。

```
"Content": "[NAME]. Hi, [NAME]. I'm [NAME] Happy to be helping you today.",
"Redaction": {
    "RedactedTimestamps": [
        {
```

```
            "BeginOffsetMillis": 32670,
            "EndOffsetMillis": 33343
        },
        {
            "BeginOffsetMillis": 33518,
            "EndOffsetMillis": 33858
        },
        {
            "BeginOffsetMillis": 34068,
            "EndOffsetMillis": 34488
        }
    ]
},
```

编译后的实时呼叫分析输出

为简洁起见，在以下转录输出中，某些内容被替换为省略号。

```
{
    "CallAnalyticsTranscriptResultStream": {
        "BadRequestException": {},
        "ConflictException": {},
        "InternalFailureException": {},
        "LimitExceededException": {},
        "ServiceUnavailableException": {},
        "UtteranceEvent": {
            "UtteranceId": "58c27f92-7277-11ec-90d6-0242ac120003",
            "ParticipantRole": "CUSTOMER",
            "IsPartial": false,
            "Transcript": "Wang Xiulan I'm tired of the network issues my phone is
 having.",
            "BeginOffsetMillis": 19010,
            "EndOffsetMillis": 22690,
            "Sentiment": "NEGATIVE",
            "Items": [{
                    "Content": "Wang",
                    "BeginOffsetMillis": 379937,
                    "EndOffsetMillis": 299375,
                    "Type": "pronunciation",
                    "Confidence": 0.9961,
                    "VocabularyFilterMatch": false
                },
                {
```

```
                "Content": "Xiulan",
                "EndOffsetMillis": 5899375,
                "BeginOffsetMillis": 3899375,
                "Type": "pronunciation",
                "Confidence": 0.9961,
                "VocabularyFilterMatch": false
            },
            ...
            {
                "Content": "network",
                "EndOffsetMillis": 199375,
                "BeginOffsetMillis": 9299375,
                "Type": "pronunciation",
                "Confidence": 0.9961,
                "VocabularyFilterMatch": false
            },
            {
                "Content": "issues",
                "EndOffsetMillis": 7899375,
                "BeginOffsetMillis": 5999375,
                "Type": "pronunciation",
                "Confidence": 0.9961,
                "VocabularyFilterMatch": false
            },
            {
                "Content": "my",
                "EndOffsetMillis": 9199375,
                "BeginOffsetMillis": 7999375,
                "Type": "pronunciation",
                "Confidence": 0.9961,
                "VocabularyFilterMatch": false
            },
            {
                "Content": "phone",
                "EndOffsetMillis": 199375,
                "BeginOffsetMillis": 9299375,
                "Type": "pronunciation",
                "Confidence": 0.9961,
                "VocabularyFilterMatch": false
            },
            ...
        ],
        "Entities": [{
            "Content": "Wang Xiulan",
```

```
                "Category": "PII",
                "Type": "NAME",
                "BeginOffsetMillis": 7999375,
                "EndOffsetMillis": 199375,
                "Confidence": 0.9989
            }],
            "IssuesDetected": [{
                "CharacterOffsets": {
                    "BeginOffsetChar": 26,
                    "EndOffsetChar": 62
                }
            }]
        },
        "CategoryEvent": {
            "MatchedCategories": [
                "network-complaint"
            ],
            "MatchedDetails": {
                "network issues" : {
                    "TimestampRanges": [
                        {
                            "BeginOffsetMillis": 9299375,
                            "EndOffsetMillis": 7899375
                        }
                    ]
                }
            }
        }
    }
}
```

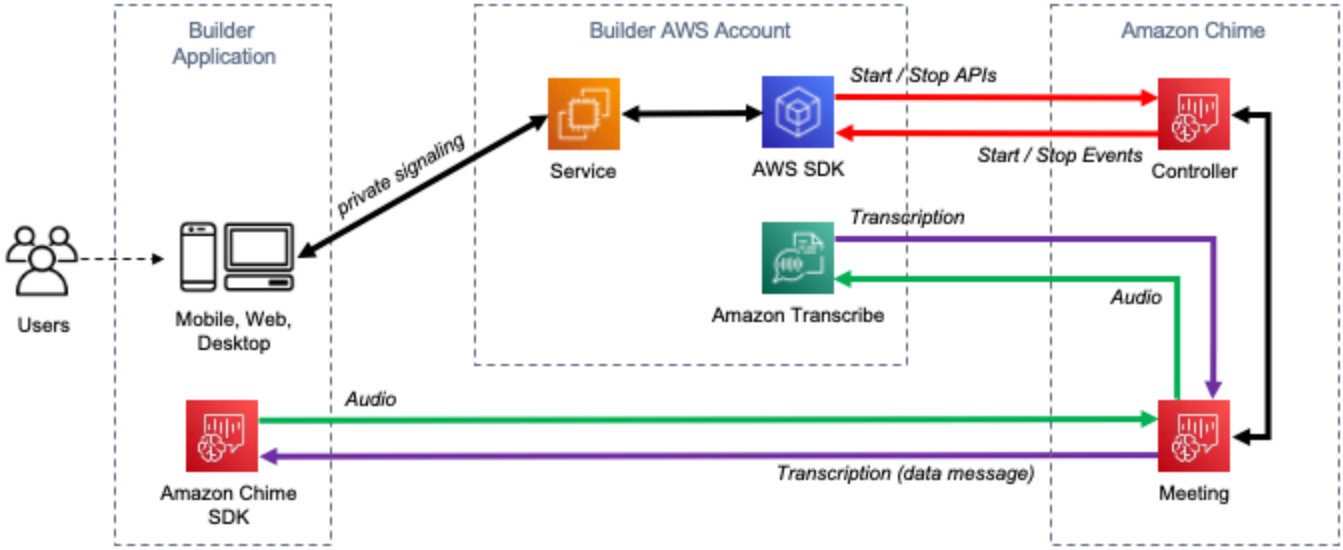
实时转录您的Amazon Chime电话

Amazon Transcribe与Amazon Chime SDK 集成，便于实时转录您的Amazon Chime呼叫。

当您使用Amazon Chime SDK API 请求转录时，会Amazon Chime开始将音频流式传输到，Amazon Transcribe并在通话期间继续这样做。

Amazon ChimeSDK 使用其 “主动说话者” 算法选择前两个活跃的说话者，然后通过单个流将他们的音频Amazon Transcribe作为两个单独的频道发送到。会议参与者通过Amazon Chime SDK 数据消息接收用户归因的转录。您可以在 Amazon ChimeSDK 开发者指南中查看交付示例。

Amazon Chime转录的数据流如下图所示：



有关如何设置实时Amazon Chime转录的更多信息和详细说明，请参阅 Amazon ChimeSDK 开发人员指南中的使用 SAmazon Chime DK 实时转录。有关 API 操作的信息，请参阅 Amazon ChimeSDK API 参考文档。

> 通过Machine LearnAWS ing 博客深入了解
>
> 要了解有关通过实时转录提高准确性的更多信息，请参阅：
>
> - Amazon ChimeSDK 会议现在支持使用Amazon Transcribe和进行实时转录Amazon Transcribe Medical
> - Amazon Chime用于远程医疗解决方案的 SDK

使用软件开发工具包的 Amazon Transcribe 的代码示例 AWS

以下代码示例展示了如何将 Amazon Transcribe 与 AWS 软件开发套件 (SDK) 一起使用。

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和跨服务示例的上下文查看操作。

场景 是展示如何通过在同一服务中调用多个函数来完成特定任务的代码示例。

跨服务示例是指跨多个 AWS 服务工作的示例应用程序。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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

- 构建 Amazon Transcribe 流式传输应用程序
- 使用 AWS SDK 将文本转换为语音并转换回文本

使用软件开发工具包进行 Amazon Transcribe 的操作 AWS

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 Transcribe AWS 操作。这些代码节选调用了 Amazon Transcribe API，是必须在上下文中运行的较大型程序的代码节选。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。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Transcribe API 参考。

示例

- 使用软件开发工具包创建自定义 Amazon Transcribe 词汇 AWS
- 使用软件开发工具包删除自定义 Amazon Transcribe 词汇 AWS
- 使用软件开发工具包删除 Amazon Transcribe 医疗转录作业 AWS
- 使用软件开发工具包删除 Amazon Transcribe 转录作业 AWS
- 使用软件开发工具包获取自定义 Amazon Transcribe 词汇 AWS
- 使用软件开发工具包获取 Amazon Transcribe 转录作业 AWS
- 使用软件开发工具包列出自定义 Amazon Transcribe 词汇表 AWS
- 使用 SDK 列出 Amazon Transcribe 医疗转录作业 AWS
- 使用软件开发工具包列出 Amazon Transcribe 转录作业 AWS
- 使用软件开发工具包使用 Amazon Transcribe 生成实时转录 AWS
- 使用软件开发工具包开始 Amazon Transcribe 医疗转录作业 AWS
- 使用软件开发工具包开始 Amazon Transcribe 转录作业 AWS
- 使用软件开发工具包更新自定义 Amazon Transcribe 词汇 AWS

使用软件开发工具包创建自定义 Amazon Transcribe 词汇 AWS

以下代码示例演示了如何创建自定义 Amazon Transcribe 词汇表。

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：

- 创建和完善自定义词汇表

.NET

AWS SDK for .NET

> **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。

```
    /// <summary>
    /// Create a custom vocabulary using a list of phrases. Custom vocabularies
    /// improve transcription accuracy for one or more specific words.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> CreateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.CreateVocabularyAsync(
            new CreateVocabularyRequest
            {
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考CreateVocabulary中的。

CLI

AWS CLI

创建自定义词汇表

以下 create-vocabulary 示例创建一个自定义词汇表。要创建自定义词汇表，您必须创建一个文本文件，其中包含要更准确地进行转录的所有术语。对于 vocabulary-file-uri，请指定该文本文件的亚马逊简单存储服务 (Amazon S3) Service URI。对于 language-code，指定与自定义词汇表的语言对应的语言代码。对于 vocabulary-name，指定所需的自定义词汇表名称。

```
aws transcribe create-vocabulary \
    --language-code language-code \
    --vocabulary-name cli-vocab-example \
    --vocabulary-file-uri s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/the-text-file-
for-the-custom-vocabulary.txt
```

输出：

```
{
    "VocabularyName": "cli-vocab-example",
    "LanguageCode": "language-code",
    "VocabularyState": "PENDING"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的自定义词汇表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateVocabulary中的。

Python

SDK for Python (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。

```
def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary
 processing
```

```
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.",
vocabulary_name)
        raise
    else:
        return response
```

- 有关 API 的详细信息，请参阅适用CreateVocabulary于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包删除自定义 Amazon Transcribe 词汇 AWS

以下代码示例演示了如何删除自定义 Amazon Transcribe 词汇表。

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：

- 创建和完善自定义词汇表

.NET

AWS SDK for .NET

> **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。

```
/// <summary>
/// Delete an existing custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.DeleteVocabularyAsync(
        new DeleteVocabularyRequest
        {
            VocabularyName = vocabularyName
        });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DeleteVocabulary中的。

CLI

AWS CLI

删除自定义词汇表

以下 delete-vocabulary 示例删除一个自定义词汇表。

```
aws transcribe delete-vocabulary \
    --vocabulary-name vocabulary-name
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的自定义词汇表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteVocabulary中的。

Python

SDK for Python (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。

```
def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

- 有关 API 的详细信息，请参阅适用DeleteVocabulary于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包删除 Amazon Transcribe 医疗转录作业 AWS

以下代码示例演示了如何删除 Amazon Transcribe Medical 转录作业。

.NET

AWS SDK for .NET

> **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。

```
    /// <summary>
    /// Delete a medical transcription job. Also deletes the transcript
 associated with the job.
    /// </summary>
    /// <param name="jobName">Name of the medical transcription job to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMedicalTranscriptionJob(string jobName)
    {
        var response = await
 _amazonTranscribeService.DeleteMedicalTranscriptionJobAsync(
            new DeleteMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DeleteMedicalTranscriptionJob中的。

CLI

AWS CLI

删除医疗转录作业

以下 delete-medical-transcription-job 示例删除一个医疗转录作业。

```
aws transcribe delete-medical-transcription-job \
    --medical-transcription-job-name medical-transcription-job-name
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon T DeleteMedicalTranscriptionJobranscribe 开发者指南》。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteMedicalTranscriptionJob中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

创建客户端。

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

删除医疗转录作业。

```
// Import the required AWS SDK clients and commands for Node.js
```

```
import { DeleteMedicalTranscriptionJobCommand } from "@aws-sdk/client-
transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // For example,
 'medical_transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteMedicalTranscriptionJobCommand(params)
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DeleteMedicalTranscriptionJob中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包删除 Amazon Transcribe 转录作业 AWS

以下代码示例演示了如何删除 Amazon Transcribe 转录作业。

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：

- 创建和完善自定义词汇表

.NET

AWS SDK for .NET

> **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。

```
    /// <summary>
    /// Delete a transcription job. Also deletes the transcript associated with
the job.
    /// </summary>
    /// <param name="jobName">Name of the transcription job to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTranscriptionJob(string jobName)
    {
        var response = await
_amazonTranscribeService.DeleteTranscriptionJobAsync(
            new DeleteTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DeleteTranscriptionJob中的。

CLI

AWS CLI

删除一个转录作业

以下 `delete-transcription-job` 示例删除一个转录作业。

```
aws transcribe delete-transcription-job \
```

```
    --transcription-job-name your-transcription-job
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon T DeleteTranscriptionJobranscribe 开发者指南》。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteTranscriptionJob中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **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。

删除转录作业。

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME", // Required. For example, 'transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteTranscriptionJobCommand(params)
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

创建客户端。

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DeleteTranscriptionJob中的。

Python

SDK for Python (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。

```
def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise
```

- 有关 API 的详细信息，请参阅适用DeleteTranscriptionJob于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包获取自定义 Amazon Transcribe 词汇 AWS

以下代码示例演示了如何获取自定义 Amazon Transcribe 词汇表。

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：

- 创建和完善自定义词汇表

.NET

AWS SDK for .NET

> **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。

```
/// <summary>
/// Get information about a custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary.</param>
/// <returns>The state of the custom vocabulary.</returns>
public async Task<VocabularyState> GetCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.GetVocabularyAsync(
        new GetVocabularyRequest()
```

```
            {
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考GetVocabulary中的。

CLI

AWS CLI

获取有关自定义词汇表的信息

以下 get-vocabulary 示例获取有关以前创建的自定义词汇表的信息。

```
aws transcribe get-vocabulary \
    --vocabulary-name cli-vocab-1
```

输出：

```
{
    "VocabularyName": "cli-vocab-1",
    "LanguageCode": "language-code",
    "VocabularyState": "READY",
    "LastModifiedTime": "2020-09-19T23:22:32.836000+00:00",
    "DownloadUri": "https://link-to-download-the-text-file-used-to-create-your-
custom-vocabulary"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的自定义词汇表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考GetVocabulary中的。

Python

SDK for Python (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。

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response =
 transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

- 有关 API 的详细信息，请参阅适用GetVocabulary于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包获取 Amazon Transcribe 转录作业 AWS

以下代码示例演示了如何获取 Amazon Transcribe 转录作业。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建和完善自定义词汇表
- 转录音频并获取作业数据

.NET

AWS SDK for .NET

> **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。

```
    /// <summary>
    /// Get details about a transcription job.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <returns>A TranscriptionJob instance with information on the requested
job.</returns>
    public async Task<TranscriptionJob> GetTranscriptionJob(string jobName)
    {
        var response = await _amazonTranscribeService.GetTranscriptionJobAsync(
            new GetTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.TranscriptionJob;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考GetTranscriptionJob中的。

CLI

AWS CLI

获取有关特定转录作业的信息

以下 `get-transcription-job` 示例获取有关特定转录作业的信息。要访问转录结果，请使用 TranscriptFileUri 参数。使用 MediaFileUri 参数查看您使用此作业转录了哪个音频文件。您可以使用 Settings 对象查看在转录作业中启用的可选功能。

```
aws transcribe get-transcription-job \
    --transcription-job-name your-transcription-job
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "your-transcription-job",
        "TranscriptionJobStatus": "COMPLETED",
        "LanguageCode": "language-code",
        "MediaSampleRateHertz": 48000,
        "MediaFormat": "mp4",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.file-
extension"
        },
        "Transcript": {
            "TranscriptFileUri": "https://Amazon-S3-file-location-of-
transcription-output"
        },
        "StartTime": "2020-09-18T22:27:23.970000+00:00",
        "CreationTime": "2020-09-18T22:27:23.948000+00:00",
        "CompletionTime": "2020-09-18T22:28:21.197000+00:00",
        "Settings": {
            "ChannelIdentification": false,
            "ShowAlternatives": false
        },
        "IdentifyLanguage": true,
        "IdentifiedLanguageScore": 0.8672199249267578
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的入门（AWS 命令行界面）。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考GetTranscriptionJob中的。

Python

SDK for Python (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。

```
def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job
```

- 有关 API 的详细信息，请参阅适用GetTranscriptionJob于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包列出自定义 Amazon Transcribe 词汇表 AWS

以下代码示例演示了如何列出自定义 Amazon Transcribe 词汇表。

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：

- 创建和完善自定义词汇表

.NET

AWS SDK for .NET

> **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。

```
   /// <summary>
   /// List custom vocabularies for the current account. Optionally specify a
name
   /// filter and a specific state to filter the vocabularies list.
   /// </summary>
   /// <param name="nameContains">Optional string the vocabulary name must
contain.</param>
   /// <param name="stateEquals">Optional state of the vocabulary.</param>
   /// <returns>List of information about the vocabularies.</returns>
   public async Task<List<VocabularyInfo>> ListCustomVocabularies(string?
nameContains = null,
       VocabularyState? stateEquals = null)
   {
       var response = await _amazonTranscribeService.ListVocabulariesAsync(
           new ListVocabulariesRequest()
           {
               NameContains = nameContains,
               StateEquals = stateEquals
           });
       return response.Vocabularies;
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考ListVocabularies中的。

CLI

AWS CLI

列出自定义词汇表

以下list-vocabularies示例列出了与您的 AWS 账户和地区关联的自定义词汇表。

```
aws transcribe list-vocabularies
```

输出：

```
{
    "NextToken": "NextToken",
    "Vocabularies": [
        {
            "VocabularyName": "ards-test-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-27T22:00:27.330000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "sample-test",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T23:04:11.044000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-test-3-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T22:12:22.277000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-test-2",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T21:53:50.455000+00:00",
            "VocabularyState": "READY"
```

```
        },
        {
            "VocabularyName": "CRLF-to-LF-1-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T21:39:33.356000+00:00",
            "VocabularyState": "READY"
        }
    ]
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的自定义词汇表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListVocabularies中的。

Python

SDK for Python (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。

```
def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string
 in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
 transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
```

```
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs
```

- 有关 API 的详细信息，请参阅适用ListVocabularies于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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 SDK 列出 Amazon Transcribe 医疗转录作业 AWS

以下代码示例演示了如何列出 Amazon Transcribe Medical 转录作业。

.NET

AWS SDK for .NET

> **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。

```
/// <summary>
/// List medical transcription jobs, optionally with a name filter.
/// </summary>
```

```
    /// <param name="jobNameContains">Optional name filter for the medical
transcription jobs.</param>
    /// <returns>A list of summaries about medical transcription jobs.</returns>
    public async Task<List<MedicalTranscriptionJobSummary>>
ListMedicalTranscriptionJobs(
        string? jobNameContains = null)
    {
        var response = await
_amazonTranscribeService.ListMedicalTranscriptionJobsAsync(
            new ListMedicalTranscriptionJobsRequest()
            {
                JobNameContains = jobNameContains
            });
        return response.MedicalTranscriptionJobSummaries;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[ListMedicalTranscriptionJobs](#)中的。

CLI

AWS CLI

列出医疗转录作业

以下`list-medical-transcription-jobs`示例列出了与您的 AWS 账户和地区相关的医疗转录作业。要获取有关特定转录作业的更多信息，请在转录输出中复制 MedicalTranscriptionJobName 参数的值，然后为命令的`MedicalTranscriptionJobName`选项指定该值。`get-medical-transcription-job`要查看更多转录作业，请复制 NextToken 参数的值，再次运行该`list-medical-transcription-jobs`命令，然后在`--next-token`选项中指定该值。

```
aws transcribe list-medical-transcription-jobs
```

输出：

```
{
    "NextToken": "3/PblzkiGhzjER3KHuQt2fmbPLF7cDYafjFMEoGn44ON/
gsuUSTIkGyanvRE6WMXFd/ZTEc2EZj+P9eii/
```

```
z1O2FDYli6RLI0WoRX4RwMisVrh9G0Kie0Y8ikBCdtqlZB10Wa9McC+ebOl
+LaDtZPC4u6ttoHLRlEfzqstHXSgapXg3tEBtm9piIaPB6MOM5BB6t86+qtmocTR/
qrteHZBBudhTfbCwhsxaqujHiiUvFdm3BQbKKWIW06yV9b+4f38oD2lVIan
+vfUs3gBYAl5VTDmXXzQPBQOHPjtwmFI+IWX15nSUjWuN3TUylHgPWzDaYT8qBtu0Z+3UG4V6b
+K2CC0XszXg5rBq9hYgNzy4XoFh/6s5DoSnzq49Q9xHgHdT2yBADFmvFK7myZBsj75+2vQZOSVpWUPy3WT/32zFAc
+mFYfUjtTZ8n/jq7aQEjQ42A
+X/7K6JgOcdVPtEg8PlDr5kgYYG3q3OmYXX37U3FZuJmnTI63VtIXsNnOU5eGoYObtpk00Nq9UkzgSJxqj84ZD5n
+S0EGy9ZUYBJRRcGeYUM3Q4DbSJfUwSAqcFdLIWZdp8qIREMQIBWy7BLwSdyqsQo2vRrd53hm5aWM7SVf6pPq6X/
IXR5+1eUOOD8/coaTT4ES2DerbV6RkV4o0VT1d0SdVX/
MmtkNG8nYj8PqU07w7988quh1ZP6D80veJS1q73tUUR9MjnGernW2tAnvnLNhdefBcD
+sZVfYq3iBMFY7wTy1P1G6NqW9GrYDYoX3tTPWlD7phpbVSyKrh/
PdYrps5UxnsGoA1b7L/FfAXDfUoGrGUB4N3JsPYXX9D++g+6gV1qBBs/
WfF934aKqfD6UTggm/zV3GAOWiBpfvAZRvEb924i6yGHyMC7y54O1ZAwSBupmI
+FFd13CaPO4kN1vJlth6aM5vUPXg4BpyUhtbRhwD/KxCvf9K0tLJGyL1A==",
    "MedicalTranscriptionJobSummaries": [
        {
            "MedicalTranscriptionJobName": "vocabulary-dictation-medical-
transcription-job",
            "CreationTime": "2020-09-21T21:17:27.016000+00:00",
            "StartTime": "2020-09-21T21:17:27.045000+00:00",
            "CompletionTime": "2020-09-21T21:17:59.561000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "DICTATION"
        },
        {
            "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
            "CreationTime": "2020-09-21T21:01:14.569000+00:00",
            "StartTime": "2020-09-21T21:01:14.592000+00:00",
            "CompletionTime": "2020-09-21T21:01:43.606000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "DICTATION"
        },
        {
            "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-21T19:09:18.171000+00:00",
            "StartTime": "2020-09-21T19:09:18.199000+00:00",
```

```
            "CompletionTime": "2020-09-21T19:10:22.516000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        },
        {
            "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-21T18:43:37.157000+00:00",
            "StartTime": "2020-09-21T18:43:37.265000+00:00",
            "CompletionTime": "2020-09-21T18:44:21.192000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        },
        {
            "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-20T23:46:44.053000+00:00",
            "StartTime": "2020-09-20T23:46:44.081000+00:00",
            "CompletionTime": "2020-09-20T23:47:35.851000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        }
    ]
}
```

欲了解更多信息，请参阅《Amazon Transcribe 开发者batch-med-transcription指南》中的 https://docs.aws.amazon.com/transcribe/latest/dg/ .html>。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListMedicalTranscriptionJobs中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

创建客户端。

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

列出医疗转录作业。

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and
 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the
 same region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
```

```
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params)
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考ListMedicalTranscriptionJobs中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包列出 Amazon Transcribe 转录作业 AWS

以下代码示例演示了如何列出 Amazon Transcribe 转录作业。

.NET

AWS SDK for .NET

> **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。

```
    /// <summary>
    /// List transcription jobs, optionally with a name filter.
```

```
    /// </summary>
    /// <param name="jobNameContains">Optional name filter for the transcription
jobs.</param>
    /// <returns>A list of transcription job summaries.</returns>
    public async Task<List<TranscriptionJobSummary>>
ListTranscriptionJobs(string? jobNameContains = null)
    {
        var response = await _amazonTranscribeService.ListTranscriptionJobsAsync(
            new ListTranscriptionJobsRequest()
            {
                JobNameContains = jobNameContains
            });
        return response.TranscriptionJobSummaries;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[ListTranscriptionJobs](#)中的。

CLI

AWS CLI

列出转录作业

以下`list-transcription-jobs`示例列出了与您的 AWS 账户和地区相关的转录作业。

```
aws transcribe list-transcription-jobs
```

输出：

```
{
    "NextToken": "NextToken",
    "TranscriptionJobSummaries": [
        {
            "TranscriptionJobName": "speak-id-job-1",
            "CreationTime": "2020-08-17T21:06:15.391000+00:00",
            "StartTime": "2020-08-17T21:06:15.416000+00:00",
            "CompletionTime": "2020-08-17T21:07:05.098000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
```

```
        },
        {
            "TranscriptionJobName": "job-1",
            "CreationTime": "2020-08-17T20:50:24.207000+00:00",
            "StartTime": "2020-08-17T20:50:24.230000+00:00",
            "CompletionTime": "2020-08-17T20:52:18.737000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "sdk-test-job-4",
            "CreationTime": "2020-08-17T20:32:27.917000+00:00",
            "StartTime": "2020-08-17T20:32:27.956000+00:00",
            "CompletionTime": "2020-08-17T20:33:15.126000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "Diarization-speak-id",
            "CreationTime": "2020-08-10T22:10:09.066000+00:00",
            "StartTime": "2020-08-10T22:10:09.116000+00:00",
            "CompletionTime": "2020-08-10T22:26:48.172000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "your-transcription-job-name",
            "CreationTime": "2020-07-29T17:45:09.791000+00:00",
            "StartTime": "2020-07-29T17:45:09.826000+00:00",
            "CompletionTime": "2020-07-29T17:46:20.831000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        }
    ]
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的入门（AWS 命令行界面）。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListTranscriptionJobs中的。

Java

适用于 Java 2.x 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。

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

            listTranscriptionJobs(transcribeClient);
        }

        public static void listTranscriptionJobs(TranscribeClient
 transcribeClient) {
            ListTranscriptionJobsRequest listJobsRequest =
 ListTranscriptionJobsRequest.builder()
                .build();


 transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
                .flatMap(response ->
 response.transcriptionJobSummaries().stream())
                .forEach(jobSummary -> {
                    System.out.println("Job Name: " +
 jobSummary.transcriptionJobName());
                    System.out.println("Job Status: " +
 jobSummary.transcriptionJobStatus());
                    System.out.println("Output Location: " +
 jobSummary.outputLocationType());
                    // Add more information as needed

                    // Retrieve additional details for the job if necessary
                    GetTranscriptionJobResponse jobDetails =
 transcribeClient.getTranscriptionJob(
                        GetTranscriptionJobRequest.builder()
```

```
 .transcriptionJobName(jobSummary.transcriptionJobName())
                            .build());

                    // Display additional details
                    System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
                    System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
                    // Add more details as needed

                    System.out.println("--------------");
                });
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListTranscriptionJobs中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **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。

列出转录作业。

```
// Import the required AWS SDK clients and commands for Node.js

import { ListTranscriptionJobsCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  JobNameContains: "KEYWORD", // Not required. Returns only transcription
  // job names containing this string
};
```

```
export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new ListTranscriptionJobsCommand(params)
    );
    console.log("Success", data.TranscriptionJobSummaries);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

创建客户端。

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考ListTranscriptionJobs中的。

Python

SDK for Python (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。

```
def list_jobs(job_filter, transcribe_client):
    """
```

```
    Lists summaries of the transcription jobs for the current AWS account.

    :param job_filter: The list of returned jobs must contain this string in
their
                       names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved transcription job summaries.
    """
    try:
        response =
transcribe_client.list_transcription_jobs(JobNameContains=job_filter)
        jobs = response["TranscriptionJobSummaries"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_transcription_jobs(
                JobNameContains=job_filter, NextToken=next_token
            )
            jobs += response["TranscriptionJobSummaries"]
            next_token = response.get("NextToken")
        logger.info("Got %s jobs with filter %s.", len(jobs), job_filter)
    except ClientError:
        logger.exception("Couldn't get jobs with filter %s.", job_filter)
        raise
    else:
        return jobs
```

- 有关 API 的详细信息，请参阅适用ListTranscriptionJobs于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包使用 Amazon Transcribe 生成实时转录 AWS

以下代码示例演示了如何通过 Amazon Transcribe 制作实时转录。

C++

SDK for C++

> **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。

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
 managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if
 the SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
                [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
 &error) {
                        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
                });
```

```
        //SetTranscriptEventCallback called for every 'chunk' of file
transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                    }
                    for (auto &&alt: r.GetAlternatives()) {
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [](AudioStream &stream) {
                Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
                if (!file.is_open()) {
                    std::cerr << "Failed to open " << FILE_NAME << '\n';
                }
                std::array<char, BUFFER_SIZE> buf;
                int i = 0;
                while (file) {
                    file.read(&buf[0], buf.size());

                    if (!file)
                        std::cout << "File: only " << file.gcount() << " could be
read"
                                  << std::endl;

                    Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
                    AudioEvent event(std::move(bits));
                    if (!stream) {
                        std::cerr << "Failed to create a stream" << std::endl;
```

```
                        break;
                    }
                    //The std::basic_istream::gcount() is used to count the
 characters in the given string. It returns
                    //the number of characters extracted by the last read()
 operation.
                    if (file.gcount() > 0) {
                        if (!stream.WriteAudioEvent(event)) {
                            std::cerr << "Failed to write an audio event" <<
 std::endl;
                            break;
                        }
                    }
                    else {
                        break;
                    }
                    std::this_thread::sleep_for(std::chrono::milliseconds(
                            25)); // Slow down because we are streaming from a
 file.
                }
                if (!stream.WriteAudioEvent(
                        AudioEvent())) {
                    // Per the spec, we have to send an empty event (an event
 without a payload) at the end.
                    std::cerr << "Failed to send an empty frame" << std::endl;
                }
                else {
                    std::cout << "Successfully sent the empty frame" <<
 std::endl;
                }
                stream.flush();
                stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
        auto OnResponseCallback = [&signaling](
                const TranscribeStreamingServiceClient * /*unused*/,
                const Model::StartStreamTranscriptionRequest & /*unused*/,
                const Model::StartStreamTranscriptionOutcome &outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

                if (!outcome.IsSuccess()) {
```

```
                    std::cerr << "Transcribe streaming error "
                              << outcome.GetError().GetMessage() << std::endl;
                }

                signaling.Release();
        };

        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
 OnResponseCallback,
                                             nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
 done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考StartStreamTranscriptionAsync中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包开始 Amazon Transcribe 医疗转录作业 AWS

以下代码示例演示了如何启动 Amazon Transcribe Medical 转录作业。

.NET

AWS SDK for .NET

> **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。

```
    /// <summary>
    /// Start a medical transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the medical transcription job.</
param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon
 S3 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="outputBucketName">Location for the output, typically an
 Amazon S3 location.</param>
    /// <param name="transcriptionType">Conversation or dictation transcription
 type.</param>
    /// <returns>A MedicalTransactionJob instance with information on the new
 job.</returns>
    public async Task<MedicalTranscriptionJob> StartMedicalTranscriptionJob(
        string jobName, string mediaFileUri,
        MediaFormat mediaFormat, string outputBucketName,
 Amazon.TranscribeService.Type transcriptionType)
    {
        var response = await
 _amazonTranscribeService.StartMedicalTranscriptionJobAsync(
            new StartMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode =
                    LanguageCode
                        .EnUS, // The value must be en-US for medical
 transcriptions.
                OutputBucketName = outputBucketName,
                OutputKey =
                    jobName, // The value is a key used to fetch the output of
 the transcription.
                Specialty = Specialty.PRIMARYCARE, // The value PRIMARYCARE must
 be set.
                Type = transcriptionType
            });
```

```
        return response.MedicalTranscriptionJob;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考StartMedicalTranscriptionJob中的。

CLI

AWS CLI

示例 1：转录存储为音频文件的医疗口述

以下 start-medical-transcription-job 示例转录一个音频文件。您可以在 OutputBucketName 参数中指定转录输出位置。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfile.json
```

myfile.json 的内容：

```
{
    "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    }
}
```

输出：

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-
job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
```

```
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T00:35:22.256000+00:00",
        "CreationTime": "2020-09-20T00:35:22.218000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的批量转录概述。

示例 2：转录存储为音频文件的临床医生与患者之间的对话

以下 start-medical-transcription-job 示例转录包含有临床医生与患者之间对话的音频文件。您可以在 OutputBucketName 参数中指定转录输出的位置。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysecondfile.json
```

mysecondfile.json 的内容：

```
{
    "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    }
}
```

输出：

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
```

```
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T23:19:49.965000+00:00",
        "CreationTime": "2020-09-20T23:19:49.941000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的批量转录概述。

示例 3：转录临床医生与患者之间对话的多声道音频文件

以下 `start-medical-transcription-job` 示例转录音频文件中每个声道的音频，并将每个声道的单独转录合并为一个转录输出。您可以在 `OutputBucketName` 参数中指定转录输出位置。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mythirdfile.json
```

`mythirdfile.json` 的内容：

```
{
    "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"DOC-EXAMPLE-BUCKET",
        "Media": {
          "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
        },
        "Settings":{
          "ChannelIdentification": true
        }
}
```

输出：

```
{
```

```
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T23:46:44.081000+00:00",
        "CreationTime": "2020-09-20T23:46:44.053000+00:00",
        "Settings": {
            "ChannelIdentification": true
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的声道识别。

示例 4：转录临床医生与患者之间对话的音频文件并在转录输出中识别发言者

以下 `start-medical-transcription-job` 示例转录一个音频文件，并在转录输出中标记每个发言者的语音。您可以在 `OutputBucketName` 参数中指定转录输出位置。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfourthfile.json
```

`myfourthfile.json` 的内容：

```
{
    "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
        },
    "Settings":{
        "ShowSpeakerLabels": true,
```

```
        "MaxSpeakerLabels": 2
        }
}
```

输出：

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T18:43:37.265000+00:00",
        "CreationTime": "2020-09-21T18:43:37.157000+00:00",
        "Settings": {
            "ShowSpeakerLabels": true,
            "MaxSpeakerLabels": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的识别发言者。

示例 5：转录存储为音频文件的医疗对话，最多两个备选转录

以下 `start-medical-transcription-job` 示例通过单个音频文件创建最多两个备选转录。每个转录具有关联的置信度。默认情况下，Amazon Transcribe 返回置信度最高的转录。您可以指定 Amazon Transcribe 返回置信度较低的其他转录。您可以在 `OutputBucketName` 参数中指定转录输出位置。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfifthfile.json
```

`myfifthfile.json` 的内容：

```
{
```

```
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    },
    "Settings":{
        "ShowAlternatives": true,
        "MaxAlternatives": 2
    }
}
```

输出：

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T19:09:18.199000+00:00",
        "CreationTime": "2020-09-21T19:09:18.171000+00:00",
        "Settings": {
            "ShowAlternatives": true,
            "MaxAlternatives": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的备选转录。

示例 6：转录医疗口述音频文件，最多具有两个备选转录

以下 `start-medical-transcription-job` 示例转录一个音频文件，并使用词汇表过滤器屏蔽任何不需要的单词。您可以在 OutputBucketName 参数中指定转录输出的位置。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysixthfile.json
```

mysixthfile.json 的内容：

```
{
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    },
    "Settings":{
          "ShowAlternatives": true,
          "MaxAlternatives": 2
    }
}
```

输出：

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T21:01:14.592000+00:00",
        "CreationTime": "2020-09-21T21:01:14.569000+00:00",
        "Settings": {
            "ShowAlternatives": true,
            "MaxAlternatives": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的备选转录。

示例 7：使用自定义词汇表更准确地转录医疗口述音频文件

以下 start-medical-transcription-job 示例转录一个音频文件，并使用您以前创建的医学自定义词汇表提高转录准确性。您可以在 OutputBucketName 参数中指定转录输出位置。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myseventhfile.json
```

mysixthfile.json 的内容：

```
{
    "MedicalTranscriptionJobName": "vocabulary-dictation-medical-transcription-
job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    },
    "Settings":{
        "VocabularyName": "cli-medical-vocab-1"
    }
}
```

输出：

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "vocabulary-dictation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T21:17:27.045000+00:00",
        "CreationTime": "2020-09-21T21:17:27.016000+00:00",
        "Settings": {
```

```
            "VocabularyName": "cli-medical-vocab-1"
        },
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的医疗自定义词汇表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考StartMedicalTranscriptionJob中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，了解如何在 AWS 代码示例存储库中进行设置和运行。

创建客户端。

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

启动医疗转录作业。

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
```

```
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and
 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the
 same region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params)
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考StartMedicalTranscriptionJob中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包开始 Amazon Transcribe 转录作业 AWS

以下代码示例演示了如何启动 Amazon Transcribe 转录作业。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建和完善自定义词汇表

- 转录音频并获取作业数据

.NET

AWS SDK for .NET

> **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。

```
    /// <summary>
    /// Start a transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon
 S3 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="languageCode">The language code of the media file, such as
 en-US.</param>
    /// <param name="vocabularyName">Optional name of a custom vocabulary.</
param>
    /// <returns>A TranscriptionJob instance with information on the new job.</
returns>
    public async Task<TranscriptionJob> StartTranscriptionJob(string jobName,
 string mediaFileUri,
        MediaFormat mediaFormat, LanguageCode languageCode, string?
 vocabularyName)
    {
        var response = await _amazonTranscribeService.StartTranscriptionJobAsync(
            new StartTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode = languageCode,
```

```
                Settings = vocabularyName != null ? new Settings()
                {
                    VocabularyName = vocabularyName
                } : null
            });
        return response.TranscriptionJob;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考StartTranscriptionJob中的。

CLI

AWS CLI

示例 1：转录音频文件

以下 `start-transcription-job` 示例转录音频文件。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfile.json
```

`myfile.json` 的内容：

```
{
    "TranscriptionJobName": "cli-simple-transcription-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发者指南》中的入门（AWS 命令行界面）。

示例 2：转录多声道音频文件

以下 `start-transcription-job` 示例转录多声道音频文件。

```
aws transcribe start-transcription-job \
    --cli-input-json file://mysecondfile.json
```

mysecondfile.json 的内容：

```
{
    "TranscriptionJobName": "cli-channelid-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "ChannelIdentification":true
    }
}
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-channelid-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
        },
        "StartTime": "2020-09-17T16:07:56.817000+00:00",
        "CreationTime": "2020-09-17T16:07:56.784000+00:00",
        "Settings": {
            "ChannelIdentification": true
        }
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的转录多声道音频。

示例 3：转录音频文件并识别不同的发言者

以下 start-transcription-job 示例转录音频文件，并在转录输出中识别发言者。

```
aws transcribe start-transcription-job \
    --cli-input-json file://mythirdfile.json
```

mythirdfile.json 的内容：

```
{
    "TranscriptionJobName": "cli-speakerid-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
    "ShowSpeakerLabels": true,
    "MaxSpeakerLabels": 2
    }
}
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-speakerid-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
        },
        "StartTime": "2020-09-17T16:22:59.696000+00:00",
        "CreationTime": "2020-09-17T16:22:59.676000+00:00",
        "Settings": {
            "ShowSpeakerLabels": true,
            "MaxSpeakerLabels": 2
        }
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的识别发言者。

示例 4：转录音频文件并在转录输出中屏蔽任何不需要的单词

以下 start-transcription-job 示例转录音频文件，并使用您以前创建的词汇表过滤器屏蔽任何不需要的单词。

```
aws transcribe start-transcription-job \
```

```
    --cli-input-json file://myfourthfile.json
```

myfourthfile.json 的内容：

```
{
    "TranscriptionJobName": "cli-filter-mask-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
          "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyFilterName": "your-vocabulary-filter",
        "VocabularyFilterMethod": "mask"
    }
}
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-filter-mask-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyFilterName": "your-vocabulary-filter",
            "VocabularyFilterMethod": "mask"
        }
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的过滤转录。

示例 5：转录音频文件并在转录输出中删除任何不需要的单词

以下 start-transcription-job 示例转录音频文件，并使用您以前创建的词汇表过滤器屏蔽任何不需要的单词。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfifthfile.json
```

myfifthfile.json 的内容：

```
{
    "TranscriptionJobName": "cli-filter-remove-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyFilterName": "your-vocabulary-filter",
        "VocabularyFilterMethod": "remove"
    }
}
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-filter-remove-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyFilterName": "your-vocabulary-filter",
            "VocabularyFilterMethod": "remove"
        }
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的过滤转录。

示例 6：使用自定义词汇表更准确地转录音频文件

以下 start-transcription-job 示例转录音频文件，并使用您以前创建的词汇表过滤器屏蔽任何不需要的单词。

```
aws transcribe start-transcription-job \
    --cli-input-json file://mysixthfile.json
```

mysixthfile.json 的内容：

```
{
    "TranscriptionJobName": "cli-vocab-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyName": "your-vocabulary"
    }
}
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-vocab-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyName": "your-vocabulary"
        }
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的过滤转录。

示例 7：识别音频文件语言并转录该文件

以下 start-transcription-job 示例转录音频文件，并使用您以前创建的词汇表过滤器屏蔽任何不需要的单词。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myseventhfile.json
```

myseventhfile.json 的内容：

```
{
    "TranscriptionJobName": "cli-identify-language-transcription-job",
    "IdentifyLanguage": true,
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
}
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-identify-language-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
        },
        "StartTime": "2020-09-18T22:27:23.970000+00:00",
        "CreationTime": "2020-09-18T22:27:23.948000+00:00",
        "IdentifyLanguage": true
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的识别语言。

示例 8：转录音频文件并编辑个人身份信息

以下 start-transcription-job 示例转录音频文件，并在转录输出中编辑任何个人身份信息。

```
aws transcribe start-transcription-job \
```

```
    --cli-input-json file://myeighthfile.json
```

myeigthfile.json 的内容：

```
{
    "TranscriptionJobName": "cli-redaction-job",
    "LanguageCode": "language-code",
    "Media": {
        "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
    },
    "ContentRedaction": {
        "RedactionOutput":"redacted",
        "RedactionType":"PII"
    }
}
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-redaction-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-25T23:49:13.195000+00:00",
        "CreationTime": "2020-09-25T23:49:13.176000+00:00",
        "ContentRedaction": {
            "RedactionType": "PII",
            "RedactionOutput": "redacted"
        }
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的自动内容编辑。

示例 9：生成个人身份信息 (PII) 经过编辑和未经过编辑的转录

以下 start-transcription-job 示例生成两个音频文件转录，一个转录中的个人身份信息经过编辑，另一个转录没有进行任何编辑。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myninthfile.json
```

myninthfile.json 的内容：

```
{
    "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
    "LanguageCode": "language-code",
    "Media": {
          "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
        },
    "ContentRedaction": {
        "RedactionOutput":"redacted_and_unredacted",
        "RedactionType":"PII"
    }
}
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-25T23:59:47.677000+00:00",
        "CreationTime": "2020-09-25T23:59:47.653000+00:00",
        "ContentRedaction": {
            "RedactionType": "PII",
            "RedactionOutput": "redacted_and_unredacted"
        }
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的自动内容编辑。

示例 10：使用您以前创建的自定义语言模型转录音频文件

以下 start-transcription-job 示例使用您以前创建的自定义语言模型转录音频文件。

```
aws transcribe start-transcription-job \
    --cli-input-json file://mytenthfile.json
```

mytenthfile.json 的内容：

```
{
    "TranscriptionJobName": "cli-clm-2-job-1",
    "LanguageCode": "language-code",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.file-extension"
    },
    "ModelSettings": {
        "LanguageModelName":"cli-clm-2"
    }
}
```

输出：

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-clm-2-job-1",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.file-
extension"
        },
        "StartTime": "2020-09-28T17:56:01.835000+00:00",
        "CreationTime": "2020-09-28T17:56:01.801000+00:00",
        "ModelSettings": {
            "LanguageModelName": "cli-clm-2"
        }
    }
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的使用自定义语言模型提高特定领域的转录准确性。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考StartTranscriptionJob中的。

Java

适用于 Java 2.x 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。

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException
 {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
```

```
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
event).transcript().results();
                    if (results.size() > 0) {
```

```
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private InputStream getStreamFromFile(String audioFileName) {
        try {
            File inputFile = new
 File(getClass().getClassLoader().getResource(audioFileName).getFile());
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
```

```
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }
```

```
        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartTranscriptionJob中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **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。

启动转录作业。

```
// Import the required AWS SDK clients and commands for Node.js
import { StartTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME",
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_LOCATION",
    // For example, "https://transcribe-demo.s3-REGION.amazonaws.com/
hello_world.wav"
  },
  OutputBucketName: "OUTPUT_BUCKET_NAME"
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartTranscriptionJobCommand(params)
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

创建客户端。

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考StartTranscriptionJob中的。

Python

SDK for Python (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。

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is
 started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when
 transcribing
                            the audio file.
    :return: Data about the job.
    """
    try:
```

```
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job
```

- 有关 API 的详细信息，请参阅适用StartTranscriptionJob于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包更新自定义 Amazon Transcribe 词汇 AWS

以下代码示例演示了如何更新自定义 Amazon Transcribe 词汇表。

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：

- 创建和完善自定义词汇表

.NET

AWS SDK for .NET

> **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。

```
    /// <summary>
    /// Update a custom vocabulary with new values. Update overwrites all
existing information.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> UpdateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.UpdateVocabularyAsync(
            new UpdateVocabularyRequest()
            {
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考UpdateVocabulary中的。

CLI

AWS CLI

使用新术语更新自定义词汇表。

以下 update-vocabulary 示例使用您提供的新术语来覆盖用于创建自定义词汇表的术语。先决条件：要替换自定义词汇表中的术语，您需要使用一个包含新术语的文件。

```
aws transcribe update-vocabulary \
    --vocabulary-file-uri s3://DOC-EXAMPLE-BUCKET/Amazon-S3-Prefix/custom-
vocabulary.txt \
    --vocabulary-name custom-vocabulary \
    --language-code language-code
```

输出：

```
{
    "VocabularyName": "custom-vocabulary",
    "LanguageCode": "language",
    "VocabularyState": "PENDING"
}
```

有关更多信息，请参阅《Amazon Transcribe 开发人员指南》中的自定义词汇表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考UpdateVocabulary中的。

Python

SDK for Python (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。

```
def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
```

```
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.",
vocabulary_name)
        raise
```

- 有关 API 的详细信息，请参阅适用UpdateVocabulary于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包进行 Amazon Transcribe 的场景 AWS

以下代码示例向您展示了如何使用软件开发工具包在 Amazon Transcribe 中实现常见场景。 AWS 这些场景向您演示如何通过在 Amazon Transcribe 中调用多个函数来完成特定任务。每个场景都包含一个指向的链接 GitHub，您可以在其中找到有关如何设置和运行代码的说明。

示例

- 使用软件开发工具包创建和完善 Amazon Transcribe 自定义词汇表 AWS
- 使用软件开发工具包使用 Amazon Transcribe 转录音频并获取工作数据 AWS

使用软件开发工具包创建和完善 Amazon Transcribe 自定义词汇表 AWS

以下代码示例展示了如何：

- 将音频文件上传到 Amazon S3。
- 运行 Amazon Transcribe 作业来转录文件并获取结果。
- 创建和完善自定义词汇表，以提高转录的准确性。
- 使用自定义词汇表运行任务并获得结果。

Python

SDK for Python (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。

转录朗读 Lewis Carroll 的《Jabberwocky》的音频文件。首先创建封装 Amazon Transcribe 操作的函数。

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is
 started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
```

```
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when
 transcribing
                            the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job



def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
```

```
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job



def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise



def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary
 processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
```

```
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.",
 vocabulary_name)
        raise
    else:
        return response



def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response =
 transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response



def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
```

```
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.",
 vocabulary_name)
        raise



def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string
 in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
 transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
```

```
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs



def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

调用包装器函数在没有自定义词汇表的情况下转录音频，然后使用不同版本的自定义词汇表来查看改进的结果。

```
def usage_demo():
    """Shows how to use the Amazon Transcribe service."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    s3_resource = boto3.resource("s3")
    transcribe_client = boto3.client("transcribe")

    print("-" * 88)
    print("Welcome to the Amazon Transcribe demo!")
    print("-" * 88)

    bucket_name = f"jabber-bucket-{time.time_ns()}"
    print(f"Creating bucket {bucket_name}.")
    bucket = s3_resource.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": transcribe_client.meta.region_name
        },
    )
    media_file_name = ".media/Jabberwocky.mp3"
    media_object_key = "Jabberwocky.mp3"
    print(f"Uploading media file {media_file_name}.")
    bucket.upload_file(media_file_name, media_object_key)
    media_uri = f"s3://{bucket.name}/{media_object_key}"

    job_name_simple = f"Jabber-{time.time_ns()}"
    print(f"Starting transcription job {job_name_simple}.")
    start_job(
        job_name_simple,
        f"s3://{bucket_name}/{media_object_key}",
        "mp3",
        "en-US",
        transcribe_client,
    )
    transcribe_waiter = TranscribeCompleteWaiter(transcribe_client)
    transcribe_waiter.wait(job_name_simple)
    job_simple = get_job(job_name_simple, transcribe_client)
    transcript_simple = requests.get(
        job_simple["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_simple['jobName']}:")
    print(transcript_simple["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Creating a custom vocabulary that lists the nonsense words to try to "
        "improve the transcription."
    )
```

```
    vocabulary_name = f"Jabber-vocabulary-{time.time_ns()}"
    create_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        phrases=[
            "brillig",
            "slithy",
            "borogoves",
            "mome",
            "raths",
            "Jub-Jub",
            "frumious",
            "manxome",
            "Tumtum",
            "uffish",
            "whiffling",
            "tulgey",
            "thou",
            "frabjous",
            "callooh",
            "callay",
            "chortled",
        ],
    )
    vocabulary_ready_waiter = VocabularyReadyWaiter(transcribe_client)
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocabulary_list = f"Jabber-vocabulary-list-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocabulary_list}.")
    start_job(
        job_name_vocabulary_list,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
        vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocabulary_list)
    job_vocabulary_list = get_job(job_name_vocabulary_list, transcribe_client)
    transcript_vocabulary_list = requests.get(
        job_vocabulary_list["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocabulary_list['jobName']}:")
```

```
    print(transcript_vocabulary_list["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Updating the custom vocabulary with table data that provides additional
"
        "pronunciation hints."
    )
    table_vocab_file = "jabber-vocabulary-table.txt"
    bucket.upload_file(table_vocab_file, table_vocab_file)
    update_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        table_uri=f"s3://{bucket.name}/{table_vocab_file}",
    )
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocab_table = f"Jabber-vocab-table-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocab_table}.")
    start_job(
        job_name_vocab_table,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
        vocabulary_name=vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocab_table)
    job_vocab_table = get_job(job_name_vocab_table, transcribe_client)
    transcript_vocab_table = requests.get(
        job_vocab_table["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocab_table['jobName']}:")
    print(transcript_vocab_table["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print("Getting data for jobs and vocabularies.")
    jabber_jobs = list_jobs("Jabber", transcribe_client)
    print(f"Found {len(jabber_jobs)} jobs:")
    for job_sum in jabber_jobs:
        job = get_job(job_sum["TranscriptionJobName"], transcribe_client)
        print(
            f"\t{job['TranscriptionJobName']}, {job['Media']['MediaFileUri']}, "
```

```
            f"{job['Settings'].get('VocabularyName')}"
        )

    jabber_vocabs = list_vocabularies("Jabber", transcribe_client)
    print(f"Found {len(jabber_vocabs)} vocabularies:")
    for vocab_sum in jabber_vocabs:
        vocab = get_vocabulary(vocab_sum["VocabularyName"], transcribe_client)
        vocab_content = requests.get(vocab["DownloadUri"]).text
        print(f"\t{vocab['VocabularyName']} contents:")
        print(vocab_content)

    print("-" * 88)
    print("Deleting demo jobs.")
    for job_name in [job_name_simple, job_name_vocabulary_list,
job_name_vocab_table]:
        delete_job(job_name, transcribe_client)
    print("Deleting demo vocabulary.")
    delete_vocabulary(vocabulary_name, transcribe_client)
    print("Deleting demo bucket.")
    bucket.objects.delete()
    bucket.delete()
    print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅 AWS SDK for Python (Boto3) API Reference 中的以下主题。
  - CreateVocabulary
  - DeleteTranscriptionJob
  - DeleteVocabulary
  - GetTranscriptionJob
  - GetVocabulary
  - ListVocabularies
  - StartTranscriptionJob
  - UpdateVocabulary

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包使用 Amazon Transcribe 转录音频并获取工作数据 AWS

以下代码示例显示了如何：

- 使用 Amazon Transcribe 开始转录作业。
- 等待作业完成。
- 获取存储转录的 URI。

有关更多信息，请参阅 Amazon Transcribe 入门。

Java

适用于 Java 2.x 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。

转录 PCM 文件。

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {
```

```
        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
```

```
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
```

```
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
```

```
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

转录来自计算机麦克风的流音频。

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
```

```
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
            throws URISyntaxException, ExecutionException, InterruptedException,
LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException
{

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }
```

```
    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private InputStream getStreamFromFile(String audioFileName) {
        try {
            File inputFile = new
 File(getClass().getClassLoader().getResource(audioFileName).getFile());
            InputStream audioStream = new FileInputStream(inputFile);
```

```
            return audioStream;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
```

```
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
```

```
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的以下主题。
  - GetTranscriptionJob
  - StartTranscriptionJob

Python

SDK for Python (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。

```
import time
import boto3


def transcribe_file(job_name, file_uri, transcribe_client):
    transcribe_client.start_transcription_job(
        TranscriptionJobName=job_name,
        Media={"MediaFileUri": file_uri},
        MediaFormat="wav",
        LanguageCode="en-US",
    )
```

```
    max_tries = 60
    while max_tries > 0:
        max_tries -= 1
        job =
 transcribe_client.get_transcription_job(TranscriptionJobName=job_name)
        job_status = job["TranscriptionJob"]["TranscriptionJobStatus"]
        if job_status in ["COMPLETED", "FAILED"]:
            print(f"Job {job_name} is {job_status}.")
            if job_status == "COMPLETED":
                print(
                    f"Download the transcript from\n"
                    f"\t{job['TranscriptionJob']['Transcript']
['TranscriptFileUri']}."
                )
            break
        else:
            print(f"Waiting for {job_name}. Current status is {job_status}.")
        time.sleep(10)


def main():
    transcribe_client = boto3.client("transcribe")
    file_uri = "s3://test-transcribe/answer2.wav"
    transcribe_file("Example-job", file_uri, transcribe_client)


if __name__ == "__main__":
    main()
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的以下主题。
  - GetTranscriptionJob
  - StartTranscriptionJob

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

使用软件开发工具包的 Amazon Tran AWS scribe 的跨服务示例

以下示例应用程序使用 AWS 软件开发工具包将 Amazon Transcribe 与其他应用程序组合在一起。AWS 服务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如何设置和运行应用程序的说明。

示例

- 构建 Amazon Transcribe 应用程序
- 构建 Amazon Transcribe 流式传输应用程序
- 使用 AWS SDK 将文本转换为语音并转换回文本

构建 Amazon Transcribe 应用程序

以下代码示例显示了如何使用 Amazon Transcribe 在浏览器中转录并显示录音。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

创建一个使用 Amazon Transcribe 在浏览器中转录和显示录音的应用程序。该应用程序使用两个 Amazon Simple Storage Service（Amazon S3）桶，一个用于托管应用程序代码，另一个用于存储转录。该应用程序使用 Amazon Cognito 用户池对您的用户进行身份验证。经过身份验证的用户拥有 AWS Identity and Access Management (IAM) 访问所需 AWS 服务的权限。

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。

该示例也可在 AWS SDK for JavaScript v3 开发人员指南中找到。

本示例中使用的服务

- Amazon Cognito Identity
- Amazon S3
- Amazon Transcribe

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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。

构建 Amazon Transcribe 流式传输应用程序

以下代码示例展示如何构建可实时录制、转录与翻译实时音频，并通过电子邮件发送结果的应用程序。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示了如何使用 Amazon Transcribe 构建可实时录制、转录与翻译实时音频，并通过 Amazon Simple Email Service (Amazon SES) 以电子邮件发送结果的应用程序。

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 Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 结合使用。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 AWS SDK 将文本转换为语音并转换回文本

以下代码示例展示了如何：

- 使用 Amazon Polly 将纯文本 (UTF-8) 输入文件合成为音频文件。
- 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）输入文件合成为音频文件，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，请参阅将此服务与 AWS SDK 结合使用。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 Transcribe 中的安全性

AWS 十分重视云安全性。作为 AWS 客户，您会从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性是 AWS 和您的共同责任。责任共担模型将其描述为云的安全性和云中的安全性：

- 云的安全性：AWS负责保护在中运行AWS服务的基础设施AWS Cloud。 AWS还向您提供可安全使用的服务。作为AWS 合规性计划的一部分，第三方审计人员将定期测试和验证安全性的有效性。要了解适用于 Amazon Transcribe 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性：您的责任是由使用的AWS服务决定的。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。

此文档将帮助您了解如何在使用 Amazon Transcribe 时应用责任共担模型 以下主题说明如何配置 Amazon Transcribe 以实现您的安全性和合规性目标。您还将了解如何使用其他AWS服务来监控和保护Amazon Transcribe资源。

主题

适用于 Amazon Transcribe 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一项 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）来使用 Amazon Transcribe 资源。IAM 是一项无需额外费用即可使用的 AWS 服务。

主题

受众

使用 AWS Identity and Access Management (IAM) 的方式因您可以在 Amazon Transcribe 中执行的操作而异。

服务用户 - 如果使用 Amazon Transcribe 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。当您使用更多 Amazon Transcribe 特征来完成工作时，您可能需要额外权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amazon Transcribe 中的特征，请参阅 对 Amazon Transcribe 身份和访问进行故障排除。

服务管理员 - 如果您在公司负责管理 Amazon Transcribe 资源，则您可能具有 Amazon Transcribe 的完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon Transcribe 特征和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon Transcribe 搭配使用的更多信息，请参阅Amazon Transcribe 如何与 IAM 协同工作。

IAM 管理员 - 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amazon Transcribe 的访问权限的详细信息。要查看您可在 IAM 中使用的 Amazon Transcribe 基于身份的策略示例，请参阅 Amazon Transcribe 基于身份的策略示例。

使用身份进行身份验证

身份验证是使用身份凭证登录 AWS 的方法。您必须作为 AWS 账户根用户、IAM 用户或通过担任 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center(IAM Identity Center) 用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。当您以联合身份登录时，管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合身份验证访问 AWS 时，您就是在间接担任角色。

根据用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录到 AWS 的更多信息，请参阅《AWS 登录 用户指南》中的如何登录到 AWS 账户。

如果您以编程方式访问 AWS，则 AWS 将提供软件开发工具包 (SDK) 和命令行界面 (CLI)，以便使用您的凭证以加密方式签署您的请求。如果您不使用 AWS 工具，则必须自行对请求签名。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的 多重身份验证 和《IAM 用户指南》中的 在 AWS 中使用多重身份验证 (MFA)。

AWS 账户 根用户

创建 AWS 账户 时，最初使用的是一个对账户中所有 AWS 服务 和资源拥有完全访问权限的登录身份。此身份称为 AWS 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实操，要求人类用户（包括需要管理员访问权限的用户）结合使用联合身份验证和身份提供程序，以使用临时凭证来访问 AWS 服务。

联合身份是来自企业用户目录、网络身份提供程序、AWS Directory Service、Identity Center 目录的用户，或任何使用通过身份源提供的凭证来访问 AWS 服务 的用户。当联合身份访问 AWS 账户时，他们担任角色，而角色提供临时凭证。

要集中管理访问权限，我们建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到自己的身份源中的一组用户和组以跨所有 AWS 账户 和应用程序使用。有关 IAM Identity Center 的信息，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 AWS 账户内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用群组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用群组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人担任。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是 AWS 账户 中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS Management Console 中暂时担任 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义网址以代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户或角色可代入 IAM 角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户存取权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为座席）。要了解用于跨账户存取的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 - 某些 AWS 服务使用其他 AWS 服务中的特征。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话：当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的政策详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而担任的 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。

- 服务相关角色 - 服务相关角色是与 AWS 服务关联的一种服务角色。服务可以担任代表您执行操作的角色。服务相关角色显示在您的 AWS 账户中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 - 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 AWS 身份或资源，以控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。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。

默认情况下，用户和角色没有权限。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。然后，管理员可以向角色添加 IAM 策略，并且用户可以担任角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 AWS Management Console、AWS CLI 或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户群组或角色）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》中的创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、群组或角色中。托管策略是可以附加到AWS 账户中的多个用户、组和角色的独立策略。托管式策略包括 AWS 托管式策略和客户管理型策略。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。

基于资源的策略

基于资源的策略是附加到资源的 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。主体可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的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来自 IAM 的 AWS 托管策略。

访问控制列表 (ACL)

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，尽管它们不使用 JSON 策略文档格式。

Amazon S3、AWS WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表 (ACL) 概览。

其他策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型授予的最大权限。

- 权限边界 - 权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可以为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，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) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 账户。如果在组织内启用了所有特征，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 - 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。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 AWS 如何确定在涉及多种策略类型时是否允许请求，请参阅《IAM 用户指南》中的策略评估逻辑。

Amazon Transcribe 如何与 IAM 协同工作

在使用 IAM 管理对 Amazon Transcribe 的访问之前，您应该了解哪些 IAM 功能可用于 Amazon Transcribe。

可以与 Amazon Transcribe 搭配使用的 IAM 特征

IAM 功能	Amazon Transcribe支持
基于身份的策略	可以
基于资源的策略	不可以
策略操作	可以
策略资源	可以
策略条件键（特定于服务）	可以
ACL	不可以
ABAC（策略中的标签）	部分
临时凭证	可以
主体权限	可以
服务角色	可以
服务相关角色	不可以

要大致了解 Amazon Transcribe 和其它 AWS 服务如何与大多数 IAM 特征一起使用，请参阅《IAM 用户指南》中的与 IAM 一起使用的 AWS 服务。

适用于 Amazon Transcribe 的基于身份的策略

支持基于身份的策略	可以

基于身份的策略是可附加到身份（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》中的创建 IAM 策略。

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。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。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。

适用于 Amazon Transcribe 的基于身份的策略示例

要查看 Amazon Transcribe 基于身份的策略的示例，请参阅Amazon Transcribe 基于身份的策略示例。

Amazon Transcribe 内基于资源的策略

支持基于资源的策略	不可以

基于资源的策略是附加到资源的 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。主体可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户存取，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。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当主体和资源处于不同的 AWS 账户 中时，则信任账户中的 IAM 管理员还必须授予主体实体（用户或角色）对资源的访问权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

适用于 Amazon Transcribe 的策略操作

支持策略操作	可以

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。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。策略操作通常与相关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看 Amazon Transcribe 操作的列表，请参阅《服务授权参考》中的 Amazon Transcribe 定义的操作。

Amazon Transcribe 中的策略操作在操作前使用 `transcribe` 前缀。要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "transcribe:action1",
      "transcribe:action2"
]
```

也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `List` 开头的所有操作，包括以下操作：

```
"Action": "transcribe:List*"
```

要查看 Amazon Transcribe 基于身份的策略的示例，请参阅Amazon Transcribe 基于身份的策略示例。

Amazon Transcribe 的策略资源

支持策略资源	可以

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。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。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称 (ARN)指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 Amazon Transcribe 资源类型及其 ARN 的列表，请参阅《服务授权参考》中的 Amazon Transcribe 定义的资源。要了解可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon Transcribe 定义的操作。

要查看 Amazon Transcribe 基于身份的策略的示例，请参阅Amazon Transcribe 基于身份的策略示例。

Amazon Transcribe 的策略条件键

支持特定于服务的策略条件键	可以

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。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。

在 Condition 元素（或 Condition 块）中，您可以指定语句生效的条件。Condition 元素是可选的。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。

如果在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个密钥，则 AWS 使用逻辑 AND 运算评估它们。如果您要为单个条件密钥指定多个值，则 AWS 使用逻辑 OR 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

您也可以在指定条件时使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM policy 元素：变量和标签。

AWS 支持全局条件键和特定于服务的条件键。要查看所有 AWS 全局条件键，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

要查看 Amazon Transcribe 条件键列表，请参阅《服务授权参考》中的 Amazon Transcribe 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 Transcribe 定义的操作。

要查看 Amazon Transcribe 基于身份的策略的示例，请参阅Amazon Transcribe 基于身份的策略示例。

Amazon Transcribe 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，尽管它们不使用 JSON 策略文档格式。

带有 Amazon Transcribe 的 ABAC

支持 ABAC（策略中的标签）	部分

基于属性的访问权限控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在 AWS 中，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）以及 AWS 资源。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件密钥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密钥，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密钥，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的什么是 ABAC？。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)。

有关标记 Amazon Transcribe 资源的更多信息，请参阅为资源添加标签。有关基于标签的访问控制的更多详细信息，请参阅使用标签控制对 AWS 资源的访问。

将临时凭证用于 Amazon Transcribe

支持临时凭证	可以

某些 AWS 服务 在使用临时凭证登录时无法正常工作。有关更多信息，包括 AWS 服务 与临时凭证配合使用，请参阅《IAM 用户指南》中的使用 IAM 的 AWS 服务。

如果您不使用用户名和密码而用其他方法登录到AWS Management Console，则使用临时凭证。例如，当您使用贵公司的单点登录（SSO）链接访问 AWS 时，该过程将自动创建临时凭证。当您以用户身份登录控制台，然后切换角色时，还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的切换到角色（控制台）。

您可以使用 AWS CLI 或者 AWS API 创建临时凭证。之后，您可以使用这些临时凭证访问 AWS。AWS 建议您动态生成临时凭证，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

Amazon Transcribe 的跨服务主体权限

支持转发访问会话（FAS）	可以

当您使用 IAM 用户或角色在AWS中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详细信息，请参阅转发访问会话。

Amazon Transcribe 的服务角色

支持服务角色	可以

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 Transcribe 的功能。只有在 Amazon Transcribe 提供指导时，才能编辑服务角色。

Amazon Transcribe 的服务相关角色

支持服务相关角色	不可以

服务相关角色是一种与 AWS 服务相关的服务角色。服务可以担任代表您执行操作的角色。服务相关角色显示在您的 AWS 账户中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

Amazon Transcribe 不支持服务相关角色。

有关创建或管理其它服务的服务相关角色的详细信息，请参阅使用 IAM 的 AWS 服务。在表中查找服务相关角色列中包含 Yes 的服务。选择 Yes 链接以查看该服务的服务相关角色文档。

防止跨服务混淆代理

困惑不解的副手是被其他实体强迫采取行动的实体（服务或账户）。这种类型的模拟可以跨账户和跨服务发生。

为了防止人数据的工具，AWS提供可帮助您保护所有服务的工具，而这些服务委托人有权限访问账户中的资源AWS 账户。本节重点介绍专门针对的跨服务混乱副手预防措施Amazon Transcribe；但是，你可以在IAM用户指南的困惑副手问题部分了解有关该主题的更多信息。

为了限制授IAM予的访问您的资源的权限，我们建议在资源策略`aws:SourceAccount`中使用全局条件上下文密钥`aws:SourceArn`。Amazon Transcribe

如果使用这两个全局条件上下文键，并且该`aws:SourceArn`值包含AWS 账户 ID，则AWS 账户在同一策略语句中使用时，`aws:SourceArn`必须使用相同的AWS 账户 ID。`aws:SourceAccount`

如果您只希望将一个资源与跨服务访问相关联，请使用 `aws:SourceArn`。如果您想将其中任何资源 AWS 账户与跨服务访问相关联，请使用`aws:SourceAccount`。

> **Note**
>
> 防范混淆代理问题最有效的方法是使用全`aws:SourceArn`局条件上下文键和资源的完整 ARN。如果您不知道完整 ARN，或者您正在指定多个资源，请针对 ARN 未知部分使用带有通配符 (*) 的`aws:SourceArn`全局上下文条件键。例如，`arn:aws:transcribe::123456789012:*`。

有关显示如何防止副手问题混淆的代入角色策略的示例，请参阅混淆代理问题防范。

Amazon Transcribe 基于身份的策略示例

默认情况下，用户和角色没有创建或修改 Amazon Transcribe 资源的权限。他们也无法使用 AWS Management Console、AWS Command Line Interface（AWS CLI）或 AWS API 执行任务。要用户用户在其需要的资源上执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

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 policy。

有关 Amazon Transcribe nitbite 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 Transcribe nitbite 的操作、资源和条件键。

主题

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon Transcribe 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，请遵循以下准则和建议：

- AWS 托管策略及转向最低权限许可入门 - 要开始向用户和工作负载授予权限，请使用 AWS 托管策略来为许多常见使用场景授予权限。您可以在 AWS 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 AWS 客户管理型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 policy 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 policy 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。如果通过特定 AWS 服务（例如 AWS CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。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 policy，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 policy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 IAM Acess Analyzer 策略验证。
- Require multi-factor authentication（MFA）[需要多重身份验证（MFA）] – 如果您所处的场景要求您的 AWS 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。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》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 AWS Management Console

要访问 Amazon Transcribe ronbite 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AWS 账户中的 Amazon Transcribe 资源的详细信息。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。

对于只需要调用 AWS CLI 或 AWS API 的用户，无需为其提供最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保实体（用户和角色）可以使用 AWS Management Console，请为其AWS附加以下托管策略之一。

- `AmazonTranscribeFullAccess`：授予创建、读取、更新、删除和运行所有Amazon Transcribe资源的完全访问权限。它还允许访问Amazon S3存储段名称`transcribe`中包含的存储桶。

- `AmazonTranscribeReadOnlyAccess`：授予Amazon Transcribe资源的只读访问权限，以便您可以获取和列出转录作业和自定义词汇表。

> **Note**
>
> 您可以通过登录到，然后按策略名称进行搜索IAMAWS Management Console，查看托管权限策略。搜索 “transcribe” 会返回上面列出的两个策略（AmazonTranscribeReadOnly和AmazonTranscribeFullAccess）。

您还可以创建自己的自定义 IAM 策略，以授予执行 Amazon Transcribe API 操作的相关权限。您可以将这些自定义策略附加到需要这些权限的实体。

创建IAM角色所需的权限

如果您创建要调IAM用的角色Amazon Transcribe，则该角色必须具有访问Amazon S3存储段的权限。如果适用，还KMS key必须使用来加密存储段的内容。有关策略的示例，请参阅以下部分。

信任政策

您用来提出转录请求的IAM实体必须具有允许担任该角色Amazon Transcribe的信任策略。使用以下Amazon Transcribe信任策略。请注意，如果您在启用了通话后分析的情况下发出实时呼叫分析请求，则必须使用 “实时呼叫分析的信任政策”。

的信任政策Amazon Transcribe

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "transcribe.amazonaws.com"
        ]
      },
      "Action": [
        "sts:AssumeRole"
      ],
      "Condition": {
```

```
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "StringLike": {
          "aws:SourceArn": "arn:aws:transcribe:us-west-2:111122223333:*"
        }
      }
    }
  ]
}
```

实时呼叫分析的信任政策

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "transcribe.streaming.amazonaws.com"
        ]
      },
      "Action": [
        "sts:AssumeRole"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "StringLike": {
          "aws:SourceArn": "arn:aws:transcribe:us-west-2:111122223333:*"
        }
      }
    }
  ]
}
```

Amazon S3输入存储桶策略

以下策略授予IAM角色访问指定输入存储桶中的文件的权限。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-INPUT-BUCKET",
            "arn:aws:s3:::DOC-EXAMPLE-INPUT-BUCKET/*"
        ]
    }
}
```

Amazon S3输出存储桶策略

以下策略授予IAM角色将文件写入指定输出存储桶的权限。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-OUTPUT-BUCKET/*"
        ]
    }
}
```

Amazon S3加密密钥所需的权限

如果您使用加密Amazon S3存储桶，请在KMS key策略中包括以下内容。KMS key这将向 Amazon Transcribe 提供对存储桶内容的访问权。有关允许访问的更多信息KMS keys，请参阅《AWS KMS开发者指南》KMS key中的 “AWS 账户允许外部人员访问”。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
      },
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-west-2:111122223333:key/KMS-Example-KeyId"
    }
  ]
}
```

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完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

AWS KMS加密上下文策略

以下策略授予IAM角色 “ExampleRole” 对此特定角色使用 “AWS KMS解密” 和 “加密” 操作的权限KMS key。此策略仅适用于具有至少一个加密上下文对的请求，在本例中为 “color:indigoBlue”。有关AWS KMS加密上下文的更多信息，请参阅AWS KMS 加密上下文。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Principal": {
              "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
          },
          "Action": [
              "kms:Decrypt",
              "kms:DescribeKey",
              "kms:Encrypt",
              "kms:GenerateDataKey*",
              "kms:ReEncrypt*"
          ],
          "Resource": "*",
          "Condition": {
              "StringEquals": {
                  "kms:EncryptionContext:color":"indigoBlue"
              }
           }
        }
   ]
}
```

混淆代理问题防范

以下是代入角色政策的示例，该政策显示了如何aws:SourceAccount使用aws:SourceArn和Amazon Transcribe来防止副手问题变得混乱。有关预防混乱的副手的更多信息，请参阅防止跨服务混淆代理。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "transcribe.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "StringLike": {
          "aws:SourceArn": "arn:aws:transcribe:us-west-2:111122223333:*"
        }
      }
    }
  ]
}
```

基于标签查看转录任务

您可以在基于身份的策略中使用条件，以便基于标签控制对 Amazon Transcribe 资源的访问。此示例演示了如何创建允许查看转录任务的策略。但是，仅当转录任务标签Owner具有值时，才会授予权限。此策略还授予有计划地使用中使用此操作的必要权限AWS Management Console。

您可以将该策略附加到您账户中的IAM实体。如果名为的角色test-role尝试查看转录作业，则必须对转录作业进行标记Owner=test-role或owner=test-role（条件键名称不区分大小写），否则将拒绝他们访问。有关更多信息，请参阅《IAM用户指南》中的IAM JSON 策略元素：条件。

有关添加标签的更多信息Amazon Transcribe，请参阅为资源添加标签。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListTranscriptionJobsInConsole",
            "Effect": "Allow",
            "Action": "transcribe:ListTranscriptionJobs",
```

```
            "Resource": "*"
        },
        {
            "Sid": "ViewTranscriptionJobsIfOwner",
            "Effect": "Allow",
            "Action": "transcribe:GetTranscriptionJobs",
            "Resource": "arn:aws:transcribe:*:*:transcription-job/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

对 Amazon Transcribe 身份和访问进行故障排除

使用以下信息可诊断和解决在使用 Amazon Transcribe 和 AWS Identity and Access Management (IAM) 时可能遇到的常见问题。

主题

- 我无权在 Amazon Transcribe 中执行操作
- 我无权执行 iam:PassRole
- 我希望允许我的 AWS 账户以外的人访问我的 Amazon Transcribe 资源

我无权在 Amazon Transcribe 中执行操作

如果您收到一个错误，表明您无权执行某个操作，则必须更新策略以允许您执行该操作。

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 transcribe:*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:
 transcribe: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 transcribe:*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。管理员是向您提供登录凭证的人。

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 Transcribe。

有些 AWS 服务允许您将现有角色传递到该服务，而不是创建新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 Transcribe 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。管理员是向您提供登录凭证的人。

我希望允许我的 AWS 账户以外的人访问我的 Amazon Transcribe 资源

您可以创建一个角色，以便其它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，可以担任角色。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 Transcribe 是否支持这些功能，请参阅 Amazon Transcribe 如何与 IAM 协同工作。
- 要了解如何为您拥有的 AWS 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 AWS 账户 中的 IAM 用户提供访问权限。
- 要了解如何为第三方AWS 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的 AWS 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）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 角色与基于资源的策略有何不同。

Amazon Transcribe 中的数据保护

AWS 责任共担模式适用于 Amazon Transcribe 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的博客文章 AWS Shared Responsibility Model and GDPR。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。我们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)。
- 使用 SSL/TLS 与 AWS 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务 中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括使用控制台、API、AWS CLI 或 AWS SDK 处理 Amazon Transcribe 或其他 AWS 服务时。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，我们强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

互联网络流量隐私

Amazon Transcribe 的 Amazon Virtual Private Cloud (Amazon VPC) 终端节点是 VPC 中的一个逻辑实体，只允许连接到 Amazon Transcribe。Amazon VPC 将请求路由到 Amazon Transcribe 并将响应路由回 VPC。有关更多信息，请参阅 AWS PrivateLink 概念。有关将 Amazon VPC 终端节点与 Amazon Transcribe 结合使用的信息，请参阅Amazon Transcribe 和接口 VPC 端点 (AWS PrivateLink)。

数据加密

数据加密指在传输中的数据和静态中的数据保护。KMS keys在传输过程中，您可以使用Amazon S3托管密钥或静态密钥以及标准传输层安全 (TLS) 来保护数据。

静态加密

Amazon Transcribe使用默认Amazon S3密钥 (SSE-S3) 对Amazon S3存储段中的记录进行服务器端加密。

使用该StartTranscriptionJob操作时，您可以指定自己的操作KMS key来加密转录作业的输出。

Amazon Transcribe 使用已采用默认密钥加密的 Amazon EBS 卷。

传输中加密

Amazon Transcribe 使用 TLS 1.2 和 AWS 证书加密传输中的数据。这包括流媒体转录。

密钥管理

Amazon Transcribe可KMS keys为您的数据提供增强的加密。使用Amazon S3，您可以在创建转录任务时对输入媒体进行加密。与的集成AWS KMS允许对StartTranscriptionJob请求的输出进行加密。

如果您未指定KMS key，则转录作业的输出将使用默认Amazon S3密钥 (SSE-S3) 进行加密。

有关更多信息AWS KMS，请参阅《AWS Key Management Service开发者指南》。

使用密钥管理AWS Management Console

要对转录任务的输出进行加密，可以选择KMS key对发出请求的使用 aAWS 账户，也可以选择使用KMS key来自另一个请求的AWS 账户。

如果您未指定KMS key，则转录作业的输出将使用默认Amazon S3密钥 (SSE-S3) 进行加密。

要启用输出加密，请执行以下操作：

1. 在输出数据下，选择加密。

2. 选择KMS key是来自AWS 账户你当前正在使用的，还是来自不同的AWS 账户。如果您想使用其他密钥AWS 账户，请从 KMS keyID 中选择密钥。如果您使用的是其他密钥AWS 账户，则必须输入密钥的 ARN。要使用其他密钥AWS 账户，调用者必须具有kms:Encrypt访问的权限KMS key。有关更多信息，请参阅创建密钥策略。

使用 API 管理密钥

要在 API 中使用输出加密，必须KMS key使用StartCallAnalyticsJobStartMedicalTranscriptionJob、或StartTranscriptionJob操作的OutputEncryptionKMSKeyId参数指定您的。

如果使用位于当前密钥AWS 账户，则可以通过以下四种KMS key方式之一指定您的密钥：

1. 使用KMS key ID 本身。例如，1234abcd-12ab-34cd-56ef-1234567890ab。
2. 使用别名作为KMS key ID。例如，alias/ExampleAlias。
3. 使用 Amazon 资源名称（ARNKMS key）。例如，arn:aws:kms:region:account-ID:key/1234abcd-12ab-34cd-56ef-1234567890ab。
4. 使用 ARN 作为KMS key别名。例如，arn:aws:kms:region:account-ID:alias/ExampleAlias。

如果使用的密钥与当前密钥AWS 账户不同AWS 账户，则可以通过以下两种KMS key方式之一指定您的密钥：

1. 使用 ARN 作为KMS key ID。例如，arn:aws:kms:region:account-ID:key/1234abcd-12ab-34cd-56ef-1234567890ab。
2. 使用 ARN 作为KMS key别名。例如，arn:aws:kms:region:account-ID:alias/ExampleAlias。

请注意，发出请求的实体必须拥有使用指定密钥的权限KMS key。

AWS KMS 加密上下文

AWS KMS加密上下文是纯文本、非密钥:值对的映射。此地图表示其他经过身份验证的数据，称为加密上下文对，可为您的数据提供额外的安全保护。 Amazon Transcribe需要对称加密密钥才能将转录输出加密到客户指定的Amazon S3存储桶中。要了解更多信息，请参阅中的非对称密钥AWS KMS。

创建加密上下文对时，请勿包含敏感信息。加密上下文不是秘密的，它在CloudTrail日志中以纯文本形式可见（因此您可以使用它来识别和分类您的加密操作）。

您的加密上下文对可以包含特殊字符，例如下划线 (_)、短划线 (-)、斜杠 (/,\) 和冒号 (:)。

> **Tip**
>
> 将加密上下文对中的值与正在加密的数据关联起来可能很有用。尽管不是必需的，但我们建议您使用与加密内容相关的非敏感元数据，例如文件名、标题值或未加密的数据库字段。

要在 API 中使用输出加密，请在`StartTranscriptionJob`操作中设置`KMSEncryptionContext`参数。为了为输出加密操作提供加密上下文，`OutputEncryptionKMSKeyId`参数必须引用对称KMS key ID。

您可以使用AWS KMS条件密钥和IAM策略，KMS key根据加密操作请求中使用的加密上下文控制对对称加密的访问权限。有关加密上下文策略的示例，请参见AWS KMS加密上下文策略。

使用加密上下文是可选的，但我们建议您这样做。有关更多信息，请参阅加密上下文。

选择不使用您的数据来改善服务

默认情况下，Amazon Transcribe存储和使用其处理的语音输入来开发服务并不断改善您的体验。您可以使用选择退出政策，选择AWS Organizations不将您的内容用于开发和改进Amazon Transcribe。有关如何退出的信息，请参阅 AI 服务选择退出策略。

监控 Amazon Transcribe

监控是保持 Amazon Transcribe 和您的其他 AWS 解决方案的可靠性、可用性和性能的重要方面。AWS 提供了以下一些监控工具来监控 Amazon Transcribe、在出现错误时进行报告并适时自动采取措施.

- Amazon CloudWatch 可实时监控您的 AWS 资源以及您在 AWS 上运行的应用程序。您可以收集和跟踪指标，创建自定义的控制面板，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以具有Amazon EC2实例的CloudWatch跟踪 CPU 使用率或其他指标并且在需要时自动启动新实例。
- Amazon CloudWatch Logs可以监控、存储和访问来自Amazon EC2实例CloudTrail、或其他来源的日志文件。 CloudWatch Logs可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。
- AWS CloudTrail捕获由您或代表该用户发出的 API 调用AWS 账户和相关事件，并将日志文件传输到您指定的Amazon S3存储桶。您可以标识哪些用户和账户调用了 AWS、从中发出调用的源 IP 地址以及调用的发生时间。

有关更多信息，请参阅 Amazon CloudWatch 用户指南。

Amazon EventBridge是一项无服务器服务，它使用事件将应用程序组件连接在一起，使您可以更轻松地构建可扩展的事件驱动应用程序。 EventBridge从您自己的应用程序、软件即服务 (SaaS) 应用程序和服务传输实时数据流，然后AWS将该数据路由到诸如之类的目标发出了什么请求发出Lambda。您可以监控服务中发生的事件，并构建事件驱动的架构。有关更多信息，请参阅 Amazon EventBridge 用户指南。

主题

使用 Amazon CloudWatch 监控 Amazon Transcribe

您可以使用 CloudWatch 监控 Amazon Transcribe，此工具可收集原始数据，并将数据处理为便于读取的近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 CloudWatch 用户指南。

将Amazon CloudWatch指标和维度与Amazon Transcribe

Amazon Transcribe支持CloudWatch指标和维度，这些数据可以帮助您监控绩效。支持的指标类别包括流量、错误、数据传输和与转录作业相关的延迟。支持的指标位于 AWS/Transcribe 命名空间CloudWatch中。

> **Note**
>
> CloudWatch监控指标是免费的，不计入CloudWatch服务配额。

有关CloudWatch指标的更多信息，请参阅使用Amazon CloudWatch指标。

使用 AWS CloudTrail 监控 Amazon Transcribe

Amazon Transcribe与AWS CloudTrail提供Amazon Transcribe由AWS Identity and Access Management (IAM) 用户或角色或服务执行的操作记录的AWS服务集成。 CloudTrail捕获的所有 API 调用Amazon Transcribe。这包括以事件方式对Amazon Transcribe API 的调用AWS Management Console和代码调用。通过创建跟踪，您可以使CloudTrail事件持续传送到Amazon S3存储桶（包括的事件）。Amazon Transcribe如果您不创建跟踪，则仍可在 in Event history (事件历史记录)CloudTrailAWS Management Console 中查看最新事件。通过使用收集的信息CloudTrail，您可以查看向发出了什么请求Amazon Transcribe、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解更多信息CloudTrail，请参阅AWS CloudTrail用户指南。

Amazon Transcribe 和 CloudTrail

在您创建 AWS 账户 账户时，即针对该账户启用了 CloudTrail。在中发生活动时Amazon Transcribe，该活动将记录在事件中，并与其他CloudTrail事件一同AWS 服务保存在CloudTrail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。

要持续记录您中的事件（包括的事件）AWS 账户Amazon Transcribe，请创建跟踪。跟踪是一种配置，可用于CloudTrail将事件作为日志文件传送到指定Amazon S3存储桶。 CloudTrail日志文件包含一个或多个日志条目。事件 表示来自任何源的单个请求。它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 日志文件不是公有 API 调用的有序堆栈跟踪，因此它们不会以任何特定顺序显示。

默认情况下，在中创建跟踪时，此跟踪时AWS Management Console，此跟踪应用于所有发出AWS 区域。此跟踪在AWS分区AWS 区域中记录所有事件，并将日志文件传送至您指定的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：

- 创建跟踪概览
- CloudTrail 支持的服务和集成

- 为 CloudTrail 配置 Amazon SNS 通知
- 接收多个区域中的 CloudTrail 日志文件和从多个账户中接收 CloudTrail 日志文件

CloudTrail记录所有Amazon Transcribe操作，这些操作记录在 API 参考中。例如，CreateVocabularyGetTranscriptionJob、和StartTranscriptionJob操作在CloudTrail日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。此信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是用户凭证还是用户凭证还是IAM用户用户凭证还是
- 请求是使用IAM角色还是联合身份用户的临时安全凭证还是联合身份用户的用户发出发出发出发出发出发出发出
- 请求是否由其他发出发出发出发出发出AWS 服务

有关更多信息，请参阅 CloudTrail userIdentity 元素。

您还可以将多个AWS 区域和多个Amazon Transcribe日志文件聚合AWS 账户到单个Amazon S3存储桶中。有关更多信息，请参阅接收多个区域中的 CloudTrail 日志文件和从多个账户中接收 CloudTrail 日志文件。

示例：Amazon Transcribe 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指定Amazon S3存储桶。 CloudTrail日志文件包含一个或多个日志条目。事件 表示来自任何源的单个请求。它包括有关所请求操作的信息（例如操作的日期和时间）以及请求参数。 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。

对StartTranscriptionJob和 GetTranscriptionJobAPI 操作的调用会创建以下条目。

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "111122223333",
                "arn": "arn:aws:iam:us-west-2:111122223333:user/my-user-name",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "my-user-name"
```

```
        },
        "eventTime": "2022-03-07T15:03:45Z",
        "eventSource": "transcribe.amazonaws.com",
        "eventName": "StartTranscriptionJob",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "[ ]",
        "requestParameters": {
            "mediaFormat": "flac",
            "languageCode": "en-US",
            "transcriptionJobName": "my-first-transcription-job",
            "media": {
                "mediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
            }
        },
        "responseElements": {
            "transcriptionJob": {
                "transcriptionJobStatus": "IN_PROGRESS",
                "mediaFormat": "flac",
                "creationTime": "2022-03-07T15:03:44.229000-08:00",
                "transcriptionJobName": "my-first-transcription-job",
                "languageCode": "en-US",
                "media": {
                    "mediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
                }
            }
        },
        "requestID": "47B8E8D397DCE7A6",
        "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",
        "eventType": "AwsApiCall",
        "recipientAccountId": "111122223333"
    },
    {
        "eventVersion": "1.05",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam:us-west-2:111122223333:user/my-user-name",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "my-user-name"
        },
        "eventTime": "2022-03-07T15:07:11Z",
        "eventSource": "transcribe.amazonaws.com",
```

```
            "eventName": "GetTranscriptionJob",
            "awsRegion": "us-west-2",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "[ ]",
            "requestParameters": {
                "transcriptionJobName": "my-first-transcription-job"
            },
            "responseElements": {
                "transcriptionJob": {
                    "settings": {

                    },
                    "transcriptionJobStatus": "COMPLETED",
                    "mediaFormat": "flac",
                    "creationTime": "2022-03-07T15:03:44.229000-08:00",
                    "transcriptionJobName": "my-first-transcription-job",
                    "languageCode": "en-US",
                    "media": {
                        "mediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
                    },
                    "transcript": {
                        "transcriptFileUri": "s3://DOC-EXAMPLE-BUCKET/my-first-
transcription-job.json"
                    }
                }
            },
            "requestID": "BD8798EACDD16751",
            "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        }
    ]
}
```

配合使用 Amazon EventBridge 和 Amazon Transcribe

利用 Amazon EventBridge，您可以通过触发其它 AWS 服务中的事件来响应您的 Amazon Transcribe 作业中的状态更改。当转录作业更改状态时，EventBridge 会自动将事件发送到事件流。您可以创建规则来定义希望在事件流中监控的事件以及在这些事件发生时 EventBridge 应采取的操作。例如，将事件路由到其他服务（或目标），然后再采取操作。例如，您可以将规则配置为在成功完成转录作业时将事件路由到 AWS Lambda 函数。要定义 EventBridge 规则，请参阅以下各节。

可以通过多个渠道接收事件通知，包括电子邮件、AWS Chatbot聊天通知或AWS Console Mobile Application推送通知。还可以在控制台通知中心中查看通知。如果要设置通知，您可以使用 AWS 用户通知服务。AWS 用户通知服务 支持聚合，这可以减少您在特定事件期间收到的通知数量。

定义 EventBridge 规则

要定义 EventBridge 规则，请使用 AWS Management Console。在定义规则时，将 Amazon Transcribe 用作服务名称。有关如何创建 EventBridge 规则的示例，请参阅 Amazon EventBridge 规则。

在使用 EventBridge 之前，请注意以下定义：

- 事件 - 事件指示某个转录作业的状态发生更改。例如，当作业的 `TranscriptionJobStatus` 从 `IN_PROGRESS` 更改为 `COMPLETED` 时。
- 目标 - 目标是其它处理事件的 AWS 服务。例如，AWS Lambda 或 Amazon Simple Notification Service (Amazon SNS)。目标接收 JSON 格式的事件。
- 规则 - 规则与您希望 EventBridge 监视的传入事件匹配并将这些事件路由到目标以进行处理。如果规则将一个事件路由到多个目标，则所有目标将并行处理该事件。规则可以自定义发送到目标的 JSON。

Amazon EventBridge 事件是尽力发出的。有关在 EventBridge 中创建和管理事件的更多信息，请参阅《Amazon EventBridge 用户指南》中的 Amazon EventBridge 事件。

以下是转录作业状态变为 `COMPLETED` 或 `FAILED` 时启动的 Amazon Transcribe 的 EventBridge 规则示例。

```
{
    "source": [
        "aws.transcribe"
    ],
    "detail-type": [
        "Transcribe Job State Change"
    ],
    "detail": {
        "TranscriptionJobStatus": [
            "COMPLETED",
            "FAILED"
        ]
    }
```

```
}
```

规则包含以下字段：

- source - 事件的来源。对于 Amazon Transcribe，这始终为 aws.transcribe。
- detail-type - 事件的详细信息的标识符。对于 Amazon Transcribe，这始终为 Transcribe Job State Change。
- detail - 转录作业的新作业状态。在该示例中，规则将在作业状态变为 COMPLETED 或 FAILED 时触发事件。

Amazon Transcribe 事件

Amazon EventBridge 记录几个 Amazon Transcribe 事件：

- 转录作业事件
- 语言识别事件
- 通话分析事件
- 通话分析通话后事件
- 词汇表事件

这些事件都包含以下共享字段：

- version：事件数据的版本。此值始终为 0。
- id：由 EventBridge 生成的事件的唯一标识符。
- detail-type：事件的详细信息的标识符。例如，Transcribe Job State Change。
- source：事件的来源。对于 Amazon Transcribe，这始终为 aws.transcribe。
- account：生成了 API 调用的账户的 AWS 账户 ID。
- time：事件的交付日期和时间。
- region：在其中发出请求的 AWS 区域。
- resources：API 调用使用的资源。对于 Amazon Transcribe，此字段始终为空。
- detail：有关事件的其它详细信息。
  - FailureReason：如果状态更改为 FAILED，则会显示此字段，并描述 FAILED 状态的原因。
  - 每种事件类型都有额外的唯一字段，显示在 detail 下面。每个事件示例之后会在以下部分中定义这些唯一字段。

转录作业事件

当作业的状态从 IN_PROGRESS 更改为 COMPLETED 或 FAILED 时，Amazon Transcribe 会生成事件。要标识已更改状态并在您的目标中发起事件的作业，请使用事件的 TranscriptionJobName 字段。Amazon Transcribe 事件包含以下信息。如果您的转录作业状态为 FAILED，则会在 detail 下添加一个 FailureReason 字段。

请注意，此事件仅适用于 StartTranscriptionJob API 操作。

```
{
    "version": "0",
    "id": "event ID",
    "detail-type":"Transcribe Job State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [ ],
    "detail": {
          "TranscriptionJobName": "my-first-transcription-job",
          "TranscriptionJobStatus": "COMPLETED" (or "FAILED")
    }
}
```

- TranscriptionJobName：您为转录作业选择的唯一名称。
- TranscriptionJobStatus ：转录作业的状态。可以是 COMPLETED 或 FAILED。

语言识别事件

启用自动语言识别时，Amazon Transcribe 会在语言识别状态为 COMPLETED 或 FAILED 时生成一个事件。要标识已更改状态并在您的目标中发起事件的作业，请使用事件的 JobName 字段。Amazon Transcribe 事件包含以下信息。如果您的语言识别状态为 FAILED，则会在 detail 下添加一个 FailureReason 字段。

请注意，此事件仅适用于包含 LanguageIdSettings 参数时的 StartTranscriptionJob API 操作。

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Language Identification State Change",
```

```
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [ ],
    "detail": {
        "JobType": "TranscriptionJob",
        "JobName": "my-first-lang-id-job",
        "LanguageIdentificationStatus": "COMPLETED" (or "FAILED")
    }
}
```

- JobType：对于转录作业，此值必须为 TranscriptionJob。
- JobName: 您的转录作业的唯一名称。
- LanguageIdentificationStatus：转录作业中语言识别的状态。可以是 COMPLETED 或 FAILED。

通话分析事件

当通话分析作业的状态从 IN_PROGRESS 更改为 COMPLETED 或 FAILED 时，Amazon Transcribe 会生成事件。要标识已更改状态并在您的目标中触发事件的通话分析作业，请使用事件的 JobName 字段。Amazon Transcribe 事件包含以下信息。如果您的通话分析作业状态为 FAILED，则会在 detail 下添加一个 FailureReason 字段。

请注意，此事件仅适用于 StartCallAnalyticsJob API 操作。

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Call Analytics Job State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [ ],
    "detail": {
        "JobName": "my-first-analytics-job",
        "JobStatus": "COMPLETED" (or "FAILED")
    }
}
```

- JobName：您的通话分析转录作业的唯一名称。
- JobStatus：您的通话分析转录作业的状态。此值可以是 COMPLETED 或 FAILED。

通话分析通话后事件

当通话后分析转录的状态从 IN_PROGRESS 变为 COMPLETED 或 FAILED 时，Amazon Transcribe 会生成一个事件。要标识已更改状态并在您的目标中触发事件的通话分析通话后作业，请使用事件的 StreamingSessionId 字段。

请注意，此事件仅适用于包含 PostCallAnalyticsSettings 参数时的 StartCallAnalyticsStreamTranscription API 操作。

COMPLETED 事件包含以下信息：

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Call Analytics Post Call Job State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [ ],
    "detail": {
        "StreamingSessionId": "session-id",
        "PostCallStatus": "COMPLETED",
        "Transcript": {
            "RedactedTranscriptFileUri": "s3://DOC-EXAMPLE-BUCKET/my-output-files/my-
redacted-file.JSON",
            "TranscriptFileUri": "s3://DOC-EXAMPLE-BUCKET/my-output-files/my-file.JSON"
        },
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-output-files/my-redacted-
file.WAV",
            "RedactedMediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-output-files/my-
redacted-file.WAV"
        }
    }
}
```

FAILED 事件包含以下信息：

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Call Analytics Post Call Job State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [ ],
    "detail": {
        "StreamingSessionId": "session-id",
        "PostCallStatus": "FAILED"
    }
}
```

- StreamingSessionId：分配给您的实时通话分析转录请求的识别码。
- PostCallStatus：您的通话后通话分析转录的状态。此值可以是 COMPLETED 或 FAILED。
- Transcript: 您已编辑和未编辑的转录的 URI。
- Media: 您已编辑和未编辑的音频文件的 URI。

词汇表事件

当自定义词汇表的状态从 PENDING 更改为 READY 或 FAILED 时，Amazon Transcribe 会生成一个事件。要标识已更改状态并在您的目标中发起事件的自定义词汇表，请使用事件的 VocabularyName 字段。Amazon Transcribe 事件包含以下信息。如果您的自定义词汇表状态为 FAILED，则会在 detail 下添加一个 FailureReason 字段。

请注意，此事件仅适用于 CreateVocabulary API 操作。

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Vocabulary State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [ ],
    "detail": {
        "VocabularyName": "unique-vocabulary-name",
```

```
        "VocabularyState": "READY" (or "FAILED")
    }
}
```

- VocabularyName: 您的自定义词汇表的唯一名称。
- VocabularyState: 您的自定义词汇表的处理状态。可以是 READY 或 FAILED。

Amazon Transcribe 的合规性验证

要了解某个 AWS 服务 是否在特定合规性计划范围内，请参阅合规性计划范围内的 AWS 服务 ，然后选择您感兴趣的合规性计划。有关常规信息，请参阅AWS 合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅在 AWS Artifact 中下载报告、。

您使用 AWS 服务 的合规性责任取决于数据的敏感度、贵公司的合规性目标以及适用的法律法规。AWS 提供以下资源来帮助满足合规性：

- 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 上部署以安全性和合规性为重点的基准环境的步骤。
- Amazon Web Services 上的 HIPAA 安全性和合规性架构设计 – 该白皮书介绍了公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。

> **Note**
>
> 并非所有 AWS 服务 都符合 HIPAA 要求。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- AWS 客户合规指南：从合规角度了解责任共担模式。这些指南总结了保护 AWS 服务 的最佳实践，并将指南映射到跨多个框架的安全控制，包括美国国家标准与技术研究院（NIST）、支付卡行业安全标准委员会（PCI）和国际标准化组织（ISO）。
- AWS Config 开发人员指南中的使用规则评估资源 – 此 AWS Config 服务评测您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub——此 AWS 服务 向您提供 AWS 中安全状态的全面视图。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实操。有关受支持服务及控制的列表，请参阅 Security Hub 控制参考。

- AWS Audit Manager：此 AWS 服务 可帮助您持续审核您的 AWS 使用情况，以简化管理风险以及与相关法规和行业标准的合规性的方式。

Amazon Transcribe 中的故障恢复能力

AWS 全球基础设施围绕 AWS 区域 和可用区构建。AWS 区域 提供多个在物理上独立且隔离的可用区，这些可用区与延迟率低、吞吐量高且冗余性高的网络连接在一起。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故障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、容错性和可扩展性。

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 Transcribe 中的基础设施安全性

作为托管服务Amazon Transcribe，受AWS全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础架构的信息，请参阅 AWS 云安全。要按照基础设施安全最佳实践设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的 基础设施保护。

您可以使用 AWS 发布的 API 调用通过网络访问 Amazon Transcribe。客户端必须支持以下内容：

- 传输层安全性协议（TLS） 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（Ephemeral Diffie-Hellman）或 ECDHE（Elliptic Curve Ephemeral 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 委托人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 Transcribe 中的漏洞分析和管理

配置和 IT 控制是AWS和您（我们的客户）之间的共同责任。有关更多信息，请参阅AWS责任共担模型。

Amazon Transcribe 和接口 VPC 端点 (AWS PrivateLink)

您可以通过创建接口 VPC 端点在 VPC 和 Amazon Transcribe 之间建立私有连接。接口端点由以下各方提供支持AWS PrivateLink，一种你可以用来私下访问的技术Amazon Transcribe没有互联网网

关、NAT 设备、VPN 连接的 API 或AWS Direct Connect连接。VPC 中的实例即使没有公有 IP 地址也可与 Amazon Transcribe API 进行通信。VPC 和 Amazon Transcribe 之间的流量不会脱离 Amazon 网络。

每个接口端点均由子网中的一个或多个弹性网络接口表示。

有关更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 终端节点 (AWS PrivateLink)。

Amazon Transcribe VPC 终端节点注意事项

在为接口设置接口 VPC 终端节点之前Amazon Transcribe，请务必查看接口端点属性和限制在Amazon VPC用户指南。

Amazon Transcribe 支持从 VPC 调用它的所有 API 操作。

为 Amazon Transcribe 创建接口 VPC 终端节点

您可以为创建一个 VPC 终端节点Amazon Transcribe使用服务Amazon VPC AWS Management Console要么AWS CLI。有关更多信息，请参见创建接口终端节点在Amazon VPC用户指南。

用于批量转录Amazon Transcribe，使用以下服务名称创建 VPC 终端节点：

- com.amazonaws。*us-west-2*. 转录

用于流式传输转录Amazon Transcribe，使用以下服务名称创建 VPC 终端节点：

- com.amazonaws。*us-west-2*. 转录直播

如果您为终端节点启用私有 DNS，则可以向 API 发出 API 请求Amazon Transcribe使用其默认 DNS 名称为AWS 区域，例如，`transcribestreaming.us-east-2.amazonaws.com`。

有关更多信息，请参见通过接口端点访问服务在Amazon VPC用户指南。

为 Amazon Transcribe 创建 VPC 终端节点策略

您可以将终端节点策略附加到您的 VPC 终端节点，以控制对流媒体服务或批量转录服务的访问权限 Amazon Transcribe。该策略指定以下信息：

- 可执行操作的委托人。
- 可执行的操作。

- 可对其执行操作的资源。

有关更多信息，请参见使用 VPC 终端节点控制对服务的访问在Amazon VPC用户指南。

示例：的 VPC 终端节点策略Amazon Transcribe批量转录操作

以下是批量转录的终端节点策略示例Amazon Transcribe。当附加到端点时，此策略会向所有资源上的所有主体授予对列出的 Amazon Transcribe 操作的访问权限。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartTranscriptionJob",
            "transcribe:ListTranscriptionJobs"
         ],
         "Resource":"*"
      }
   ]
}
```

示例：的 VPC 终端节点策略Amazon Transcribe直播转录操作

以下是中用于流式转录的终端节点策略的示例Amazon Transcribe。当附加到端点时，此策略会向所有资源上的所有主体授予对列出的 Amazon Transcribe 操作的访问权限。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartStreamTranscription",
            "transcribe:StartStreamTranscriptionWebsocket"
         ],
         "Resource":"*"
      }
   ]
}
```

共享子网

您无法在与您共享的子网中创建、描述、修改或删除 VPC 终端节点。但是，您可以在与您共享的子网中使用 VPC 端点。有关 VPC 共享的信息，请参阅与其他账户共享您的 VPC在Amazon Virtual Private Cloud指南。

Amazon Transcribe 的安全最佳实践

以下最佳实践是一般指导原则，并不代表完整安全解决方案。由于这些最佳实践可能不适合您的环境或访问控制，因此将其作为有用的考虑因素而不是访问控制措施。

- 使用数据加密，例如AWS KMS加密上下文

  AWS KMS加密上下文是纯文本、非密钥:值对的映射。此地图表示其他经过身份验证的数据，称为加密上下文对，可为您的数据提供额外的安全保护。

  有关更多信息，请参阅 AWS KMS 加密上下文。
- 尽可能使用临时证书

  在可能的情况下，使用临时凭证，而不是访问密钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥钥 对于需要IAM具有编程控制权权权权权权权权权权权权权权权权权权权权权权权限和访问密钥。定期轮换长期凭证有助于您熟悉该流程。这在您遇到必须轮换凭证的情况（例如员工离开公司时）下非常有用。我们建议您使用IAM上次使用的信息，来安全地轮换和删除访问密钥。

  有关更多信息，请参阅中的轮换访问密钥和安全最佳实践IAM。
- 为需要Amazon Transcribe访问的应用程序和AWS服务使用IAM角色

  使用IAM角色来管理需要访问控制Amazon Transcribe。在使用角色时，您不需要将长期凭证（如密码或访问密钥）分配给Amazon EC2实例或AWS服务。 IAM角色可以提供临时权限供应用程序在请求资源时，应用程序在请求AWS资源时。

  如需了解更多信息，请参阅阅：IAM角色和访问控制、访问控制、访问控制和访问控制策略。
- 使用基于标签的控制控制

  您可以使用标签来控制控制AWS 账户。在Amazon Transcribe. 标签可以添加到：转录作业、自定义词汇表、自定义词汇过滤器和自定义语言模型。

  有关更多信息，请参阅 基于标签的访问控制。
- 使用AWS监控工具

监控是保持 Amazon Transcribe 和您的 AWS 解决方案的可靠性、安全性、可用性和性能的重要环节。您可以Amazon Transcribe使用进行监控CloudTrail。

有关更多信息，请参阅 使用 AWS CloudTrail 监控 Amazon Transcribe。

- Enable AWS Config (启用 Gem)

  AWS Config可以评估、审计和评估您的AWS资源配置。使用AWS Config，您可以查看配置的更改以及AWS资源之间的关系。您还可以查看详细的资源配置历史记录，并判断您的配置在整体上是否符合内部指南中所指定的配置要求。这可以帮助您简化合规性审核、安全性分析、变更管理和操作故障排除。

  有关更多信息，请参阅什么是AWS Config？

Amazon Transcribe Medical

Amazon TranscribeMedical 是一项自动语音识别 (ASR) 服务，专为想要转录医疗相关语音（例如医生口述笔记、药物安全监测、远程医疗预约或医患对话）的医疗专业人员而设计。 Amazon TranscribeMedical 可以通过实时直播（通过麦克风）或转录上传的文件（批处理）获得。

> **Important**
>
> Amazon Transcribe医疗不能替代专业的医疗建议、诊断或治疗。为您的使用案例确定正确的置信度阈值，并在需要高准确度的情况下使用高置信度阈值。对于某些用例，应由经过适当培训的人工审查人员对结果进行审查和验证。 Amazon Transcribe医学转录只能在经过训练的医疗专业人员审查后才能在患者护理场景中使用，以确保准确性和合理的医学判断。

Amazon TranscribeMedical 在分担责任模式下运营，AWS即负责保护运行 MAmazon Transcribe edical 的基础设施，您负责管理您的数据。有关更多信息，请参阅责任共担模式。

Amazon Transcribe医疗提供美式英语 (en-US) 版本。

为获得最佳效果，请使用 PCM 16 位编码的无损音频格式，例如 FLAC 或 WAV。 Amazon Transcribe Medical支持 16,000 Hz 或更高的采样率。

要分析您的成绩单，您可以使用其他AWS 服务，例如Amazon Comprehend Medical。

支持的专业

专业	子专业	音频输入
心脏病学	无	仅限直播
神经病学	无	仅限直播
肿瘤学	无	仅限直播
初级保健	家庭医疗	流流流流流批量
初级保健	内科	流流流流流批量
初级保健	妇产科 (OB-GYN)	流流流流流批量

专业	子专业	音频输入
初级保健	儿科	流流流流流批量
放射学	无	仅限直播
泌尿外科	无	仅限直播

区域可用性和配额

以下方面支持呼叫分析AWS 区域：

区域	转录类型
af-south-1（开普敦）	批处理
ap-east-1（香港）	批处理
ap-northeast-1 (东京)	流流流流流批量
ap-northeast-2 (首尔)	流流流流流批量
ap-south-1（孟买）	批处理
ap-southeast-1 (新加坡)	批处理
ap-southeast-2 (悉尼)	流流流流流批量
ca-central-1（加拿大，中部）	流流流流流批量
eu-central-1 (法兰克福)	流流流流流批量
eu-noth-1（斯德哥尔摩）	批处理
eu-west-1 (爱尔兰)	流流流流流批量
eu-west-2（伦敦）	流流流流流批量
eu-west-3（巴黎）	批处理

区域	转录类型
me-south-1（巴林）	批处理
sa-east-1（圣保罗）	流流流流流批量
us-east-1（弗吉尼亚北部）	流流流流流批量
us-east-2（俄亥俄）	流流流流流批量
us-gov-east-1（GovCloud，美国东部）	流流流流流批量
us-gov-west-1（GovCloud，美国西部）	流流流流流批量
us-west-1（旧金山）	批处理
us-west-2 (俄勒冈)	流流流流流批量

请注意，Amazon Transcribe、Amazon Transcribe Medical和呼叫分析的区域支持有所不同。 。

要获取每个支持区域的终端节点，请参阅AWS一般参考中的服务终端节点。

有关与您的转录相关的配额列表，请参阅AWS一般参考中的服务配额。某些配额可以根据要求更改。如果 “可调” 列包含 “是”，则可以请求增加。为此，请选择提供的链接。

医学专业和术语

创建医疗转录作业时，您需要指定源文件的语言、医学专业和音频类型。您输入美国英语作为语言，并输入 PRIMARYCARE 作为医学专业。输入初级保健作为值使可让您从以下医学专业的源音频生成转录：

- 家庭医疗
- 内科
- 妇产科 (OB-GYN)
- 儿科

您可以选择口述和对话作为您的音频类型。对于医生在其中报告有关患者就诊或手术的音频文件选择口述。对于涉及医生与患者之间对话或医生之间对话的音频文件选择对话。

要存储转录作业的输出，请选择您已创建的 Amazon S3 存储桶。有关Amazon S3存储桶的更多信息，请参阅入门Amazon Simple Storage Service。

以下是在示例 JSON 中输入的最小请求参数数。

```
{
   "MedicalTranscriptionJobName": "my-first-transcription-job",
   "LanguageCode": "en-US",
   "Media": {
       "MediaFileUri": "s3://path to your audio file"
   },
   "OutputBucketName": “your output bucket name",
   "Specialty": "PRIMARYCARE",
   "Type": "CONVERSATION"
}
```

Amazon TranscribeMedical 使您能够生成替代转录。有关更多信息，请参阅生成替代转录：

您还可以启用扬声器分区或识别音频中的频道。有关更多信息，请参阅 启用扬声器分区 和 转录多声道音频。

转录医学术语和测量结果

Amazon Transcribe医学可以转录医学术语和测量结果。 Amazon Transcribe医疗输出口语的缩写。例如，“血压”被转录为 BP。您可以在本页的表格中找到 MAmazon Transcribe edical 在医学术语和测量中使用的惯例列表。说出的术语 列是指源音频中说出的术语。输出 列指您在转录结果中看到的缩写。

您可以在此处查看源音频中所说的术语如何与转录输出对应。

源音频中所说的术语	输出中使用的缩写	输出示例
摄氏度	C	患者的体温为 37.4 摄氏度。
摄氏度	C	患者的体温为 37.4 摄氏度。
华氏度	F	患者的体温是 101 华氏度。
克	网	从患者提取了 100 克的质量。
米	m	患者身高 1.8 米。
英尺	ft	患者身高 6 英尺。

源音频中所说的术语	输出中使用的缩写	输出示例
千克	kg	患者体重 80 千克。
千克	kg	患者体重 80 千克。
c c	cc	给患者注射 100 毫升的盐水溶液。
立方厘米	cc	给患者注射 100 毫升的盐水溶液。
毫升	mL	患者排出 100 毫升尿液。
血压	BP	患者血压升高。
b p	BP	患者血压升高。
X / Y	X/Y	患者血压为 120/80。
每分钟心跳数	BPM	患者患有心房颤动，心率为 160 BPM。
每分钟心跳数	BPM	患者患有心房颤动，心率为 160 BPM。
O 2	氧气	患者血氧饱和度为 98%。
二氧化碳	二氧化碳	患者需要呼吸支持，以防二氧化碳浓度升高。
术后	POSTOP	患者来接受 POSTOP 评估。
术后	POSTOP	患者来接受 POSTOP 评估。
CAT 扫描	CT 扫描	脑出血的患者指示需要使用 CT 扫描。
脉搏 80	P 80	患者生命体征为 P 80、R 17、...

源音频中所说的术语	输出中使用的缩写	输出示例
呼吸 17	R 17	患者生命体征为 P 80、R 17、...
输入和输出	I/O	患者为 I/O 窦性心律
L5	L5	在 L4 和 L5 之间进行腰椎穿刺

转录数字

Amazon Transcribe医学将数字转录为数字而不是单词。例如，语音读出来的数字“one thousand two hundred forty-two”会转录成 1242。

数字是根据以下规则转录的。

Rule	描述
将大于 10 的基数词转换成数字。	• “Fifty five”> 55 • “a hundred”> 100 • “One thousand and thirty one”> 1031 • "One hundred twenty-three million four hundred fifty six thousand seven hundred eight nine" > 123456789
当“million”或“billion”后面没有数字时，将后跟“million”或“billion”的基数词转换为数字后跟一个单词。	• "one hundred million" > 100 million • "one billion" > 1 billion • "two point three million" > 2.3 million
将大于 10 的序数词转换成数字。	• “Forty third”> 43rd • “twenty sixth avenue”> 26th avenue
将小数转换成数字格式。	• “a quarter”> 1/4 • “three sixteenths”> 3/16 • “a half”> 1/2 • "a hundredth" > 1/100

Rule	描述
将小于 10 的数字转换成阿拉伯数字（如果一行中有多个数字的话）。	• “three four five”> 345 • “My phone number is four two five five five five one two one two”> 4255551212
小数点由“dot”或“point”表示。	• “three hundred and three dot five”> 303.5 • “three point twenty three”> 3.23 • “zero point four”> 0.4 • “point three”> 0.3
将数字后的单词“percent”转换成百分号。	• “twenty three percent”> 23% • “twenty three point four five percent”> 23.45%
将数字后面的单词“dollar”、“Australian dollar”、“AUD”、“US dollar”或“USD”转换成美元符号加在数字之前。	• “one dollar and fifteen cents”> $1.15 • "twenty three USD" > $23 • "twenty three Australian dollars" > $23
将单词“pounds”或“milligrams”转换为“lbs”或“mg”。	• "twenty three pounds" > 23 lbs • “forty-five milligrams” > 45 mg
将数字后面的单词“rupees”、“Indian rupees”或“INR”转换卢比符号 (#) 加在数字之前。	• "twenty three rupees" > #23 • "fifty rupees thirty paise" > #50.30
将时间转换成数字。	• “seven a m eastern standard time”> 7 a.m. eastern standard time • “twelve thirty p m”> 12:30 p.m.
将以两位数表示的年份组合成四位数的年份。 仅适用于 20、21 和 22 世纪的年份。	• “nineteen sixty two”> 1962 • “the year is twenty twelve”> the year is 2012 • “twenty nineteen”> 2019 • “twenty one thirty” > 2130

Rule	描述
将日期转换成数字。	• “May fifth twenty twelve”> May 5th 2012 • “May five twenty twelve”> May 5 2012 • “five May twenty twelve”> 5 May 2012
数字范围用单词“to”来分隔。	• “twenty three to thirty seven”> 23 to 37

转录医学谈话

您可以使用 MAmazon Transcribe edical 使用批量转录作业或实时直播来转录临床医生和患者之间的医疗对话。Batch 转录作业使您能够转录音频文件。为确保Amazon Transcribe Medical 以尽可能高的准确度生成转录结果，您必须在转录工作或直播中指定临床医生的医学专业。

您可以转录以下医学专业的临床医生-患者就诊情况：

- 心脏病学 — 仅以流媒体转录形式提供
- 神经病学 — 仅以流媒体转录形式提供
- 肿瘤学 — 仅以流媒体转录形式提供
- 初级保健 — 包括以下类型的医疗实践：
  - 家庭医疗
  - 内科
  - 妇产科 (OB-GYN)
  - 儿科
- 泌尿科——仅提供流媒体转录

您可以通过使用医学自定义词汇来提高转录准确性。有关医学自定义词汇表的工作原理的信息，请参阅使用医学自定义词汇提高转录准确性。

默认情况下，MAmazon Transcribe edical 返回具有最高置信度的转录内容。如果您想将其配置为返回备用转录，请参阅生成替代转录。

有关数字和医学测量值在转录输出中如何显示的信息，请参见转录数字和转录医学术语和测量结果。

主题

转录医学对话的音频文件

使用批量转录作业转录医学对话的音频文件。你可以用它来转录临床医生与患者的对话。您可以在StartMedicalTranscriptionJobAPI 或中启动批量转录作业AWS Management Console。

当您使用 StartMedicalTranscriptionJobAPI 开始医学转录作业时，您可以指定PRIMARYCARE为Specialty参数的值。

AWS Management Console

转录临床医生与患者的对话 (AWS Management Console)

要使用转录临床医生与患者的对话，请创建转录作业并选择 “对话” 作为音频输入类型。AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中的 MedicalinceratiAmazon Transcribe onshots 下，选择 Tran cap
3. 请选择 Create job (创建任务)。
4. 在指定Job 详细信息页面的作业设置下，指定以下内容。
    a. 名称 – 转录任务的名称。
    b. 音频输入类型-会话
5. 对于其余字段，请指定音频文件的Amazon S3位置以及要存储转录作业输出的位置。
6. 选择 Next (下一步)。
7. 选择创建。

API

使用批量转录作业 (API) 转录医学对话

- 对于 StartMedicalTranscriptionJobAPI，指定以下内容。

a. 对于MedicalTranscriptionJobName，请指定一个在您的名称中独一无二的名称AWS 账户。

b. 对于LanguageCode，请指定与您的音频文件中所说的语言和词汇过滤器的语言相对应的语言代码。

c. 对于Media对象的MediaFileUri参数，指定要转录的音频文件的名称。

d. 对于Specialty，将音频文件中讲话的临床医生的医学专业指定为PRIMARYCARE。

e. 对于 Type，请指定 CONVERSATION。

f. 对于OutputBucketName，指定用于Amazon S3存储转录结果的存储桶。

以下是使用AWS SDK for Python (Boto3)转录PRIMARYCARE专业临床医生和患者的医疗谈话的示例请求。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-med-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-file.flac"
transcribe.start_medical_transcription_job(
      MedicalTranscriptionJobName = job_name,
      Media = {
        'MediaFileUri': job_uri
      },
      OutputBucketName = 'DOC-EXAMPLE-BUCKET',
      OutputKey = 'output-files/',
      LanguageCode = 'en-US',
      Specialty = 'PRIMARYCARE',
      Type = 'CONVERSATION'
  )

while True:
    status = transcribe.get_medical_transcription_job(MedicalTranscriptionJobName =
 job_name)
    if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
```

```
print(status)
```

以下示例代码显示了临床医生与患者对话的转录结果。

```
{
    "jobName": "conversation-medical-transcription-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "... come for a follow up visit today..."
            }
        ],
        "items": [
            {
            ...
                "start_time": "4.85",
                "end_time": "5.12",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "come"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "5.12",
                "end_time": "5.29",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "for"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "5.29",
                "end_time": "5.33",
```

```
                "alternatives": [
                    {
                        "confidence": "0.9955",
                        "content": "a"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "5.33",
                "end_time": "5.66",
                "alternatives": [
                    {
                        "confidence": "0.9754",
                        "content": "follow"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "5.66",
                "end_time": "5.75",
                "alternatives": [
                    {
                        "confidence": "0.9754",
                        "content": "up"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "5.75",
                "end_time": "6.02",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "visit"
                    }
                ]
                ...
    },
    "status": "COMPLETED"
}
```

AWS CLI

使用批量转录作业转录医学谈话 (AWS CLI)

- 运行以下代码。

```
aws transcribe start-medical-transcription-job \
--region us-west-2 \
--cli-input-json file://example-start-command.json
```

以下代码显示的内容example-start-command.json。

```
{
      "MedicalTranscriptionJobName": "my-first-med-transcription-job",
      "Media": {
          "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-
file.flac"
      },
      "OutputBucketName": "DOC-EXAMPLE-BUCKET",
      "OutputKey": "my-output-files/",
      "LanguageCode": "en-US",
      "Specialty": "PRIMARYCARE",
      "Type": "CONVERSATION"
  }
```

在实时直播中转录医学对话

您可以使用 HTTP/2 或WebSocket 协议转录医疗对话的音频流。有关如何使用 WebSocket 协议启动直播的信息，请参阅设置直 WebSocket 播。要启动 HTTP/2 数据流，请使用 StartMedicalStreamTranscriptionAPI。

您可以转录以下医学专业的流媒体音频：

- 心脏病学
- 神经病学
- 肿瘤学

- 初级保健
- 泌尿外科

每个医学专业都包括许多类型的手术和预约。因此，临床医生会口述多种不同类型的笔记。使用以下示例作为指导，帮助您指定 WebSocket 请求的`specialty` URI 参数的值或 `StartMedicalStreamTranscription`API 的`Specialty`参数：

- 如需电生理学或超声心动图咨询，请选择`CARDIOLOGY`。
- 对于肿瘤内科、外科肿瘤学或放射肿瘤学咨询，请选择`ONCOLOGY`。
- 对于为中风（短暂性脑缺血发作或脑血管发作）的患者提供咨询的医生，请选择`NEUROLOGY`。
- 如需有关尿失禁的咨询，请选择`UROLOGY`。
- 对于年度检查或紧急护理就诊，请选择`PRIMARYCARE`。
- 对于住院医生的就诊，请选择`PRIMARYCARE`。
- 有关生育能力、输卵管结扎、宫内节育器插入或流产的咨询，请选择`PRIMARYCARE`。

AWS Management Console

转录直播医学对话 (AWS Management Console)

要使用在AWS Management Console实时直播中转录临床医生与患者的对话，请选择转录医疗对话、开始直播，然后开始对着麦克风说话。

1. 登录到 AWS Management Console。
2. 在导航窗格中的 MAmazon Transcribe edical 下，选择实时转录。
3. 选择 “对话”。
4. 对于医学专业，请选择临床医生的专业。
5. 选择 Start streaming（开始流式传输）。
6. 对着麦克风说话。

在 HTTP/2 流中转录医学对话

以下是 HTTP/2 请求参数的语法。

要转录医疗对话的 HTTP/2 流，请使用 `StartMedicalStreamTranscription`API 并指定以下内容：

- LanguageCode— 语言代码。有效值为en-US
- MediaEncoding— 用于输入音频的编码。有效值包括 pcm、ogg-opus 和 flac。
- Specialty— 医疗专业人员的专业。
- Type – CONVERSATION

要提高实时直播中特定术语的转录准确性，请使用自定义词汇。要启用自定义词汇表，请将VocabularyName参数的值设置为要使用的自定义词汇表的名称。有关更多信息，请参阅使用医学自定义词汇提高转录准确性：

要标记来自不同发言者的语音，请将ShowSpeakerLabel参数设置为true。有关更多信息，请参阅启用扬声器分区：

有关设置 HTTP/2 流以转录医疗对话的更多信息，请参阅设置 HTTP/2 音频流。

在 WebSocket 直播中转录医学谈话

您可以使用 WebSocket 请求来转录医疗谈话。当你发出 WebSocket 请求时，你会创建一个预签名的 URI。此 URI 包含在应用程序和 MAmazon Transcribe edical 之间建立音频流所需的信息。有关创建 WebSocket 请求的更多信息，请参阅设置直 WebSocket 播。

使用以下模板来创建您的预签名 URI。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/medical-stream-
transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding=flac
&sample-rate=16000
&session-id=sessionId
&specialty=medicalSpecialty
&type=CONVERSATION
&vocabulary-name=vocabularyName
&show-speaker-label=boolean
```

要提高实时直播中特定术语的转录准确性，请使用自定义词汇。要启用自定义词汇表，`vocabulary-name`请将的值设置为要使用的自定义词汇表的名称。有关更多信息，请参阅使用医学自定义词汇提高转录准确性：

要标记来自不同发言者的语音，请将`show-speaker-label`参数设置为`true`。有关更多信息，请参阅启用扬声器分区：

有关创建预签名 URI 的更多信息，请参阅设置直 WebSocket 播。

启用扬声器分区

要在 MAmazon Transcribe edical 中启用扬声器分区，请使用扬声器分区。这使您能够看到患者在转录输出中说了什么以及临床医生说了什么。

当您启用扬声器计时功能时，Amazon TranscribeMedic al 会使用每个说话者的唯一标识符标记每个说话者的话语。话语是一种语音单位，通常通过沉默与其他话语分开。在批量转录中，临床医生的`spk_0`话语可以获得标签，而患者可以获得标签`spk_1`。

如果一个说话者的话与另一位说话者的话重叠，MAmazon Transcribe edical 会按他们的开始时间在转录中命令他们。在输入音频中重叠的表达不会在转录输出中重叠。

当您使用批量转录作业或在实时直播中转录音频文件时，可以启用扬声器排序。

主题

- 在批量转录中启用扬声器分区
- 在实时直播中启用扬声器分区

在批量转录中启用扬声器分区

您可以使用 `StartMedicalTranscriptionJob`API 或在批量转录作业中启用扬声器分区AWS Management Console。这使您能够在临床医生与患者对话中对每个发言者的文本进行分区，并确定转录输出中谁说了什么。

AWS Management Console

要在AWS Management Console转录作业中使用启用扬声器分区，您需要先启用音频识别，然后再启用扬声器分区。

1. 登录到 AWS Management Console。
2. 在导航窗格中的 MedicalinceratiAmazon Transcribe onshots 下，选择 Tran cap

3. 请选择 Create job (创建任务)。
4. 在指定作业详细信息页面上，提供有关您的转录作业的信息。
5. 选择下一步。
6. 启用音频识别。
7. 对于音频识别类型，选择扬声器分区。
8. 在 “最大发言者数量” 中，输入您认为在音频文件中发言的最大发言者数量。
9. 选择创建。

API

使用批量转录作业 (API) 启用扬声器分区

- 对于 StartMedicalTranscriptionJobAPI，指定以下内容。
  a. 对于MedicalTranscriptionJobName，请指定一个在您的名称中是唯一的AWS 账户。
  b. 对于LanguageCode，指定与音频文件中所说的语言相对应的语言代码。
  c. 对于Media对象的MediaFileUri参数，指定要转录的音频文件的名称。
  d. 对于Specialty，请指定在音频文件中发言的临床医生的医学专业。
  e. 对于 Type，请指定 CONVERSATION。
  f. 对于OutputBucketName，指定用于Amazon S3存储转录结果的存储桶。
  g. 对于Settings对象，指定以下内容。
     i. ShowSpeakerLabels – true.
     ii. MaxSpeakerLabels— 介于 2 到 10 之间的整数，表示您认为在音频中说话的发言者数量。

以下请求使用在AWS SDK for Python (Boto3)启用扬声器分区的情况下启动初级保健临床医生患者对话的批量转录作业。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
```

```
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media={
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
Settings = {'ShowSpeakerLabels': True,
         'MaxSpeakerLabels': 2
         }
         )
while True:
    status = transcribe.get_medical_transcription_job(MedicalTranscriptionJobName =
 job_name)
    if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

以下示例代码显示了启用扬声器分区的情况下转录作业的转录结果。

```
{
    "jobName": "job ID",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "Professional answer."
            }
        ],
        "speaker_labels": {
            "speakers": 1,
            "segments": [
                {
```

```
                "start_time": "0.000000",
                "speaker_label": "spk_0",
                "end_time": "1.430",
                "items": [
                    {
                        "start_time": "0.100",
                        "speaker_label": "spk_0",
                        "end_time": "0.690"
                    },
                    {
                        "start_time": "0.690",
                        "speaker_label": "spk_0",
                        "end_time": "1.210"
                    }
                ]
            }
        ]
    },
    "items": [
        {
            "start_time": "0.100",
            "end_time": "0.690",
            "alternatives": [
                {
                    "confidence": "0.8162",
                    "content": "Professional"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "0.690",
            "end_time": "1.210",
            "alternatives": [
                {
                    "confidence": "0.9939",
                    "content": "answer"
                }
            ],
            "type": "pronunciation"
        },
        {
            "alternatives": [
                {
```

```
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ]
    },
    "status": "COMPLETED"
}
```

AWS CLI

转录执业初级保健的临床医生与患者之间对话的音频文件 (AWS CLI)

- 运行以下代码。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--cli-input-json file://example-start-command.json
```

以下代码显示的内容example-start-command.json。

```
{
    "MedicalTranscriptionJobName": "my-first-med-transcription-job",
     "Media": {
          "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-
file.flac"
      },
      "OutputBucketName": "DOC-EXAMPLE-BUCKET",
      "OutputKey": "my-output-files/",
      "LanguageCode": "en-US",
      "Specialty": "PRIMARYCARE",
      "Type": "CONVERSATION",
      "Settings":{
          "ShowSpeakerLabels": true,
          "MaxSpeakerLabels": 2
        }
```

```
}
```

在实时直播中启用扬声器分区

要对发言者进行分区并在实时直播中标记其语音，请使用AWS Management Console或流式传输请求。扬声器分区最适合直播中的两到五个扬声器。尽管Amazon Transcribe Medical 可以在一个直播中对五个以上的扬声器进行分区，但如果你超过这个数字，分区的准确性就会降低。

要启动 HTTP/2 请求，请使用 StartMedicalStreamTranscriptionAPI。要启动 WebSocket 请求，请使用预签名 URI。URI 包含在应用程序和 MAmazon Transcribe edical 之间建立双向通信所需的信息。

在向麦克风说话的音频中启用扬声器分区 (AWS Management Console)

您可以使用AWS Management Console开始实时直播临床医生与患者的对话，也可以开始对着麦克风进行实时听写。

1. 登录到 AWS Management Console。
2. 在导航窗格中，对于 “Amazon Transcribe医疗”，选择 “实时转录”。
3. 对于音频输入类型，选择要转录的医学语音类型。
4. 要进行其他设置，请选择扬声器分区。
5. 选择 “开始直播” 以开始转录您的实时音频。
6. 对着麦克风说话。

在 HTTP/2 流中启用扬声器分区

要在医疗对话的 HTTP/2 流中启用发言者分区，请使用 StartMedicalStreamTranscriptionAPI 并指定以下内容：

- 对于`LanguageCode`，指定与流中语言对应的语言代码。有效值为 `en-US`。
- 对于`MediaSampleHertz`，指定音频的采样率。
- 对于`Specialty`，请指定提供者的医学专业。
- `ShowSpeakerLabel` – `true`

有关设置 HTTP/2 流以转录医疗对话的更多信息，请参阅设置 HTTP/2 音频流。

在 WebSocket 请求中启用扬声器分区

要使用 API 对 WebSocket 直播中的发言者进行分区，请使用以下格式创建预签名 URI 以启动 WebSocket 请求并show-speaker-label将其设置为true。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/medical-stream-
transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding=flac
&sample-rate=16000
&session-id=sessionId
&specialty=medicalSpecialty
&type=CONVERSATION
&vocabulary-name=vocabularyName
&show-speaker-label=boolean
```

以下代码显示了流式请求的截断示例响应。

```
{
  "Transcript": {
    "Results": [
      {
        "Alternatives": [
          {
            "Items": [
              {
                "Confidence": 0.97,
                "Content": "From",
                "EndTime": 18.98,
                "Speaker": "0",
                "StartTime": 18.74,
                "Type": "pronunciation",
                "VocabularyFilterMatch": false
```

```
        },
        {
          "Confidence": 1,
          "Content": "the",
          "EndTime": 19.31,
          "Speaker": "0",
          "StartTime": 19,
          "Type": "pronunciation",
          "VocabularyFilterMatch": false
        },
        {
          "Confidence": 1,
          "Content": "last",
          "EndTime": 19.86,
          "Speaker": "0",
          "StartTime": 19.32,
          "Type": "pronunciation",
          "VocabularyFilterMatch": false
        },
      ...
        {
          "Confidence": 1,
          "Content": "chronic",
          "EndTime": 22.55,
          "Speaker": "0",
          "StartTime": 21.97,
          "Type": "pronunciation",
          "VocabularyFilterMatch": false
        },
        ...
          "Confidence": 1,
          "Content": "fatigue",
          "EndTime": 24.42,
          "Speaker": "0",
          "StartTime": 23.95,
          "Type": "pronunciation",
          "VocabularyFilterMatch": false
        },
        {
          "EndTime": 25.22,
          "StartTime": 25.22,
          "Type": "speaker-change",
          "VocabularyFilterMatch": false
        },
```

```
              {
                "Confidence": 0.99,
                "Content": "True",
                "EndTime": 25.63,
                "Speaker": "1",
                "StartTime": 25.22,
                "Type": "pronunciation",
                "VocabularyFilterMatch": false
              },
              {
                "Content": ".",
                "EndTime": 25.63,
                "StartTime": 25.63,
                "Type": "punctuation",
                "VocabularyFilterMatch": false
              }
            ],
            "Transcript": "From the last note she still has mild sleep deprivation and
 chronic fatigue True."
          }
        ],
        "EndTime": 25.63,
        "IsPartial": false,
        "ResultId": "XXXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXX",
        "StartTime": 18.74
      }
    ]
  }
}
```

Amazon TranscribeMedical 会根据自然的语音片段（例如扬声器更改或音频暂停）来中断您的传入音频流。转录将逐步返回到您的应用程序，每个响应包含更多转录的语音，直到整个片段被转录。前面的代码是完全转录的语音段的截断示例。扬声器标签仅出现在完全转录的片段中。

以下列表显示了流式转录输出中对象和参数的组织结构。

`Transcript`

每个语音段都有自己的`Transcript`对象。

Results

每个Transcript对象都有自己的Results对象。此对象包含该isPartial字段。当其值为时false，返回的结果是整个语音段的。

Alternatives

每个Results对象都有一个Alternatives对象。

Items

每个Alternatives对象都有自己的Items对象，其中包含有关转录输出中每个单词和标点符号的信息。启用扬声器分区时，每个单词都有一个用于完全转录的语音片段的Speaker标签。 Amazon TranscribeMedical 使用此标签为直播中的每位发言者分配一个唯一的整数。值为的Type参数speaker-change表示一个人已经停止说话，另一个人即将开始说话。

Transcript

每个 Items 对象都包含一个转录的语音段作为该Transcript字段的值。

有关 WebSocket 请求的更多信息，请参阅设置直 WebSocket 播。

转录多声道音频

如果您的音频文件或直播有多个频道，则可以使用频道识别来转录每个频道的语音。 Amazon TranscribeMedical 分别转录来自每个频道的语音。它将每个通道的单独转录合并为单个转录输出。

使用频道识别来识别音频中的各个频道，并转录每个频道的语音。在呼叫者和代理场景等情况下启用此功能。在进行药物安全监控的联络中心的录音或直播中，使用它来区分呼叫者和代理人。

您可以为批处理和实时流媒体启用信道识别。以下列表描述了如何为每种方法启用它。

- Batch 转录AWS Management Console和 StartMedicalTranscriptionJobAPI
- 流媒体转录 — WebSocket 流媒体和 StartMedicalStreamTranscriptionAPI

转录多声道音频文件

当您转录音频文件时，MAmazon Transcribe edical 会返回每个频道的项目列表。项目是转录的单词或标点符号。每个单词都有开始时间和结束时间。如果一个频道上的某人代替另一个频道上的某人说话，则当这些人互相说话时，每个频道的项目的开始时间和结束时间会重叠。

默认情况下，您可以转录带有两个声道的音频文件。如果您需要转录包含两个以上频道的文件，则可以申请增加配额。有关请求增加配额的信息，请参阅AWS 服务配额。

要在批量转录作业中转录多声道音频，请使用AWS Management Console或StartMedicalTranscriptionJobAPI。

AWS Management Console

要使用在AWS Management Console批量转录作业中启用频道识别，您需要先启用音频识别，然后再启用频道识别。信道识别是中音频识别的子集AWS Management Console。

1. 登录到 AWS Management Console。
2. 在导航窗格中的 MedicalinceratiAmazon Transcribe onshots 下，选择 Tran cap
3. 请选择 Create job (创建任务)。
4. 在指定作业详细信息页面上，提供有关您的转录作业的信息。
5. 选择下一步。
6. 启用音频识别。
7. 对于音频识别类型，选择频道识别。
8. 选择创建。

API

转录多声道音频文件 (API)

- 对于 StartMedicalTranscriptionJobAPI，指定以下内容。
    a. 对于TranscriptionJobName，请指定一个专属于您的名称AWS 账户。
    b. 对于LanguageCode，指定与音频文件中所说的语言相对应的语言代码。有效值为 en-US。
    c. 对于Media对象的MediaFileUri参数，指定要转录的媒体文件的名称。
    d. 对于Settings对象，设置ChannelIdentification为true。

以下是使用请求的示例AWS SDK for Python (Boto3)。

```
from __future__ import print_function
import time
import boto3
```

```
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_name = "my-first-med-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_medical_transcription_job(
      MedicalTranscriptionJobName = job_name,
      Media = {
        'MediaFileUri': job_uri
      },
      OutputBucketName = 'DOC-EXAMPLE-BUCKET',
      OutputKey = 'output-files/',
      LanguageCode = 'en-US',
      Specialty = 'PRIMARYCARE',
      Type = 'CONVERSATION',
      Settings = {
        'ChannelIdentification': True
      }
)
while True:
    status = transcribe.get_transcription_job(MedicalTranscriptionJobName = job_name)
    if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

AWS CLI

使用批量转录作业转录多声道音频文件 (AWS CLI)

- 运行以下代码。

```
aws transcribe start-medical-transcription-job \
--region us-west-2 \
--cli-input-json file://example-start-command.json
```

以下是的代码example-start-command.json。

```
{
      "MedicalTranscriptionJobName": "my-first-med-transcription-job",
      "Media": {
          "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-
file.flac"
      },
      "OutputBucketName": "DOC-EXAMPLE-BUCKET",
      "OutputKey": "my-output-files/",
      "LanguageCode": "en-US",
      "Specialty": "PRIMARYCARE",
      "Type": "CONVERSATION",

        "Settings":{
          "ChannelIdentification": true
        }
}
```

以下代码显示了在两个通道上进行对话的音频文件的转录输出。

```
{
  "jobName": "job id",
  "accountId": "111122223333",
  "results": {
    "transcripts": [
      {
        "transcript": "When you try ... It seems to ..."
      }
    ],
    "channel_labels": {
      "channels": [
        {
          "channel_label": "ch_0",
          "items": [
            {
              "start_time": "12.282",
              "end_time": "12.592",
              "alternatives": [
                {
                  "confidence": "1.0000",
                  "content": "When"
```

```
                }
              ],
              "type": "pronunciation"
            },
            {
              "start_time": "12.592",
              "end_time": "12.692",
              "alternatives": [
                {
                  "confidence": "0.8787",
                  "content": "you"
                }
              ],
              "type": "pronunciation"
            },
            {
              "start_time": "12.702",
              "end_time": "13.252",
              "alternatives": [
                {
                  "confidence": "0.8318",
                  "content": "try"
                }
              ],
              "type": "pronunciation"
            },
            ...
        ]
    },
    {
        "channel_label": "ch_1",
        "items": [
          {
            "start_time": "12.379",
            "end_time": "12.589",
            "alternatives": [
              {
                "confidence": "0.5645",
                "content": "It"
              }
            ],
            "type": "pronunciation"
          },
          {
```

```
            "start_time": "12.599",
            "end_time": "12.659",
            "alternatives": [
              {
                "confidence": "0.2907",
                "content": "seems"
              }
            ],
            "type": "pronunciation"
          },
          {
            "start_time": "12.669",
            "end_time": "13.029",
            "alternatives": [
              {
                "confidence": "0.2497",
                "content": "to"
              }
            ],
            "type": "pronunciation"
          },
          ...
        ]
    }
}
```

转录多声道音频流

您可以使用 StartMedicalStreamTranscriptionAPI 在 HTTP/2 或 WebSocket直播中转录来自不同频道的音频。

默认情况下，您可以转录带有两个频道的直播。如果您需要转录包含两个以上频道的直播，则可以申请增加配额。有关请求增加配额的信息，请参阅AWS服务配额。

在 HTTP/2 流中转录多声道音频

要转录 HTTP/2 流中的多声道音频，请使用 StartMedicalStreamTranscriptionAPI 并指定以下内容：

- LanguageCode— 音频的语言代码。有效值为 en-US。
- MediaEncoding— 音频的编码。有效值包括 ogg-opus、flac 和 pcm。
- EnableChannelIdentification – true
- NumberOfChannels— 流音频中包含的通道数量。

有关设置 HTTP/2 流以转录医疗对话的更多信息，请参阅设置 HTTP/2 音频流。

在 WebSocket 流中转录多声道音频

要对 WebSocket 直播中的发言者进行分区，请使用以下格式创建预签名 URI 并启动 WebSocket 请求。enable-channel-identification将您的直播中的频道数指定为truenumber-of-channels。预签名的 URI 包含在应用程序和 MAmazon Transcribe edical 之间建立双向通信所需的信息。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/medical-stream-
transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding=flac
&sample-rate=16000
&session-id=sessionId
&enable-channel-identification=true
&number-of-channels=2
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

有关 WebSocket 请求的更多信息，请参阅设置直 WebSocket 播。

多通道流媒体输出

对于 HTTP/2 和 WebSocket请求，流式转录的输出是相同的。下面是一个示例输出。

```
{
    "resultId": "XXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX",
    "startTime": 0.11,
    "endTime": 0.66,
    "isPartial": false,
    "alternatives": [
        {
```

```
            "transcript": "Left.",
            "items": [
                {
                    "startTime": 0.11,
                    "endTime": 0.45,
                    "type": "pronunciation",
                    "content": "Left",
                    "vocabularyFilterMatch": false
                },
                {
                    "startTime": 0.45,
                    "endTime": 0.45,
                    "type": "punctuation",
                    "content": ".",
                    "vocabularyFilterMatch": false
                }
            ]
        }
    ],
    "channelId": "ch_0"
}
```

对于每个语音段，都有一个channelId标志，指示该语音属于哪个频道。

转录医学听写

您可以使用 MAmazon Transcribe edical 使用批量转录作业或实时直播转录临床医生口述的医疗笔记。Batch 转录作业使您能够转录音频文件。您可以在转录工作或直播中指定临床医生的医学专业，以确Amazon Transcribe保 Medical 以尽可能高的准确度生成转录结果。

您可以转录以下专业的医学听写：

- 心脏病学 — 仅以流媒体转录形式提供
- 神经病学 — 仅以流媒体转录形式提供
- 肿瘤学 — 仅以流媒体转录形式提供
- 初级保健 — 包括以下类型的医疗实践：
  - 家庭医疗
  - 内科
  - 妇产科 (OB-GYN)
  - 儿科

- 放射学 — 仅以流媒体转录形式提供
- 泌尿科——仅提供流媒体转录

您可以通过使用自定义词汇来提高转录准确性。有关医学自定义词汇表的工作原理的信息，请参阅使用医学自定义词汇提高转录准确性。

默认情况下，MAmazon Transcribe edical 返回具有最高置信度的转录内容。如果您想将其配置为返回备用转录，请参阅生成替代转录。

有关数字和医学测量值在转录输出中如何显示的信息，请参见转录数字和转录医学术语和测量结果。

主题

- 转录医学听写的音频文件
- 在实时直播中转录医学听写

转录医学听写的音频文件

使用批量转录作业转录医学对话的音频文件。你可以用它来转录临床医生与患者的对话。您可以在StartMedicalTranscriptionJobAPI 或中启动批量转录作业AWS Management Console。

当您使用 StartMedicalTranscriptionJobAPI 开始医学转录作业时，您可以指定PRIMARYCARE为Specialty参数的值。

AWS Management Console

转录临床医生与患者的对话 (AWS Management Console)

要使用转录临床医生与患者的对话，请创建转录作业并选择 “对话” 作为音频输入类型。AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中的 Medicial 下Amazon Transcribe，选择 scrapions。
3. 请选择 Create job (创建任务)。
4. 在指定Job 详细信息页面的作业设置下，指定以下内容。
    a. 名称 – 转录任务的名称。
    b. 音频输入类型-听写
5. 对于其余字段，请指定音频文件的Amazon S3位置以及要存储转录作业输出的位置。

6. 选择 Next (下一步)。
7. 选择创建。

API

使用批量转录作业 (API) 转录医学对话

- 对于 StartMedicalTranscriptionJobAPI，指定以下内容。

  a. 对于MedicalTranscriptionJobName，请指定一个在您的名称中独一无二的名称AWS 账户。

  b. 对于LanguageCode，请指定与您的音频文件中所说的语言和词汇过滤器的语言相对应的语言代码。

  c. 在Media对象的MediaFileUri参数中，指定要转录的音频文件的名称。

  d. 对于Specialty，请指定在音频文件中发言的临床医生的医学专业。

  e. 对于 Type，请指定 DICTATION。

  f. 对于OutputBucketName，指定用于Amazon S3存储转录结果的存储桶。

  以下是使用AWS SDK for Python (Boto3)转录该PRIMARYCARE专业临床医生的医学听写的示例请求。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-med-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-file.flac"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'DICTATION'
)
```

```
while True:
    status = transcribe.get_medical_transcription_job(MedicalTranscriptionJobName =
 job_name)
    if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

以下示例代码显示了医学听写的转录结果。

```
{
    "jobName": "dictation-medical-transcription-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "... came for a follow up visit today..."
            }
        ],
        "items": [
            {
            ...
                "start_time": "4.85",
                "end_time": "5.12",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "came"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "5.12",
                "end_time": "5.29",
                "alternatives": [
                    {
                        "confidence": "1.0",
```

```
                    "content": "for"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "5.29",
            "end_time": "5.33",
            "alternatives": [
                {
                    "confidence": "0.9955",
                    "content": "a"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "5.33",
            "end_time": "5.66",
            "alternatives": [
                {
                    "confidence": "0.9754",
                    "content": "follow"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "5.66",
            "end_time": "5.75",
            "alternatives": [
                {
                    "confidence": "0.9754",
                    "content": "up"
                }
            ],
            "type": "pronunciation"
        },
        {
            "start_time": "5.75",
            "end_time": "6.02",
            "alternatives": [
                {
                    "confidence": "1.0",
```

```
                        "content": "visit"
                    }
                ]
                ...
    },
    "status": "COMPLETED"
}
```

AWS CLI

在批量转录作业中启用扬声器分区 (AWS CLI)

- 运行以下代码。

```
aws transcribe start-medical-transcription-job \
--region us-west-2 \
--cli-input-json file://example-start-command.json
```

以下代码显示的内容example-start-command.json。

```
{
      "MedicalTranscriptionJobName": "my-first-med-transcription-job",
      "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-file.flac"
      },
      "OutputBucketName": "DOC-EXAMPLE-BUCKET",
      "OutputKey": "my-output-files/",
      "LanguageCode": "en-US",
      "Specialty": "PRIMARYCARE",
      "Type": "DICTATION"
}
```

在实时直播中转录医学听写

使用 WebSocket 直播将医疗听写转录为音频流。您也可以使用AWS Management Console将您或其他人所说的语音直接转录到麦克风中。

对于 HTTP/2 或 WebSocket 直播，您可以转录以下医学专业的音频：

- 心脏病学
- 肿瘤学
- 神经病学
- 初级保健
- 放射学
- 泌尿外科

每个医学专业都包括许多类型的手术和预约。因此，临床医生会口述许多不同类型的笔记。使用以下示例作为指导，帮助您指定 WebSocket 请求的`specialty` URI 参数的值或[StartMedicalStreamTranscription](#)API 的`Specialty`参数：

- 要在电生理学或超声心动图检查后进行听写，请选择`CARDIOLOGY`。
- 要在外科肿瘤学或放射肿瘤学手术后听写，请选择`ONCOLOGY`。
- 对于医生口述表明诊断为脑炎的笔记，请选择`NEUROLOGY`。
- 要听取分解膀胱结石的手术说明，请选择`UROLOGY`。
- 要在内科咨询后听取临床医生的笔记，请选择`PRIMARYCARE`。
- 要听取医生传达 CT 扫描、PET 扫描、MRI 或 X 光检查结果的口述，请选择`RADIOLOGY`。
- 要听取妇科咨询后的医生笔记，请选择`PRIMARYCARE`。

要提高实时直播中特定术语的转录准确性，请使用自定义词汇。要启用自定义词汇表，`vocabulary-name`请将的值设置为要使用的自定义词汇表的名称。

使用麦克风将听写转录到麦克风中AWS Management Console

要使用转录医学听写的流媒体音频，请选择转录医学听写的选项，开始直播，然后开始对着麦克风说话。AWS Management Console

转录医学听写的流媒体音频 (AWS Management Console)

1. 登录到 AWS Management Console。
2. 在导航窗格中的 MedicaiAmazon Transcribe al 下，选择 Loaat。
3. 选择 “听写”。
4. 对于医学专业，请选择在直播中发言的临床医生的医学专业。

5. 选择 Start streaming（开始流式传输）。
6. 对着麦克风说话。

在 HTtp/2 流中转录听实例

要转录医疗听写的 HTTP/2 流，请使用 StartMedicalStreamTranscriptionAPI 并指定以下内容：

- LanguageCode— 语言代码。有效值为en-US
- MediaEncoding— 用于输入音频的编码。有效值包括 pcm、ogg-opus 和 flac。
- Specialty— 医疗专业人员的专业。
- Type – DICTATION

有关设置 HTTP/2 流以转录医学听写的更多信息，请参阅设置 HTTP/2 音频流。

使用 WebSocket流媒体请求转录医疗听写

要使用 WebSocket请求在实时直播中转录医疗听写，您需要创建预签名的 URI。该 URI 中包含在应用程序和Amazon Transcribe医疗之间建立音频流所需的信息。有关创建 WebSocket 请求的更多信息，请参阅设置直 WebSocket 播。

使用以下模板来创建预签名的 URI。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/medical-stream-
transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding=flac
&sample-rate=16000
&session-id=sessionId
&specialty=medicalSpecialty
&type=DICTATION
&vocabulary-name=vocabularyName
```

```
&show-speaker-label=boolean
```

有关创建预签名 URI 的更多信息，请参阅设置直 WebSocket 播。

使用医学自定义词汇提高转录准确性

要提高 Medical 中的Amazon Transcribe转录准确性，请创建和使用一个或多个医学自定义词汇。自定义词汇表是特定领域的单词或短语的集合。该集合有助于提高 MedicalAmazon Transcribe 在转录这些单词或短语方面的表现。

使用Amazon Transcribe Medical 时，您应对自己数据的完整性负责。不要在自定义词汇表中输入机密信息、个人信息 (PII) 或受保护的健康信息 (PHI)。

为了获得最佳效果，请创建单独的小型自定义词汇表，每个词汇表都有助于转录特定的录音。与创建一个用于所有录音的大型自定义词汇表相比，转录准确性的提高幅度更大。

默认情况下，您的每个最多可以有 100 个词汇表AWS 账户。自定义词汇表的大小不能超过 50 KB。有关请求增加可以包含的自定义词汇表数量的信息AWS 账户，请参阅AWS服务配额。

自定义词汇表以美国英语 (en-US) 提供。

主题

- 为您的医学自定义词汇创建文本文件
- 使用文本文件创建医学自定义词汇
- 使用医学自定义词汇表转录音频文件
- 使用医学自定义词汇转录实时直播
- Amazon Transcribe医疗用字符集

为您的医学自定义词汇创建文本文件

要创建自定义词汇表，您需要创建 UTF-8 格式的文本文件。在此文件中，创建一个四列表，每列指定一个字段。每个字段都告诉Amazon Transcribe Medical 特定领域的术语是如何发音的，或者如何在转录中显示这些术语。您将包含这些字段的文本文件存储在Amazon S3存储段中。

了解如何格式化您的文本文件

要创建医学自定义词汇表，请将列名作为标题行输入。在标题行下方输入每列的值。

以下是表中四列的名称：

- Phrase— 列为必填项，值为必填项
- IPA— 列为必填项，值可以是可选的
- SoundsLike— 列为必填项，值可以是可选的
- DisplayAs— 列为必填项，值可以是可选的

在创建自定义词汇表时，请务必：

- 用单个 Tab 字符分隔每列。 Amazon Transcribe如果您尝试用空格或多个 Tab 字符分隔列，则会引发一条错误消息。
- 确保列中每个值后面没有尾部空格或空格。

确保为每列输入的值：

- 少于 256 个字符，包括连字符
- 仅使用允许的字符集中的字符，请参见Amazon Transcribe医疗用字符集。

输入表中各列的值

以下信息显示如何为表的四列指定值：

- **Phrase**— 应识别的单词或短语。您必须在该列中输入值。

  如果条目为短语，请用连字符 (-) 隔开各个单词。例如，以 **cerebral-autosomal-dominant-arteriopathy-with-subcortical-infarcts-and-leukoencephalopathy** 形式输入 **cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy**。

  以单个字母后跟圆点的形式（例如 **D.N.A.** 或 **S.T.E.M.I.**），输入字母应单独发音的首字母缩写词或其他单词。要输入首字母缩写词的复数形式（例如“STEMIs”），请使用连字符将“s”与首字母缩写词隔开：“**S.T.E.M.I-s**”。您可以在首字母缩写词中使用大写或小写字母。

  Phrase 列是必填的。您可以使用输入语言允许的任何字符。有关允许的字符，请参阅Amazon Transcribe医疗用字符集。如果您未指定该DisplayAs列，MedAmazon Transcribe ical 将在输出文件中使用该Phrase列的内容。
- **IPA**（必填列，值可以是可选的）-要指定单词或短语的发音，可以在此列中包括国际语音字母 (IPA) 中的字符。IPA 列不能包含前导空格或尾随空格，

并且您必须使用单个空格以隔开输入中的每个音素。例如，在英语中，您以 **ə k j u t # # s p # # ə t # # i d # s t # # s s # n d # o# m** 形式输入短语 **acute-respiratory-distress-syndrome**。您以 **e# # l # l** 形式输入短语 **A.L.L.**。

即使未指定 IPA 列的内容，您也必须包含空白的 IPA 列。如果在 IPA 列中包含值，则不能为 SoundsLike 列提供值。

有关特定语言允许的 IPA 字符列表，请参阅Amazon Transcribe医疗用字符集。美国英语是Amazon Transcribe医学中唯一可用的语言。

- **SoundsLike**（列为必填项，值可以是可选的）— 您可以将单词或短语分成较小的部分，并使用该语言的标准拼写法为每个段提供发音，以模仿单词的发音。例如，您可以按以下方式为短语 **cerebral-autosomal-dominant-arteriopathy-with-subcortical-infarcts-and-leukoencephalopathy** 提供发音提示：**sir-e-brul-aut-o-som-ul-dah-mi-nant-ar-ter-ri-o-pa-thy-with-sub-cor-ti-cul-in-farcts-and-lewk-o-en-ce-phul-ah-pu-thy**。短语 **atrioventricular-nodal-reentrant-tachycardia** 的提示如下所示：**ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia**。您使用连字符 (-) 分隔提示的每个部分。

  即使没有为 SoundsLike 列提供值，您也必须包含空白的 SoundsLike 列。如果在 SoundsLike 列中包含值，则不能为 IPA 列提供值。

  您可以使用输入语言允许的任何字符。有关允许的字符列表，请参阅Amazon Transcribe医疗用字符集。

- **DisplayAs**（必填列，值可以是可选的）-定义单词或短语在输出时的外观。例如，如果单词或短语为 **cerebral-autosomal-dominant-arteriopathy-with-subcortical-infarcts-and-leukoencephalopathy**，您可以将显示形式指定为 cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy，以便不会显示连字符。如果要在输出中显示首字母缩写词而不是完整术语，您也可以使用 CADASIL 形式指定 DisplayAs。

  如果您未指定该DisplayAs列，MedAmazon Transcribe icalPhrase 将在输出中使用输入文件中的列。

  您可以在 DisplayAs 列中使用任何 UTF-8 字符。

只能为IPA和DisplayAs列中的值添加空格。

要创建自定义词汇表的文本文件，请将文本文件中的每个单词或短语放在单独的行中。使用制表符分隔列。仅在 IPA 和 DisplayAs 列中包含空格值。将带有扩展名的文件保存在Amazon S3存储桶.txt中，与使用 MAmazon Transcribe edAWS 区域 ical 创建自定义词汇表的存储桶相同。

如果您在 Windows 中编辑文本文件，请确保文件采用 LF 格式，而不是 CRLF 格式。否则，您无法创建自定义词汇表。通过使用某些文本编辑器，您可以使用查找和替换命令更改格式。

以下示例显示可用于创建自定义词汇表的文本。要从这些示例中创建自定义词汇表，请将一个示例复制到文本编辑器中，将 [TAB] 替换为制表符，然后将保存的文本文件上传到 Amazon S3。

```
Phrase[TAB]IPA[TAB]SoundsLike[TAB]DisplayAs
acute-respiratory-distress-syndrome[TAB][TAB][TAB]acute respiratory distress syndrome
A.L.L.[TAB]e# # l # l[TAB][TAB]ALL
atrioventricular-nodal-reentrant-tachycardia[TAB][TAB]ay-tree-o-ven-trick-u-lar-node-
al-re-entr-ant-tack-ih-card-ia[TAB]
```

您可以按任意顺序输入列。以下示例显示了自定义词汇表输入文件的其他有效结构。

```
Phrase[TAB]SoundsLike[TAB]IPA[TAB]DisplayAs
acute-respiratory-distress-syndrome[TAB][TAB][TAB]acute respiratory distress syndrome
A.L.L.[TAB][TAB]e# # l # l[TAB]ALL
atrioventricular-nodal-reentrant-tachycardia[TAB]ay-tree-o-ven-trick-u-lar-node-al-re-
entr-ant-tack-ih-card-ia[TAB][TAB]
```

```
DisplayAs[TAB]SoundsLike[TAB]IPA[TAB]Phrase
acute respiratory distress syndrome[TAB][TAB][TAB]acute-respiratory-distress-syndrome
ALL[TAB][TAB]e# # l # l[TAB]A.L.L.
[TAB]ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia[TAB]
[TAB]atrioventricular-nodal-reentrant-tachycardia
```

为了便于阅读，下表以 html 格式更清楚地显示上述示例。它们只是为了说明这些例子。

Phrase	IPA	SoundsLike	DisplayAs
acute-respiratory-distress-syndrome			acute respiratory distress syndrome

Phrase	IPA	SoundsLike	DisplayAs
A.L.L。	eɪ ɛ l ɛ l		ALL
atrioventricular-nodal-reentrant-tachycardia		ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia	

Phrase	SoundsLike	IPA	DisplayAs
acute-respiratory-distress-syndrome			acute respiratory distress syndrome
atrioventricular-nodal-reentrant-tachycardia	ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia		
A.L.L。		eɪ ɛ l ɛ l	ALL

DisplayAs	SoundsLike	IPA	Phrase
acute respiratory distress syndrome			acute-respiratory-distress-syndrome
ALL		eɪ ɛ l ɛ l	A.L.L。
	ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia		atrioventricular-nodal-reentrant-tachycardia

使用文本文件创建医学自定义词汇

要创建自定义词汇表，您必须准备好一个包含单词或短语集合的文本文件。 Amazon TranscribeMedical 使用此文本文件创建自定义词汇表，可使用此词汇表提高这些单词或短语的转录准

确度。您可以使用 CreateMedicalVocabularyAPI 或 MAmazon Transcribe edical 控制台创建自定义词汇表。

AWS Management Console

要使用创建自定义词汇表，您需要提供包含您的单词或短语的文本文件的Amazon S3 URI。AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中的 Medical 项下Amazon Transcribe，选择自定义词汇表。
3. 对于 “名称”，在 “词汇设置” 下，为您的自定义词汇表选择一个名称。
4. 指定音频文件或视频文件在 Amazon S3 中的位置：
   - 对于词汇设置下的 S3 上的词汇输入文件位置，指定用于识别您将用于创建自定义词汇表的文本文件的Amazon S3 URI。
   - 对于词汇输入文件在 S3 中的位置，选择 Browse S3 浏览文本文件并将其选中。
5. 选择 “创建词汇表”。

您可以在中查看自定义词汇表的处理状态AWS Management Console。

API

创建医学自定义词汇 (API)

- 对于 StartTranscriptionJobAPI，指定以下内容。
  a. 对于 LanguageCode，请指定 en-US。
  b. 对于VocabularyFileUri，请指定用于定义自定义词汇表的文本文件Amazon S3的位置。
  c. 对于VocabularyName，请为您的自定义词汇表指定一个名称。您指定的名称在您的中必须是唯一的AWS 账户。

要查看自定义词汇表的处理状态，请使用 GetMedicalVocabularyAPI。

以下是使用此词汇表AWS SDK for Python (Boto3)创建自定义词汇表。

```
from __future__ import print_function
import time
```

```
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
vocab_name = "my-first-vocabulary"
response = transcribe.create_medical_vocabulary(
    VocabularyName = job_name,
    VocabularyFileUri = 's3://DOC-EXAMPLE-BUCKET/my-vocabularies/my-vocabulary-
table.txt'
    LanguageCode = 'en-US',
  )

while True:
    status = transcribe.get_medical_vocabulary(VocabularyName = vocab_name)
    if status['VocabularyState'] in ['READY', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

AWS CLI

在批量转录作业中启用扬声器分区 (AWS CLI)

- 运行以下代码。

```
aws transcribe create-medical-vocabulary \
--vocabulary-name my-first-vocabulary \
--vocabulary-file-uri s3://DOC-EXAMPLE-BUCKET/my-vocabularies/my-vocabulary-
file.txt \
--language-code en-US
```

使用医学自定义词汇表转录音频文件

使用StartMedicalTranscriptionJob或启动AWS Management Console转录作业，使用自定义词汇来提高转录准确性。

AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中的 MedicaAmazon Transcribe l 项下，选择转录作业。
3. 请选择 Create job (创建任务)。

4. 在指定作业详细信息页面上，提供有关您的转录作业的信息。
5. 选择下一步。
6. 在 “自定义” 下，启用 “自定义词汇”。
7. 在 “词汇选择” 下，选择一个自定义词汇。
8. 选择创建。

API

使用批量转录作业 (API) 在音频文件中启用扬声器分区

- 对于 StartMedicalTranscriptionJobAPI，指定以下内容。
  a. 对于MedicalTranscriptionJobName，请指定一个在您的名称中是唯一的AWS 账户。
  b. 对于LanguageCode，请指定与您的音频文件中所说的语言和词汇过滤器的语言相对应的语言代码。
  c. 对于Media对象的MediaFileUri参数，指定要转录的音频文件的名称。
  d. 对于Specialty，请指定在音频文件中发言的临床医生的医学专业。
  e. 对于Type，请指定音频文件是对话还是听写。
  f. 对于OutputBucketName，指定用于Amazon S3存储转录结果的存储桶。
  g. 对于Settings对象，指定以下内容。
     - VocabularyName— 您的自定义词汇表的名称。

以下请求使用使用自定义词汇表启动批量转录作业。AWS SDK for Python (Boto3)

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-med-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_medical_transcription_job(
   MedicalTranscriptionJobName = job_name,
   Media = {
       'MediaFileUri': job_uri
   },
```

```
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    Settings = {
        'VocabularyName': 'example-med-custom-vocab'
        }
 )

while True:
   status = transcribe.get_medical_transcription_job(MedicalTranscriptionJobName =
 job_name)
   if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

使用医学自定义词汇转录实时直播

为了提高实时流中的转录准确性，您可以使用 HTTP/2 或 stre WebSocket ams 使用自定义词汇。要启动 HTTP/2 请求，请使用 StartMedicalStreamTranscriptionAPI。您可以使用AWS Management Console、StartMedicalStreamTranscriptionAPI 或使用 WebSocket 协议实时使用自定义词汇表。

将听写内容转录到麦克风中 (AWS Management Console)

要使用转录医学听写的流媒体音频，请选择转录医学听写的选项，开始直播，然后开始对着麦克风说话。AWS Management Console

转录医学听写的流媒体音频 (AWS Management Console)

1. 登录到 AWS Management Console。
2. 在导航窗格中的 MAmazon Transcribe edical 项下，选择实时转录。
3. 对于医学专业，请选择在直播中发言的临床医生的医学专业。
4. 对于音频输入类型，选择 “对话” 或 “听写”。
5. 对于其他设置，请选择自定义词汇。

- 要选择词汇，请选择自定义词汇。

6. 选择 Start streaming（开始流式传输）。
7. 对着麦克风说话。

在 HTTP/2 流中启用扬声器分区

以下是 HTTP/2 请求参数的语法。

```
POST /medical-stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
authorization: Generated value
x-amz-target: com.amazonaws.transcribe.Transcribe.StartMedicalStreamTranscription
x-amz-content-sha256: STREAMING-MED-AWS4-HMAC-SHA256-EVENTS
x-amz-date: 20220208T235959Z
x-amzn-transcribe-session-id: my-first-http2-med-stream
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-transcribe-vocabulary-name: my-first-med-vocab
x-amzn-transcribe-specialty: PRIMARYCARE
x-amzn-transcribe-type: CONVERSATION
x-amzn-transcribe-show-speaker-label: true
Content-type: application/vnd.amazon.eventstream
transfer-encoding: chunked
```

参数描述：

- 主机：使用AWS 区域您正在调用的AWS 区域（前面的示例中的 “us-west-2”）进行更新。有关有效列表AWS 区域，请参阅AWS 区域和终端节点。
- 授权：这是一个生成的字段。要了解有关创建签名的更多信息，请参阅使用签名版本 4 签署AWS请求。
- x-amz-target: 请勿更改此字段；请使用前面示例中显示的内容。
- x-amz-content-sha256：这是生成的字段。要了解有关计算签名的更多信息，请参阅使用签名版本 4 签署AWS请求。
- x-amz-date：创建签名的日期和时间。格式为 YYYYMDDTHHMSSZ，其中 yyyy=YEAR、mm=month、dd=day、HH=Hour、mm=minute、ss=seconds 以及 'T' 和 'Z' 是固定字符。有关更多信息，请参阅签名版本 4 中的处理日期。
- x-amzn-transcribe-session-id：您的流媒体会话的名称。

- x-amzn-transcribe-language-code：用于输入音频的编码。有关有效值支持的语言和特定语言的特征的列表，请参阅StartMedicalStreamTranscription或。
- x-amzn-transcribe-media-enco ding：用于输入音频的编码。有效值包括 pcm、ogg-opus 和 flac。
- x-amzn-transcribe-sample-r ate：输入音频的采样率（以赫兹为单位）。 Amazon Transcribe支持从 8,000 Hz 到 48,000 Hz 的范围。低质量音频，例如电话音频，通常在 8,000 Hz 左右。高质量音频的范围通常介于 16,000 Hz 到 48,000 Hz 之间。请注意，您指定的采样率必须与音频的采样率相匹配。
- x-amzn-transcribe-vocabulary-nam e：您想在转录中使用的词汇的名称。
- x-amzn-transcribe-specialty: 医学专业正在转录。
- x-amzn-transcribe-type：选择这是听写还是对话。
- x-amzn-transcribe-show-speaker-label：要启用降序功能，此值必须为true。
- c@@ ontent-type：请勿更改此字段；请使用前面示例中显示的内容。

在 WebSocket请求中启用扬声器分区

要使用 API 对 WebSocket 直播中的发言者进行分区，请使用以下格式创建预签名 URI 以启动 WebSocket 请求并设置为vocabulary-name自定义词汇表的名称。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/medical-stream-
transcription-websocket
?language-code=en-US
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding=flac
&sample-rate=16000
&session-id=sessionId
&specialty=medicalSpecialty
&type=CONVERSATION
&vocabulary-name=vocabularyName
&show-speaker-label=boolean
```

Amazon Transcribe医疗用字符集

要在 Medical 中Amazon Transcribe使用自定义词汇，请使用以下字符集。

英语字符集

对于英语自定义词汇表，您可以在 Phrase 和 SoundsLike 列中使用以下字符：

- a - z
- A - Z
- ' (撇号)
- - (连字符)
- . (句点)

您可以在词汇表输入文件的 IPA 列中使用以下国际音标字母 (IPA) 字符。

字符	代码	字符	代码
aʊ	0061 028A	w	0077
aɪ	0061 026A	z	007A
b	0062	æ	00E6
d	0064	ð	00F0
eɪ	0065 026A	ŋ	014B
f	0066	ɑ	0251
g	0067	ɔ	0254
h	0068	ɔɪ	0254 026A
i	0069	ə	0259
j	006A	ɛ	025B
k	006B	ɝ	025D

字符	代码	字符	代码
l	006C	g	0261
l̩	006C 0329	ɪ	026A
m	006D	ɹ	0279
n	006E	ʃ	0283
n̩	006E 0329	ʊ	028A
oʊ	006F 028A	ʌ	028C
p	0070	ʍ	028D
s	0073	ʒ	0292
t	0074	ʤ	02A4
u	0075	ʧ	02A7
v	0076	θ	03B8

识别转录中的个人健康信息 (PHI)

使用个人Health 信息识别在转录结果中标记个人健康信息 (PHI)。通过查看标签，您可以找到可用于识别患者的 PHI。

您可以使用实时流或批量转录作业识别 PHI。

您可以使用自己的后处理来编辑转录输出中识别的 PHI。

使用个人Health 信息识别来识别以下类型的 PHI：

- 个人PHI:
  - 姓名 — 全名或姓氏和首字母缩写
  - 性别
  - 年龄
  - 电话号码

  - 与患者直接相关的日期（不包括年份）
  - 电子邮件地址
- 地理PHI:
  - 物理地址
  - 邮政编码
  - 医疗中心或诊所的名称
- 账户 PHI：
  - 传真号码
  - 社会安全号码 (SSN)
  - Health 保险受益人号码
  - 账号
  - 证书或许可证号
- 车辆 PHI：
  - 车辆识别号码 (VIN, Vehicle Identification Number)
  - 车牌号
- 其他 PHI:
  - Web 统一资源位置 (URL)
  - 互联网协议 (IP) 地址号

Amazon Transcribe医疗保险流通与责任法案 (HIPAA A) 有关更多信息，请参阅Amazon Transcribe Medical：有关识别音频文件中的 PHI 的信息，请参阅识别音频文件中的 PHI。有关在数据流中识别 PHI 的信息，请参阅在实时直播中识别 PHI。

主题

- 识别音频文件中的 PHI
- 在实时直播中识别 PHI

识别音频文件中的 PHI

使用批量转录作业转录音频文件并识别其中的个人健康信息 (PHI)。当您激活个人Health 信息 (PHI) 识别时，Amazon Transcribe医疗会对其在转录结果中识别的 PHI 进行标记。有关 MAmazon Transcribe edical 可以识别的 PHI 的信息，请参阅识别转录中的个人健康信息 (PHI)。

您可以使用 StartMedicalTranscriptionJobAPI 或启动批量转录作业AWS Management Console。

AWS Management Console

要使用转录临床医生与患者的对话，请创建转录作业并选择 “对话” 作为音频输入类型。AWS Management Console

转录音频文件并识别其 PHI (AWS Management Console)

1. 登录到 AWS Management Console。
2. 在导航窗格上的 MAmazon Transcribe ICAL MICA L
3. 请选择 Create job (创建任务)。
4. 在指定Job 详细信息页面的作业设置下，指定以下内容。
   a. 名称 — 您的专属转录作业的名称AWS 账户。
   b. 音频输入类型 — 对话或听写。
5. 对于其余字段，请指定音频文件的Amazon S3位置以及要存储转录作业输出的位置。
6. 选择下一步。
7. 在 “音频设置” 下，选择 PHI 识别。
8. 选择创建。

API

使用批量转录作业 (API) 转录音频文件并识别其 PHI

- 对于 StartMedicalTranscriptionJobAPI，指定以下内容。
  a. 对于MedicalTranscriptionJobName，请指定一个专属于您的名称AWS 账户。
  b. 对于LanguageCode，请指定与您的音频文件中所说的语言相对应的语言代码。
  c. 对于Media对象的MediaFileUri参数，指定要转录的音频文件的名称。
  d. 对于Specialty，将音频文件中讲话的临床医生的医学专业指定为PRIMARYCARE。
  e. 对于 Type，请指定 CONVERSATION 或 DICTATION。
  f. 对于OutputBucketName，指定要将转录结果存储到的结果存储Amazon S3桶。

  以下是使用转录音频文件并识别患者的 PHI 的示例请求。AWS SDK for Python (Boto3)

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-file.flac"
transcribe.start_medical_transcription_job(
      MedicalTranscriptionJobName = job_name,
      Media = {'MediaFileUri': job_uri},
      LanguageCode = 'en-US',
      ContentIdentificationType = 'PHI',
      Specialty = 'PRIMARYCARE',
      Type = 'type', # Specify 'CONVERSATION' for a medical conversation. Specify
 'DICTATION' for a medical dictation.
      OutputBucketName = 'DOC-EXAMPLE-BUCKET'
  )
while True:
    status = transcribe.get_medical_transcription_job(MedicalTranscriptionJobName =
 job_name)
    if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

以下示例代码显示了识别患者 PHI 后的转录结果。

```
{
    "jobName": "my-medical-transcription-job-name",
    "accountId": "111122223333",
    "results": {
        "transcripts": [{
            "transcript": "The patient's name is Bertrand."
        }],
        "items": [{
            "start_time": "0.0",
            "end_time": "0.37",
```

```
            "alternatives": [{
                "confidence": "0.9993",
                "content": "The"
            }],
            "type": "pronunciation"
        }, {
            "start_time": "0.37",
            "end_time": "0.44",
            "alternatives": [{
                "confidence": "0.9981",
                "content": "patient's"
            }],
            "type": "pronunciation"
        }, {
            "start_time": "0.44",
            "end_time": "0.52",
            "alternatives": [{
                "confidence": "1.0",
                "content": "name"
            }],
            "type": "pronunciation"
        }, {
            "start_time": "0.52",
            "end_time": "0.92",
            "alternatives": [{
                "confidence": "1.0",
                "content": "is"
            }],
            "type": "pronunciation"
        }, {
            "start_time": "0.92",
            "end_time": "0.9989",
            "alternatives": [{
                "confidence": "1.0",
                "content": "Bertrand"
            }],
            "type": "pronunciation"
        }, {
            "alternatives": [{
                "confidence": "0.0",
                "content": "."
            }],
            "type": "punctuation"
        }],
```

```
        "entities": [{
            "content": "Bertrand",
            "category": "PHI*-Personal*",
            "startTime": 0.92,
            "endTime": 1.2,
            "confidence": 0.9989
        }],
    },
    "status": "COMPLETED"
}
```

AWS CLI

使用批量转录作业转录音频文件并识别 PHI (AWS CLI)

- 运行以下代码。

```
aws transcribe start-medical-transcription-job \
--medical-transcription-job-name my-medical-transcription-job-name\
--language-code en-US \
--media MediaFileUri="s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-file.flac" \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--specialty PRIMARYCARE \
--type type \ # Choose CONVERSATION to transcribe a medical conversation.
 Choose DICTATION to transcribe a medical dictation.
--content-identification-type PHI
```

在实时直播中识别 PHI

您可以在 HTTP/2 或 WebSocket 直播中识别个人Health 信息 (PHI)。当您激活 PHI 识别时，Amazon TranscribeMedical 会对其在转录结果中识别的 PHI 进行标记。有关 MAmazon Transcribe edical 可以识别的 PHI 的信息，请参阅识别转录中的个人健康信息 (PHI)。

在对着麦克风说出的听写中识别 PHI

要使用转录麦克风拾取的语音并识别任何 PHI，请选择 D ictation 作为音频输入类型，开始直播，然后开始对着计算机上的麦克风说话。AWS Management Console

要在听写中识别 PHI，请使用AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格中，选择 Real-time transcription (实时转录)。
3. 对于 “音频输入类型”，选择 “听写”。
4. 对于其他设置，请选择 PHI 识别。
5. 选择 “开始直播”，然后对着麦克风讲话。
6. 选择 “停止直播” 以结束听写。

在 HTTP/2 流中的 P/2 流中识别PHI

要在激活 PHI 识别的情况下启动 HTTP/2 数据流，请使用 StartMedicalStreamTranscriptionAPI 并指定以下内容：

- 对于LanguageCode，指定流中使用的语言的语言代码。对于美式英语，请指定en-US。
- 对于MediaSampleHertz，指定音频的采样率。
- 对于 content-identification-type，请指定 PHI。

识别 WebSocket 直播中的 PHI

要在激活 PHI 识别的情况下开始 WebSocket 直播，请使用以下格式创建预签名 URL。

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/medical-stream-
transcription-websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
&specialty=medical-specialty
&content-identification-type=PHI
```

参数定义可以在 API 参考中找到；所有AWS API 操作的通用参数列在 “通用参数” 部分中。

生成替代转录

当你使用 MAmazon Transcribe edical 时，你得到的转录可信度最高。但是，您可以将 MAmazon Transcribe edical 配置为以较低的置信度返回其他转录。

使用备选转录可查看对所转录音频的不同解释。例如，在允许用户查看转录的应用程序中，您可以提供备选转录供用户选择。

您可以使用AWS Management Console或 StartMedicalTranscriptionJobAPI 生成替代转录。

AWS Management Console

要使用生成备用转录，请在配置作业时启用替代结果。AWS Management Console

1. 登录到 AWS Management Console。
2. 在导航窗格上的 LOAAmazon Transcribe D 下，选择 Target。
3. 请选择 Create job (创建任务)。
4. 在指定作业详细信息页面上，提供有关您的转录作业的信息。
5. 选择 Next（下一步）。
6. 启用替代结果。
7. 对于最大备选方案，输入介于 2 到 10 之间的整数值，以表示输出中想要的最大备选转录数。
8. 选择 Create（创建）。

API

使用批量转录作业 (API) 在音频文件中分隔每个发言者的文本

- 对于 StartMedicalTranscriptionJobAPI，指定以下内容。
    a. 对于MedicalTranscriptionJobName，请指定一个在您的名称中是唯一的AWS 账户。
    b. 对于LanguageCode，请指定与您的音频文件中所说的语言和词汇过滤器的语言相对应的语言代码。
    c. 在Media对象的MediaFileUri参数中，指定要转录的音频文件的位置。
    d. 对于Specialty，请指定在音频文件中发言的临床医生的医学专业。
    e. 对于Type，请指定你是在转录医学对话还是听写。
    f. 对于OutputBucketName，指定用于Amazon S3存储转录结果的存储桶。

g. 对于Settings对象，指定以下内容。

i. ShowAlternatives – true.

ii. MaxAlternatives-介于 2 到 10 之间的整数，表示转录输出中想要的替代转录数量。

以下请求使用启动AWS SDK for Python (Boto3)转录作业，该作业最多生成两个备用转录。

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-file.flac
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    Settings = {
        'ShowAlternatives': True,
        'MaxAlternatives': 2
    }
)

while True:
   status = transcribe.get_medical_transcription_job(MedicalTranscriptionJobName =
 job_name)
   if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
 'FAILED']:
       break
   print("Not ready yet...")
   time.sleep(5)
print(status)
```

AWS CLI

在音频文件中转录初级保健临床医生与患者对话的音频文件 (AWS CLI)

- 运行以下代码。

```
aws transcribe start-transcription-job \
--cli-input-json file://filepath/example-start-command.json
```

以下代码显示example-start-command.json。

```
{
      "MedicalTranscriptionJobName": "my-first-transcription-job",
      "LanguageCode": "en-US",
      "Specialty": "PRIMARYCARE",
      "Type": "CONVERSATION",
      "OutputBucketName":"DOC-EXAMPLE-BUCKET",
      "Media": {
          "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-audio-
file.flac"
        },
      "Settings":{
          "ShowAlternatives": true,
          "MaxAlternatives": 2
        }
}
```

Amazon Transcribe医疗和接口 VPC 终端节点 (AWS PrivateLink)

您可以在您的 VPC 和之间建立私有连接Amazon Transcribe通过创建一个来进行医疗接口 VPC 端点。接口端点由以下各方提供动力AWS PrivateLink，一项使您能够私下访问的技术Amazon Transcribe没有互联网网关、NAT 设备、VPN 连接的医疗 API 或AWS Direct Connect连接。您的 VPC 中的实例不需要与公有 IP 地址进行通信Amazon Transcribe医疗原料药。您的 VPC 和之间的流量Amazon Transcribe医疗不会离开亚马逊网络。

每个接口端点均由子网中的一个或多个弹性网络接口表示。

有关更多信息，请参见接口 VPC 端点 (AWS PrivateLink)在Amazon VPC用户指南。

的注意事项Amazon Transcribe医疗 VPC 端点

在为接口设置接口 VPC 终端节点之前Amazon Transcribe医疗，请务必复查接口端点属性和限制在Amazon VPC用户指南。

Amazon TranscribeMedical 支持从您的 VPC 调用其所有 API 操作。

为创建接口 VPC 终端节点Amazon Transcribe医疗

您可以为创建一个 VPC 终端节点Amazon Transcribe使用以下任一方式提供医疗服务AWS Management Console或者AWS CLI。有关更多信息，请参见创建接口终端节点在Amazon VPC用户指南。

用于批量转录Amazon TranscribeMedical，使用以下服务名称创建 VPC 终端节点：

- com.amazonaws。*us-west-2*. 转录

用于流式转录Amazon TranscribeMedical，使用以下服务名称创建 VPC 终端节点：

- com.amazonaws。*us-west-2*. 转录直播

如果您为终端节点启用私有 DNS，则可以向 API 发出 API 请求Amazon TranscribeMedical 使用其默认 DNS 名称为AWS 区域，例如，`transcribestreaming.us-east-2.amazonaws.com`。

有关更多信息，请参见通过接口端点访问服务在Amazon VPC用户指南。

为创建 VPC 终端节点策略Amazon Transcribe医疗直播

您可以将终端节点策略附加到您的 VPC 终端节点，以控制对的访问Amazon Transcribe医疗。该策略指定以下信息：

- 可执行操作的委托人。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅 Amazon VPC 用户指南中的使用 VPC 端点控制对服务的访问。

示例：的 VPC 终端节点策略Amazon Transcribe医疗直播转录操作

以下是中用于流式转录的终端节点策略的示例Amazon Transcribe医疗。连接到终端节点时，此策略授予对所列终端节点的访问权限Amazon Transcribe在所有资源上对所有校长采取医疗行动。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartMedicalStreamTranscription",
         ],
         "Resource":"*"
      }
   ]
}
```

示例：的 VPC 终端节点策略Amazon Transcribe医疗批量转录操作

以下是批量转录的端点策略示例Amazon Transcribe医疗。连接到终端节点时，此策略授予对所列终端节点的访问权限Amazon Transcribe在所有资源上对所有校长采取医疗行动。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartMedicalTranscriptionJob"
         ],
         "Resource":"*"
      }
   ]
}
```

共享子网

您无法在与您共享的子网中创建、描述、修改或删除 VPC 终端节点。但是，您可以在与您共享的子网中使用 VPC 端点。有关 VPC 共享的信息，请参阅与其他账户共享您的 VPC在Amazon Virtual Private Cloud指南。

AWS HealthScribe

AWS HealthScribe 是一项符合 HIPAA 资格的新型机器学习 (ML) 功能，它结合了语音识别和生成式人工智能，可转录患者与临床医生的对话并生成易于查看的临床笔记。AWSHealthScribe 有助于医疗保健软件供应商构建临床应用程序，从而减轻文档管理负担并改善咨询体验。该服务可自动提供丰富的对话转录、识别发言者的角色、对对话进行分类、提取医学术语并生成初步的临床笔记。AWSHealthScribe 结合了这些功能，无需集成和优化单独的人工智能服务，从而使您能够加快实施速度。

常见使用案例：

- 缩短文档记录时间 - 让临床医生能够使用人工智能生成的临床笔记快速完成临床文档记录，这些临床笔记易于在您的应用程序中查看、调整和完成。
- 提高医疗抄写员效率 - 为医疗抄写员配备人工智能生成的转录和临床笔记以及咨询音频，以缩短文档记录周转时间。
- 高效的患者就诊回顾 - 打造这样一种体验：使用户能够在您的应用程序中快速回忆他们对话的关键要点。

> **Important**
>
> AWS HealthScribe 生成的结果是概率性的，由于各种因素的影响，可能并非始终是准确的，这些因素包括音频质量、背景噪声、发言者说话清晰程度、医学术语复杂性、上下文特定的语言细微差别以及机器学习和生成式人工智能特性。AWSHealthScribe 旨在为临床医生和医疗抄写员提供帮助。AWS只应在患者护理场景中使用 HealthScribe 输出，包括但不限于作为电子健康记录的一部分，并在使用之前由经过训练的医疗专业人员进行准确性审查和合理的医疗判断。AWSHealthScribe 输出不能替代专业的医疗建议、诊断或治疗，也不能用于治愈、治疗、缓解、预防或诊断任何疾病或健康状况。

AWS HealthScribe 在责任共担模式下运营，AWS 负责保护运行 AWS HealthScribe 的基础设置，而您则负责管理您的数据。有关更多信息，请参阅责任共担模式。

AWS HealthScribe 在美国东部（弗吉尼亚州北部）区域提供。

该服务提供美国英语 (en-US) 版本。为了获得最佳效果，请使用无损音频格式，如 FLAC 或采用 PCM 16 位编码的 WAV。AWSHealthScribe 支持 16000 Hz 或更高的采样率。

AWS HealthScribe 目前支持内科和骨科。

AWS HealthScribe 作业分析医疗咨询以生成两个 JSON 输出文件：转录文件和临床文档文件。

在转录文件中，除了带有单词级时间戳的标准逐向转录输出外，AWS HealthScribe 还为您提供：

- 参与者角色检测，这样您就可以在对话转录中将患者与临床医生区分开来。
- 转录划分，根据转录对话的临床相关性（例如闲聊、主观、客观等）对转录对话进行分类。这可以用来显示转录的特定部分。
- 临床实体，包括对话中提到的药物、医疗状况和治疗等结构化信息。

在临床文档文件中，AWS HealthScribe 为您提供：

- 摘要包含临床文档关键部分的摘要说明，例如主诉、现病史、系统回顾、既往病史、评估和计划。
- 证据链接将 AI 生成的摘要中使用的每个句子链接到原始咨询转录，以使用户更轻松地在应用程序中验证摘要的准确性。

特定于 AWS HealthScribe 的 API 操作：

- StartMedicalScribeJob
- ListMedicalScribeJobs
- GetMedicalScribeJob
- DeleteMedicalScribeJob

要查看示例 AWS HealthScribe 请求，请参阅启动 AWS HealthScribe 作业。

转录文件

转录文件以逐向格式提供对话内容。

此外，还为每一次对话提供了以下见解：

- 参与者角色 - 每位参与者都被标记为临床医生或患者。如果对话的每个类别中有多个参与者，则会为每个参与者分配一个号码。例如：`CLINICIAN_1`、`CLINICIAN_2` 和 `PATIENT_1`、`PATIENT_2`。
- 部分 - 根据识别的内容，将每个对话轮次分配给 4 个可能的部分之一。
  - 主观 - 患者提供的健康问题相关信息。

- 客观 - 临床医生通过体检、实验室检查、影像学或诊断测试观察到的信息。
- 评估和计划 - 与医生的评估和治疗计划相关的信息。
- 就诊流程管理 - 与闲聊或过渡性对话相关的信息。

- 见解 - 提取对话中存在的临床相关实体 (`ClinicalEntity`)。AWSHealthScribe 检测 Amazon Comprehend Medical 支持的所有临床实体。

有关更详细的输出信息，请参阅示例转录输出。

临床文档文件

文档见解文件包含临床文档的以下关键部分的摘要。

部分	描述
主诉	简要描述患者去看临床医生的原因。
现病史	提供患者疾病信息的笔记，包括严重程度、发作情况、症状出现时间、当前治疗方法和受影响区域。
系统审查	患者报告的不同身体系统症状的评估。
既往病史	详细介绍患者以前的医疗状况、手术和治疗情况。
评测	提供临床医生对患者健康信息评测的笔记。
计划	提及任何医疗治疗、生活方式调整和进一步预约的笔记。

`Summary` 中的每句话都包含对原始咨询转录的引用，使用户可以更轻松地在应用程序中验证摘要的准确性。为 AI 生成的见解提供可追溯性和透明度符合负责任的 AI 原则，例如可解释性。向临床医生或医学抄写员提供这些参考文献以及摘要说明有助于培养信任并鼓励在临床环境中安全地使用人工智能。

`Summary` 中 `EvidenceLinks` 随附的每句话都为总结的转录中的相关对话提供 `SegmentId`。

有关更详细的输出信息，请参阅示例临床文档输出。

启动 AWS HealthScribe 作业

您可以使用 AWS CLI 或 AWS SDK 启动 AWS HealthScribe 作业；有关示例，请参阅以下内容。

AWS CLI

该示例使用 start-medical-scribe-job 命令。有关更多信息，请参阅 StartMedicalScribeJob。

```
aws transcribe start-medical-scribe-job \
--region us-west-2 \
--medical-scribe-job-name my-first-medical-scribe-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--DataAccessRoleArn=arn:aws:iam::111122223333:role/ExampleRole \
--settings ShowSpeakerLabels=false,ChannelIdentification=true \
--channel-definitions ChannelId=0,ParticipantRole=CLINICIAN
 ChannelId=1,ParticipantRole=PATIENT
```

以下是另一个使用 start-medical-scribe-job 命令的示例，以及带有其它设置的请求正文。

```
aws transcribe start-medical-scribe-job \
--region us-west-2 \
--cli-input-json file://filepath/my-first-medical-scribe-job.json
```

my-first-medical-scribe-job.json 文件包含以下请求正文。

```
{
  "MedicalScribeJobName": "my-first-medical-scribe-job",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
   },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole",
  "Settings": {
    "ShowSpeakerLabels": false,
    "ChannelIdentification": true
  },
```

```
  "ChannelDefinitions": [
    {
      "ChannelId": 0,
      "ParticipantRole":"CLINICIAN"
    }, {
      "ChannelId": 1,
      "ParticipantRole":"PATIENT"
    }
  ]
}
```

AWS SDK for Python (Boto3)

以下示例使用 AWS SDK for Python (Boto3) 发出 start_medical_scribe_job 请求。有关更多信息，请参阅 StartMedicalScribeJob。

```
from __future__ import print_functionimport timeimport boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-medical-scribe-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_medical_scribe_job(
    MedicalScribeJobName = job_name,
    Media = {
      'MediaFileUri': job_uri
    },
    OutputBucketName = 'DOC-EXAMPLE-BUCKET',
    DataAccessRoleArn = 'arn:aws:iam::111122223333:role/ExampleRole',
    Settings = {
      'ShowSpeakerLabels': false,
      'ChannelIdentification': true
    },
    ChannelDefinitions = [
      {
        'ChannelId': 0,
        'ParticipantRole': 'CLINICIAN'
      }, {
        'ChannelId': 1,
        'ParticipantRole': 'PATIENT'
      }
    ]
)
```

```
while True:
    status = transcribe.get_medical_scribe_job(MedicalScribeJobName = job_name)
    if status['MedicalScribeJob']['MedicalScribeJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

> **Note**
>
> AWS 管理控制台目前不支持 AWS HealthScribe 作业。

示例输出

除了转录以外，StartMedicalScribeJob 请求还生成一个单独的临床文档文件。这两个文件均采用 JSON 格式，存储在您在请求中指定的输出位置。下面是每种输出类型的示例：

转录输出示例

AWS HealthScribe 转录文件（来自 StartMedicalScribeJob 请求）具有以下格式：

```
{
  "Conversation": {
    "ConversationId": "sampleConversationUUID",
    "JobName": "sampleJobName",
    "JobType": "ASYNC",
    "LanguageCode": "en-US",
    "ClinicalInsights": [
      {
        "Attributes": [],
        "Category": "MEDICAL_CONDITION",
        "InsightId": "insightUUID1",
        "InsightType": "ClinicalEntity",
        "Spans": [
          {
            "BeginCharacterOffset": 12,
            "Content": "pain",
            "EndCharacterOffset": 15,
            "SegmentId": "uuid1"
```

```
          }
        ],
        "Type": "DX_NAME"
      },
      {
        "Attributes": [],
        "Category": "TEST_TREATMENT_PROCEDURE",
        "InsightId": "insightUUID2",
        "InsightType": "ClinicalEntity",
        "Spans": [
          {
            "BeginCharacterOffset": 4,
            "Content": "mammogram",
            "EndCharacterOffset": 12,
            "SegmentId": "uuid2"
          }
        ],
        "Type": "TEST_NAME"
      },
      {
        "Attributes": [],
        "Category": "TEST_TREATMENT_PROCEDURE",
        "InsightId": "insightUUID3",
        "InsightType": "ClinicalEntity",
        "Spans": [
          {
            "BeginCharacterOffset": 15,
            "Content": "pap smear",
            "EndCharacterOffset": 23,
            "SegmentId": "uuid3"
          }
        ],
        "Type": "TEST_NAME"
      },
      {
        "Attributes": [],
        "Category": "MEDICATION",
        "InsightId": "insightUUID4",
        "InsightType": "ClinicalEntity",
        "Spans": [
          {
            "BeginCharacterOffset": 28,
            "Content": "phentermine",
            "EndCharacterOffset": 38,
```

```
            "SegmentId": "uuid4"
          }
        ],
        "Type": "GENERIC_NAME"
      },
      {
        "Attributes": [
          {
            "AttributeId": "attributeUUID1",
            "Spans": [
              {
                "BeginCharacterOffset": 38,
                "Content": "high",
                "EndCharacterOffset": 41,
                "SegmentId": "uuid5"
              }
            ],
            "Type": "TEST_VALUE"
          }
        ],
        "Category": "TEST_TREATMENT_PROCEDURE",
        "InsightId": "insightUUID5",
        "InsightType": "ClinicalEntity",
        "Spans": [
          {
            "BeginCharacterOffset": 14,
            "Content": "weight",
            "EndCharacterOffset": 19,
            "SegmentId": "uuid6"
          }
        ],
        "Type": "TEST_NAME"
      },
      {
        "Attributes": [],
        "Category": "ANATOMY",
        "InsightId": "insightUUID6",
        "InsightType": "ClinicalEntity",
        "Spans": [
          {
            "BeginCharacterOffset": 60,
            "Content": "heart",
            "EndCharacterOffset": 64,
            "SegmentId": "uuid7"
```

```
          }
        ],
        "Type": "SYSTEM_ORGAN_SITE"
      }
    ],
    "TranscriptItems": [
      {
        "Alternatives": [
          {
            "Confidence": 0.7925,
            "Content": "Okay"
          }
        ],
        "BeginAudioTime": 0.16,
        "EndAudioTime": 0.6,
        "Type": "PRONUNCIATION"
      },
      {
        "Alternatives": [
          {
            "Confidence": 0,
            "Content": "."
          }
        ],
        "BeginAudioTime": 0,
        "EndAudioTime": 0,
        "Type": "PUNCTUATION"
      },
      {
        "Alternatives": [
          {
            "Confidence": 1,
            "Content": "Good"
          }
        ],
        "BeginAudioTime": 0.61,
        "EndAudioTime": 0.92,
        "Type": "PRONUNCIATION"
      },
      {
        "Alternatives": [
          {
            "Confidence": 1,
            "Content": "afternoon"
```

```
        }
      ],
      "BeginAudioTime": 0.92,
      "EndAudioTime": 1.54,
      "Type": "PRONUNCIATION"
    },
    {
      "Alternatives": [
        {
          "Confidence": 0,
          "Content": "."
        }
      ],
      "BeginAudioTime": 0,
      "EndAudioTime": 0,
      "Type": "PUNCTUATION"
    },
    {
      "Alternatives": [
        {
          "Confidence": 0.9924,
          "Content": "You"
        }
      ],
      "BeginAudioTime": 1.55,
      "EndAudioTime": 1.88,
      "Type": "PRONUNCIATION"
    },
    {
      "Alternatives": [
        {
          "Confidence": 1,
          "Content": "lost"
        }
      ],
      "BeginAudioTime": 1.88,
      "EndAudioTime": 2.19,
      "Type": "PRONUNCIATION"
    },
    {
      "Alternatives": [
        {
          "Confidence": 1,
          "Content": "one"
```

```
          }
        ],
        "BeginAudioTime": 2.19,
        "EndAudioTime": 2.4,
        "Type": "PRONUNCIATION"
      },
      {
        "Alternatives": [
          {
            "Confidence": 1,
            "Content": "lb"
          }
        ],
        "BeginAudioTime": 2.4,
        "EndAudioTime": 2.97,
        "Type": "PRONUNCIATION"
      }
    ],
    "TranscriptSegments": [
      {
        "BeginAudioTime": 0.16,
        "Content": "Okay.",
        "EndAudioTime": 0.6,
        "ParticipantDetails": {
          "ParticipantRole": "CLINICIAN_0"
        },
        "SectionDetails": {
          "SectionName": "SUBJECTIVE"
        },
        "SegmentId": "uuid1"
      },
      {
        "BeginAudioTime": 0.61,
        "Content": "Good afternoon.",
        "EndAudioTime": 1.54,
        "ParticipantDetails": {
          "ParticipantRole": "CLINICIAN_0"
        },
        "SectionDetails": {
          "SectionName": "OTHER"
        },
        "SegmentId": "uuid2"
      },
      {
```

```
        "BeginAudioTime": 1.55,
        "Content": "You lost one lb.",
        "EndAudioTime": 2.97,
        "ParticipantDetails": {
          "ParticipantRole": "CLINICIAN_0"
        },
        "SectionDetails": {
          "SectionName": "SUBJECTIVE"
        },
        "SegmentId": "uuid3"
      },
      {
        "BeginAudioTime": 2.98,
        "Content": "Yeah, I think it, uh, do you feel more energy?",
        "EndAudioTime": 6.95,
        "ParticipantDetails": {
          "ParticipantRole": "CLINICIAN_0"
        },
        "SectionDetails": {
          "SectionName": "SUBJECTIVE"
        },
        "SegmentId": "uuid5"
      },
      {
        "BeginAudioTime": 6.96,
        "Content": "Yes.",
        "EndAudioTime": 7.88,
        "ParticipantDetails": {
          "ParticipantRole": "CLINICIAN_0"
        },
        "SectionDetails": {
          "SectionName": "SUBJECTIVE"
        },
        "SegmentId": "uuid6"
      },
      {
        "BeginAudioTime": 7.89,
        "Content": "Uh, how about craving for the carbohydrate or sugar or fat or
anything?",
        "EndAudioTime": 17.93,
        "ParticipantDetails": {
          "ParticipantRole": "CLINICIAN_0"
        },
        "SectionDetails": {
```

```
          "SectionName": "SUBJECTIVE"
        },
        "SegmentId": "uuid7"
      }
    ]
  }
}
```

以下是另一个使用 start-medical-scribe-job 命令的示例，以及带有其它设置的请求正文。

```
aws transcribe start-medical-scribe-job \
--region us-west-2 \
--cli-input-json file://filepath/my-first-medical-scribe-job.json
```

my-first-medical-scribe-job.json 文件包含以下请求正文。

```
{
  "MedicalScribeJobName": "my-first-medical-scribe-job",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
   },
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole",
  "Settings": {
    "ShowSpeakerLabels": false,
    "ChannelIdentification": true
  },
  "ChannelDefinitions": [
    {
      "ChannelId": 0,
      "ParticipantRole":"CLINICIAN"
    }, {
      "ChannelId": 1,
      "ParticipantRole":"PATIENT"
    }
  ]
}
```

示例临床文档输出

文档见解文件（来自 StartMedicalScribeJob 请求）的格式如下：


```
{
  "ClinicalDocumentation": {
    "Sections": [
      {
        "SectionName": "CHIEF_COMPLAINT",
        "Summary": [
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid1"
              },
              {
                "SegmentId": "uuid2"
              },
              {
                "SegmentId": "uuid3"
              },
              {
                "SegmentId": "uuid4"
              },
              {
                "SegmentId": "uuid5"
              },
              {
                "SegmentId": "uuid6"
              }
            ],
            "SummarizedSegment": "Weight loss."
          }
        ]
      },
      {
        "SectionName": "HISTORY_OF_PRESENT_ILLNESS",
        "Summary": [
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid7"
              },
```

```
              {
                "SegmentId": "uuid8"
              },
              {
                "SegmentId": "uuid9"
              },
              {
                "SegmentId": "uuid10"
              }
            ],
            "SummarizedSegment": "The patient is seen today for a follow-up of weight
loss."
          },
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid11"
              },
              {
                "SegmentId": "uuid12"
              },
              {
                "SegmentId": "uuid13"
              }
            ],
            "SummarizedSegment": "They report feeling more energy and craving
carbohydrates, sugar, and fat."
          },
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid14"
              },
              {
                "SegmentId": "uuid15"
              },
              {
                "SegmentId": "uuid16"
              }
            ],
            "SummarizedSegment": "The patient is up to date on their mammogram and pap
smear."
          },
          {
```

```
            "EvidenceLinks": [
              {
                "SegmentId": "uuid17"
              },
              {
                "SegmentId": "uuid18"
              },
              {
                "SegmentId": "uuid19"
              },
              {
                "SegmentId": "uuid20"
              }
            ],
            "SummarizedSegment": "The patient is taking phentermine and would like to
continue."
          }
        ]
      },
      {
        "SectionName": "REVIEW_OF_SYSTEMS",
        "Summary": [
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid21"
              },
              {
                "SegmentId": "uuid22"
              }
            ],
            "SummarizedSegment": "Patient reports intermittent headaches, occasional
chest pains but denies any recent fevers or chills."
          },
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid23"
              },
              {
                "SegmentId": "uuid24"
              }
            ],
```

```
            "SummarizedSegment": "No recent changes in vision, hearing, or any
respiratory complaints."
          }
        ]
      },
      {
        "SectionName": "PAST_MEDICAL_HISTORY",
        "Summary": [
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid25"
              },
              {
                "SegmentId": "uuid26"
              }
            ],
            "SummarizedSegment": "Patient has a history of hypertension and was
diagnosed with Type II diabetes 5 years ago."
          },
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid27"
              },
              {
                "SegmentId": "uuid28"
              }
            ],
            "SummarizedSegment": "Underwent an appendectomy in the early '90s and had a
fracture in the left arm during childhood."
          }
        ]
      },
      {
        "SectionName": "ASSESSMENT",
        "Summary": [
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid29"
              },
              {
                "SegmentId": "uuid30"
```

```
              }
            ],
            "SummarizedSegment": "Weight loss"
          }
        ]
      },
      {
        "SectionName": "PLAN",
        "Summary": [
          {
            "EvidenceLinks": [
              {
                "SegmentId": "uuid31"
              },
              {
                "SegmentId": "uuid32"
              },
              {
                "SegmentId": "uuid33"
              },
              {
                "SegmentId": "uuid34"
              }
            ],
            "SummarizedSegment": "For the condition of Weight loss: The patient was
 given a 30-day supply of phentermine and was advised to follow up in 30 days."
          }
        ]
      }
    ]
  }
}
```


AWS HealthScribe 的静态数据加密

AWS HealthScribe 默认提供加密，以使用 Amazon S3 托管式密钥保护静态的敏感客户数据。

- Amazon S3 托管式密钥 (SSE-S3) - AWS HealthScribe 默认使用 Amazon S3 托管式密钥自动加密中间文件。您无法查看、管理或使用 Amazon S3 托管式密钥，也无法审核其使用情况。不过，您不必采取任何措施或更改任何计划以保护对数据进行加密的密钥。有关更多信息，请参阅 SSE-S3。

默认情况下，静态数据加密有助于降低保护敏感数据的操作开销和复杂性。同时，它还支持构建符合严格加密合规性和监管要求的安全应用程序。

虽然您无法禁用该加密层或选择替代加密类型，但您可以在使用 AWS HealthScribe 创建作业时选择客户管理型密钥，从而在现有 Amazon S3 托管式密钥基础上添加第二层加密。

- 客户管理型密钥 - AWS HealthScribe 支持使用您创建、拥有和管理的对称客户管理型密钥，以在 AWS 拥有的现有加密基础上添加第二层加密。由于您可以完全控制这层加密，因此可以执行以下任务：
  - 制定和维护密钥策略
  - 制订和维护 IAM 策略和授权
  - 启用和禁用密钥策略
  - 轮换密钥加密材料
  - 添加标签
  - 创建密钥别名
  - 计划删除密钥

有关更多信息，请参阅 AWS Key Management Service Developer Guide 中的 customer managed key。

> **Note**
>
> AWS HealthScribe 使用 AWS 拥有的密钥自动启用静态加密，以免费保护个人身份数据。不过，使用客户管理型密钥时，会收取 AWS KMS 费用。有关定价的更多信息，请参阅 AWS Key Management Service 定价。
> 有关 AWS KMS 的更多信息，请参阅什么是 AWS Key Management Service。

创建客户管理型密钥

您可以使用 AWS Management Console 或 AWS KMS API 创建对称的客户管理型密钥。要创建对称客户管理型密钥，请按照 AWS Key Management Service Developer Guide 中的 Creating symmetric customer managed key 的步骤进行操作。

密钥策略控制对客户管理型密钥的访问。每个客户管理型密钥必须只有一个密钥策略，其中包含确定谁可以使用密钥以及如何使用密钥的声明。创建客户管理型密钥时，可以指定密钥策略。有关更多

信息，请参阅 AWS Key Management Service Developer Guide 中的 Managing access to customer managed keys。

如果您使用密钥的账户与 StartMedicalScribeJob 请求中指定为 DataAccessRoleArn 的 IAM 角色相同，则无需更新密钥策略。要在与您的 DataAccessRole 不同的账户中使用您的客户管理型密钥，您必须在密钥策略中信任该 DataAccessRoleArn 才能执行以下操作：

- kms:Encrypt - 允许使用客户管理型密钥进行加密
- kms:Decrypt - 允许使用客户管理型密钥进行解密
- kms:DescribeKey - 提供客户管理型密钥详细信息以允许 AWS HealthScribe 验证密钥

以下是一个示例策略语句，您可以添加该语句为 IAM 角色授予跨账户权限以使用客户管理型密钥：

```
"Statement" : [
  {
    "Sid": "Allow access to the DataAcessRole for StartMedicalScribeJob",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:role/DataAccessRole"
    },
    "Action": [
      "kms:DescribeKey",
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:GenerateDataKey"
    ],
    "Resource" : "*"
  }
]
```

无论客户管理型密钥和 DataAccessRole 位于同一账户还是不同账户中，DataAccessRole 都需要具有权限才能使用客户管理型密钥执行上述操作。以下是您可以添加到 DataAccessRole 的示例策略语句：

```
"Statement" : [
  {
    "Sid": "Allow role to perform AWS KMS actions for customer managed key",
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:Encrypt",
      "kms:Decrypt"
```

```
    ],
    "Resource": "*"
  }
]
```

有关在策略中指定权限的更多信息，请参阅 AWS Key Management Service Developer Guide。有关密钥访问故障排除的更多信息，请参阅 AWS Key Management Service Developer Guide。

为 AWS HealthScribe 指定客户管理型密钥

您可以指定客户管理型密钥以作为 StartMedicalScribeJob 请求的第二层加密。在创建 StartMedicalScribeJob 请求时，您可以在请求中包含 OutputEncryptionKMSKeyId 字段以指定客户管理型密钥。

AWS KMS 加密上下文

AWS KMS 加密上下文是纯文本、非机密键值对的映射。该映射表示经过身份验证的额外数据（称为加密上下文对），它为数据提供额外的安全层。AWSHealthScribe 需要使用对称加密密钥加密 AWS HealthScribe 输出，并将其存储到客户指定的 Amazon S3 存储桶中。要了解更多信息，请参阅 AWS KMS 中的非对称密钥。

创建加密上下文对时，请勿包含敏感信息。加密上下文不是秘密的 - 它在 CloudTrail 日志中以明文形式显示（因此，您可以使用它识别加密操作并进行分类）。加密上下文对可以包含特殊字符，如下划线 (_)、短划线 (-)、斜杠（/、\）和冒号 (:)。

> **Tip**
>
> 将加密上下文对中的值与正在加密的数据关联会很有用。尽管此操作并非必需，但我们建议您使用与加密内容相关的非敏感元数据，例如文件名、标头值或未加密的数据库字段。
> 要在 API 中使用输出加密，请在 StartMedicalScribeJob 操作中设置 KMSEncryptionContext 参数。要为输出加密操作提供加密上下文，OutputEncryptionKMSKeyId 参数必须引用对称 AWS KMS 密钥 ID。
> 您可以将 AWS KMS 条件键与 IAM 策略一起使用，以根据加密操作请求中使用的加密上下文控制对对称加密 AWS KMS 密钥的访问。有关示例加密上下文策略，请参阅 AWS KMS 加密上下文策略。
> 使用加密上下文是可选操作，但建议使用。有关更多信息，请参阅加密上下文。

Amazon Transcribe 的文档历史记录

- 最新文档更新日期：2023 年 11 月 13 日

下表介绍乐每一版 Amazon Transcribe 中的重大更改。要获得本文档的更新通知，您可以订阅 RSS 源。

变更	说明	日期
功能更新	Amazon Transcribe 现在支持生成式通话摘要。	2023 年 11 月 29 日
章节更新	更新新的 PII 编辑和语言识别输出格式。	2023 年 11 月 13 日
功能更新	现在可以将分类与声道识别相结合。	2023 年 3 月 6 日
功能更新	现在可以将声道识别与分类相结合。	2023 年 3 月 6 日
章节更新	IAM 最佳实践已更新。	2023 年 2 月 13 日
新语言	Amazon Transcribe 现在支持越南语和瑞典语。	2022 年 12 月 6 日
新特征	Amazon Transcribe 现在支持实时通话分析。	2022 年 11 月 28 日
功能更新	现在支持印地语和泰语的流式转录编辑和识别。	2022 年 11 月 11 日
章节更新	新的 PII 类别可用于流式转录编辑和识别。	2022 年 9 月 14 日
章节更新	自定义语言模型部分已修订。	2022 年 6 月 18 日
章节更新	批量语言识别现在可以识别每个音频文件的多种语言。	2022 年 5 月 31 日

指南更新	Amazon Transcribe API 参考现在是一份独立指南。	2022 年 4 月 1 日
新增章节	增加了 Amazon Transcribe、Amazon Transcribe Medical 和 Amazon Transcribe 通话分析的新对比表。	2022 年 3 月 21 日
新增章节	增加了新的 SDK 代码示例章节。	2022 年 3 月 21 日
功能更新	通话分析现在提供通话摘要。	2022 年 3 月 21 日
章节更新	现在，介绍性章节展示了 Amazon Transcribe 使用案例。	2022 年 3 月 21 日
章节更新	入门章节已更新为特定于方法。	2022 年 3 月 21 日
章节更新	流式转录章节已更新和重组。	2022 年 3 月 21 日
功能更新	语言识别现在针对流式转录支持自定义词汇表和自定义词汇表过滤器。	2022 年 3 月 11 日
新事件	新增了一种事件类型：词汇表事件。	2022 年 2 月 7 日
章节更新	自定义词汇表部分已更新。	2022 年 1 月 20 日
新特征	语言识别现在可用于流式转录。	2021 年 11 月 23 日
新特征	语言识别现在可以与自定义语言模型、自定义词汇表、词汇表过滤和内容编辑一起使用。	2021 年 10 月 29 日

新特征	Amazon Transcribe 现在针对流式转录支持自定义语言模型。	2021 年 10 月 20 日
新特征	Amazon Transcribe 现在可以为您的视频文件生成字幕。	2021 年 9 月 16 日
新特征	Amazon Transcribe 现在针对流式转录支持 PII 编辑和识别。	2021 年 9 月 14 日
新特征	Amazon Transcribe 现在支持 AWS KMS 加密上下文，以提高 AWS 账户 资源的安全性。	2021 年 9 月 10 日
新语言	Amazon Transcribe 现在支持南非荷兰语、丹麦语、汉语（繁体）、泰语、新西兰英语和南非英语。	2021 年 8 月 26 日
新特征	Amazon Transcribe 现在支持资源标记。	2021 年 8 月 24 日
新特征	Amazon Transcribe 现在针对批量转录作业支持通话分析。	2021 年 8 月 4 日
新特征	Amazon Transcribe 现在支持结合批量自定义语言模型使用自定义词汇表。	2021 年 5 月 12 日
新特征	Amazon Transcribe 现在针对流式转录支持部分结果稳定功能。	2021 年 5 月 11 日
新特征	Amazon Transcribe 现在支持澳大利亚英语、英式英语、印地语和美国西班牙语作为自定义语言模型。	2021 年 3 月 19 日

新特征	Amazon Transcribe 现在针对流音频转录支持 OGG/OPUS 和 FLAC 编解码器。	2020 年 11 月 24 日
新语言	Amazon Transcribe 增加了对意大利语和德语流音频转录的支持。	2020 年 11 月 4 日
AWS 区域扩展	Amazon Transcribe 现已在法兰克福 (eu-central-1) 和伦敦 (eu-west-2) 提供。	2020 年 11 月 4 日
新特征	Amazon Transcribe 增加了对批处理转录中的接口 VPC 端点的支持。	2020 年 10 月 9 日
新特征	Amazon Transcribe 增加了对流式转录中声道识别的支持。	2020 年 9 月 17 日
新特征	Amazon Transcribe 增加了对批量转录中自动语言识别的支持。	2020 年 9 月 15 日
新特征	Amazon Transcribe 增加了对流式转录中发言者划分的支持。	2020 年 8 月 19 日
新特征	Amazon Transcribe 增加了对自定义语言模型的支持。	2020 年 8 月 5 日
新特征	Amazon Transcribe 增加了对流式转录中接口 VPC 端点的支持。	2020 年 6 月 26 日
新特征	Amazon Transcribe 增加了对流式转录中词汇表过滤的支持。	2020 年 5 月 20 日

新特征	Amazon Transcribe 增加了对自动修订个人身份信息的支持。	2020 年 2 月 26 日
新特征	Amazon Transcribe 增加了一项新功能，可以创建要在转录过程中过滤的单词的自定义词汇表。	2019 年 12 月 20 日
新特征	Amazon Transcribe 增加了一项新功能，可以对转录作业进行排队。	2019 年 12 月 19 日
新语言	Amazon Transcribe 增加了对海湾阿拉伯语、希伯来语、日语、马来语、瑞士德语、泰卢固语和土耳其语的支持。	2019 年 11 月 21 日
AWS 区域扩展	Amazon Transcribe 现已在亚太地区（东京）(ap-northeast-1) 提供。	2019 年 11 月 21 日
新特征	Amazon Transcribe 添加了对备选转录的支持。	2019 年 11 月 20 日
新语言	Amazon Transcribe 增加了对荷兰语、波斯语、印度尼西亚语、爱尔兰语、葡萄牙语、苏格兰英语、泰米尔语和威尔士英语的支持。	2019 年 11 月 12 日
新语言	Amazon Transcribe 现支持用澳大利亚英语 (en-au) 进行流式转录。	2019 年 10 月 25 日

AWS 区域扩展	Amazon Transcribe 现已在中国（北京）(cn-north-1) 和中国（宁夏）(cn-northwest-1) 提供。	2019 年 10 月 9 日
新特征	Amazon Transcribe 使您能够提供自己的 KMS key来加密转录输出文件。有关更多信息，请参阅 StartStreamTranscription API 的 OutputEncryptionKMSKeyId 参数。	2019 年 9 月 24 日
新语言	Amazon Transcribe 增加了对中文（普通话）、中国大陆简体中文和俄语的支持。	2019 年 8 月 23 日
新特征	Amazon Transcribe 增加了对使用 WebSocket 协议进行流音频转录的支持。	2019 年 7 月 19 日
新特征	AWS CloudTrail 现在为 StartStreamTranscription API 记录事件。	2019 年 7 月 19 日
AWS 区域扩展	Amazon Transcribe 现已在美国西部（北加利福尼亚）(us-west-1) 区域提供。	2019 年 6 月 27 日
新语言	Amazon Transcribe 增加了对现代标准阿拉伯语的支持。	2019 年 5 月 28 日
新特征	Amazon Transcribe 现在将数字单词转录成美国英语的数字。例如，“forty-two”将转录成“42”。	2019 年 5 月 23 日
新语言	Amazon Transcribe 增加了对印地语和印度英语的支持。	2019 年 5 月 15 日

新 SDK	适用于 C++ 的 AWS SDK 现在支持 Amazon Transcribe。	2019 年 5 月 8 日
新语言	Amazon Transcribe 添加了对西班牙语的支持。	2019 年 4 月 19 日
AWS 区域扩展	Amazon Transcribe 现在欧洲（法兰克福）(eu-central-1) 和亚太地区（首尔）(ap-northeast-2) 区域提供。	2019 年 4 月 18 日
新语言	Amazon Transcribe 添加了对用英国英语、法语和加拿大法语进行流式转录的支持。	2019 年 4 月 5 日
新特征	适用于 Ruby V3 的 AWS SDK 现在支持 Amazon Transcribe	2019 年 3 月 25 日
新特征	Amazon Transcribe 支持自定义词汇表，这是一个列表，其中包含您希望 Amazon Transcribe 在您的音频输入中识别的特殊词。	2019 年 3 月 25 日
新语言	Amazon Transcribe 添加了对德语和韩语的支持。	2019 年 3 月 22 日
新语言	Amazon Transcribe 现支持用美国西班牙语 (es-US) 进行流式转录。	2019 年 2 月 7 日
AWS 区域扩展	Amazon Transcribe 现在南美洲（圣保罗）(sa-east-1) 区域提供。	2019 年 2 月 7 日

AWS 区域扩展	Amazon Transcribe 现在亚太地区（孟买）(ap-south-1)、亚太地区（新加坡）(ap-southeast-1)、欧洲（伦敦）(eu-west-2) 和欧洲（巴黎）(eu-west3) 提供。	2019 年 1 月 24 日
新语言	Amazon Transcribe 添加了对法语、意大利语和巴西葡萄牙语的支持。	2018 年 12 月 20 日
新特征	Amazon Transcribe 现在支持音频流转录。	2018 年 11 月 19 日
新语言	Amazon Transcribe 添加了对澳大利亚英语、英国英语和加拿大法语的支持。	2018 年 11 月 15 日
AWS 区域扩展	Amazon Transcribe 现已在加拿大（中部）(ca-central-1) 和亚太地区（悉尼）(ap-southeast-2) 推出。	2018 年 7 月 17 日
新特征	现在，您可以指定自己的位置来存储来自转录作业的输出。	2018 年 7 月 11 日
新特征	增加了 AWS CloudTrail 和 Amazon CloudWatch Events 集成。	2018 年 6 月 28 日
新特征	Amazon Transcribe 增加了对自定义词汇表的支持。	2018 年 4 月 4 日
新指南	这是 Amazon Transcribe 开发人员指南的首次发布。	2017 年 11 月 29 日

AWS 术语表

有关最新的 AWS 术语，请参阅《AWS 词汇表参考》中的 AWS 词汇表。

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。