aws

開発者ガイド

Amazon Transcribe

Amazon Transcribe: 開発者ガイド

Table of Contents

Amazon Transcribe とは?

Amazon Transcribeは、機械学習モデルを使用して音声をテキストに変換する自動音声認識サービスです。Amazon Transcribeスタンドアロンの文字起こしサービスとして使用することも、speech-to-text任意のアプリケーションに機能を追加することもできます。

を使用するとAmazon Transcribe、言語をカスタマイズして特定のユースケースの精度を向上させたり、コンテンツをフィルタリングして顧客のプライバシーや視聴者に適切な言語を表示したり、マルチチャンネルオーディオでコンテンツを分析したり、個々のスピーカーの音声を分割したりすることができます。

メディアをリアルタイム（ストリーミング）で転記することも、Amazon S3バケットにあるメディアファイルを転記する（バッチ）こともできます。文字起こしの種類ごとにどの言語がサポートされているかについては、サポートされている言語および言語固有の機能表を参照してください。

トピック

- Amazon Transcribe と HIPAA の適格性
- 料金
- 提供リージョンとクォータ

「とはAmazon Transcribe?」を参照してください。 このサービスの短いビデオツアーをご覧ください。

詳細については、「Amazon Transcribe の仕組み」および「Amazon Transcribe の開始方法」を参照してください。

> **Tip**
>
> API に関する情報は Amazon TranscribeAPI リファレンスにあります。

Amazon Transcribe と HIPAA の適格性

Amazon TranscribeAWSのHIPAA資格およびBAAの対象となるため、BAAのお客様は保管中および転送中のすべてのPHIを使用時に暗号化する必要があります。PHIの自動識別は、追加料金なしで、Amazon Transcribe事業を行っているすべての地域で利用できます。詳細については、HIPAAの資格とBAAを参照してください。

料金

Amazon Transcribeは、pay-as-you-goサービスです。料金は、録音された音声の秒数に基づいており、月単位で請求されます。

料金は 1 秒ごとに課金され、1 リクエストごとに課金される料金は 1 秒ごとに課金され、15 秒未満の場合は 15 秒の場合は 15 秒の場合は 15 秒の場合は 15 秒料金が発生します。PII コンテンツの編集やカスタム言語モデルなどの機能には追加料金がかかりますのでご注意ください。

それぞれのコスト情報についてはAWS リージョン、「Amazon Transcribe価格設定」を参照してください。

提供リージョンとクォータ

Amazon TranscribeAWS リージョン以下でサポートされています。

[Region] (リージョン)	トランスクリプションタイプ
af-south-1 (ケープタウン)	バッチ、ストリーミング
ap-east-1 (香港)	バッチ
ap-northeast-1 (東京)	バッチ、ストリーミング
ap-northeast-2 (ソウル)	バッチ、ストリーミング
ap-south-1 (ムンバイ)	バッチ、ストリーミング
ap-southeast-1 (シンガポール)	バッチ、ストリーミング
ap-southeast-2 (シドニー)	バッチ、ストリーミング
ca-central-1 (カナダ、中部)	バッチ、ストリーミング
eu-central-1 (フランクフルト)	バッチ、ストリーミング
eu-north-1 (ストックホルム)	バッチ
eu-west-1 (アイルランド)	バッチ、ストリーミング
eu-west-2 (ロンドン)	バッチ、ストリーミング

[Region] (リージョン)	トランスクリプションタイプ
eu-west-3 (パリ)	バッチ
me-south-1 (バーレーン)	バッチ
sa-east-1 (サンパウロ)	バッチ、ストリーミング
us-east-1 (バージニア北部)	バッチ、ストリーミング
us-east-2 (オハイオ)	バッチ、ストリーミング
us-gov-east-1 (GovCloud、米国東部)	バッチ、ストリーミング
us-gov-west-1 (GovCloud、米国西部)	バッチ、ストリーミング
us-west-1 (サンフランシスコ)	バッチ
us-west-2 (オレゴン)	バッチ、ストリーミング

Important

、、Amazon Transcribeおよびコール分析ではAmazon Transcribe Medical、リージョンのサポートが異なります。

サポートされている各リージョンのエンドポイントを取得するには、『AWS総合リファレンス』の「サービスエンドポイント」を参照してください。

トランスクリプションに関連するクォータのリストについては、AWS 全般リファレンスの「サービスクォータ」を参照してください。一部のクォータは、リクエストに応じて変更できます。[調整可能] 列に [はい] と表示されている場合は、引き上げをリクエストできます。これを行うには、提供されたリンクを選択します。

Amazon Transcribe features

どの Amazon Transcribe ソリューションがユースケースに最も適しているかを判断するために、次の表に特徴の比較を示します。

「バッチ」と 'post-call' は、 Amazon S3 バケットにあるファイルの文字起こしを指し、「ストリーミング」と 'real-time' は、メディアをリアルタイムで文字起こしすることを指します。

機能	Amazon Transcribe	Amazon Transcribe Medical[1]	Amazon Transcribe 通話分析
設定オプション			
代替文字起こし	バッチ、ストリーミング	バッチ、ストリーミング	なし
チャネル識別	バッチ、ストリーミング	バッチ、ストリーミング	post-call, real-time
ジョブキューイング	バッチ	なし	post-call
言語識別	バッチ、ストリーミング	なし	post-call
多言語識別	バッチ、ストリーミング	なし	なし
話者ダイアライゼーション	バッチ、ストリーミング	バッチ、ストリーミング	post-call
文字起こし番号[2]	バッチ、ストリーミング	バッチ、ストリーミング	post-call, real-time
会話分析			
コールの特性	なし	なし	post-call
コールサマリー[2]	なし	なし	post-call

機能	Amazon Transcribe	Amazon Transcribe Medical[1]	Amazon Transcribe 通話分析
カスタムの分類	なし	なし	post-call
リアルタイムのカテゴリイベント	なし	なし	real-time
リアルタイムの問題検出[2]	なし	なし	real-time
リアルタイムのスピーカーの感情	なし	なし	real-time
スピーカーの感情	なし	なし	post-call
言語のカスタマイズ			
カスタム言語モデル[2]	バッチ、ストリーミング	なし	post-call, real-time
カスタム語彙	バッチ、ストリーミング	バッチ、ストリーミング	post-call, real-time
リソース組織			
タグ付け	バッチ	バッチ	post-call
機密データ			
個人の健康情報の識別[2]	なし	バッチ、ストリーミング	なし
個人を特定できる情報の識別[2]	ストリーミング	なし	real-time
音声の編集[2]	なし	なし	post-call, real-time
編集済みトランスクリプト[2]	バッチ、ストリーミング	なし	post-call, real-time

機能	Amazon Transcribe	Amazon Transcribe Medical[1]	Amazon Transcribe 通話分析
語彙フィルタリング	バッチ、ストリーミング	なし	post-call, real-time
動画			
字幕	バッチ	なし	なし

[1] Amazon Transcribe Medical は米国英語でのみ利用できます。
[2] この機能は一部の言語ではご利用いただけません。詳細については サポートされている言語および言語固有の機能 の表をご覧ください。

サポートされている言語および言語固有の機能

でサポートされている言語を次の表 Amazon Transcribe に示します。また、言語固有の機能も示しています。文字起こしを進める前に、使用する機能がメディアの言語に対応していることを確認してください。

Amazon Transcribe 機能の完全なリストを表示するには、「機能の概要」を参照してください。

次の表では、「バッチ」とは Amazon S3 バケットにあるメディアファイルの文字起こしを指し、「ストリーミング」とはストリーミングされたメディアをリアルタイムで文字起こしすることを指します。コール分析文字起こしの場合、 'post-call'は Amazon S3 バケットにあるメディアファイルを文字起こしすることを示し、 はストリーミングされたメディアをリアルタイムで文字起こしすること'real-time'を指します。

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
アブハズ	ab-GE	バッチ	なし	バッチ	なし	いいえ	なし
アフリカーンス語	af-ZA	バッチ	なし	バッチ	なし	いいえ	なし
アラビア語、(ガルフ)	ar-AE	バッチ	なし	いいえ	いいえ	なし	post-call
アラビア語、(モダンスタンダード)	ar-SA	バッチ	なし	いいえ	いいえ	いいえ	なし
アルメニア語	hy-AM	バッチ	なし	バッチ	なし	いいえ	なし
アストゥリアス語	ast-ES	バッチ	なし	バッチ	なし	いいえ	なし

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
アゼルバイジャン語	az-AZ	バッチ	なし	バッチ	なし	いいえ	なし
バシキール語	ba-RU	バッチ	なし	バッチ	なし	いいえ	なし
バスク語	eu-ES	バッチ	なし	バッチ	なし	いいえ	なし
ベラルーシ語	be-BY	バッチ	なし	バッチ	なし	いいえ	なし
ベンガル語	bn-IN	バッチ	なし	バッチ	なし	いいえ	なし
ボスニア語	bs-BA	バッチ	なし	バッチ	なし	いいえ	なし
ブルガリア語	bg-BG	バッチ	なし	バッチ	なし	いいえ	なし
カタロニア語	ca-ES	バッチ	なし	バッチ	なし	いいえ	なし
中部クルド語 (イラン)	ckb-IR	バッチ	なし	バッチ	なし	いいえ	なし
中部クルド語 (イラク)	ckb-IQ	バッチ	なし	バッチ	なし	いいえ	なし
中国語 (簡体字)	zh-CN	バッチ、ストリーミング	なし	いいえ	いいえ	なし	post-call

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
中国語(繁体字)	zh-TW	バッチ	なし	いいえ	いいえ	いいえ	なし
クロアチア語	hr-HR	バッチ	なし	バッチ	なし	いいえ	なし
チェコ語	cs-CZ	バッチ	なし	バッチ	なし	いいえ	なし
デンマーク語	da-DK	バッチ	なし	バッチ	なし	いいえ	なし
オランダ語	nl-NL	バッチ	なし	バッチ	なし	いいえ	なし
英語 英語(オーストラリア)	en-AU	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	ストリーミング	post-call, real-time
英語 英語(イギリス)	en-GB	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	ストリーミング	post-call, real-time
英語 英語(インド)	en-IN	バッチ	バッチ	バッチ	なし	なし	post-call
英語 英語(アイルランド)	en-IE	バッチ	バッチ	バッチ	なし	なし	post-call
英語 英語(ニュージーランド)	en-NZ	バッチ	バッチ	バッチ	なし	いいえ	なし

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
英語 英語 (スコットランド)	en-AB	バッチ	バッチ	バッチ	なし	なし	post-call
英語 英語 (南アフリカ)	en-ZA	バッチ	バッチ	バッチ	なし	いいえ	なし
英語、アメリカ	en-US	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	post-call, real-time
英語 英語 (ウェールズ)	en-WL	バッチ	バッチ	バッチ	なし	なし	post-call
エストニア語	et-ET	バッチ	なし	バッチ	なし	いいえ	なし
ペルシア語	fa-IR	バッチ	なし	いいえ	いいえ	いいえ	なし
フィンランド語	fi-FI	バッチ	なし	バッチ	なし	いいえ	なし
フランス語	fr-FR	バッチ、ストリーミング	なし	バッチ、ストリーミング	なし	なし	post-call, real-time
フランス語 フランス語 (カナダ)	fr-CA	バッチ、ストリーミング	なし	バッチ、ストリーミング	なし	なし	post-call, real-time

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
ガリシア語	gl-ES	バッチ	なし	バッチ	なし	いいえ	なし
グルジア語	ka-GE	バッチ	なし	バッチ	なし	いいえ	なし
ドイツ語	de-DE	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	なし	post-call, real-time
ドイツ語 ドイツ語 (スイス)	de-CH	バッチ	バッチ	バッチ	なし	なし	post-call
ギリシャ語	el-GR	バッチ	なし	バッチ	なし	いいえ	なし
グジャラート語	gu-IN	バッチ	なし	バッチ	なし	いいえ	なし
ハウサ語	ha-NG	バッチ	なし	バッチ	なし	いいえ	なし
ヘブライ語	he-IL	バッチ	なし	いいえ	いいえ	いいえ	なし
ヒンディー語 英語 (インド)	hi-IN	バッチ、ストリーミング	なし	バッチ、ストリーミング	バッチ	なし	post-call
ハンガリー語	hu-HU	バッチ	なし	バッチ	なし	いいえ	なし
アイスランド語	is-IS	バッチ	なし	バッチ	なし	いいえ	なし

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
インドネシア語	id-ID	バッチ	なし	バッチ	なし	いいえ	なし
イタリア語	it-IT	バッチ、ストリーミング	なし	バッチ、ストリーミング	なし	なし	post-call, real-time
日本語	ja-JP	バッチ、ストリーミング	なし	なし	バッチ、ストリーミング	なし	post-call
カビル語	kab-DZ	バッチ	なし	バッチ	なし	いいえ	なし
カンナダ語	kn-IN	バッチ	なし	バッチ	なし	いいえ	なし
カザフ語	kk-KZ	バッチ	なし	バッチ	なし	いいえ	なし
キニヤルワンダ語	rw-RW	バッチ	なし	バッチ	なし	いいえ	なし
韓国語	ko-KR	バッチ、ストリーミング	なし	いいえ	いいえ	なし	post-call
キルギス語	ky-KG	バッチ	なし	バッチ	なし	いいえ	なし
ラトビア語	lv-LV	バッチ	なし	バッチ	なし	いいえ	なし
リトアニア語	lt-LT	バッチ	なし	バッチ	なし	いいえ	なし

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
ルガンダ語	lg-IN	バッチ	なし	バッチ	なし	いいえ	なし
マケドニア語	mk-MK	バッチ	なし	バッチ	なし	いいえ	なし
マレー語	ms-MY	バッチ	なし	バッチ	なし	いいえ	なし
マラヤーラム語	ml-IN	バッチ	なし	バッチ	なし	いいえ	なし
マルタ語	mt-MT	バッチ	なし	バッチ	なし	いいえ	なし
マラーティー語	mr-IN	バッチ	なし	バッチ	なし	いいえ	なし
牧地マリ語	mhr-RU	バッチ	なし	バッチ	なし	いいえ	なし
モンゴル語	mn-MN	バッチ	なし	バッチ	なし	いいえ	なし
ノルウェーブークモール語	no-NO	バッチ	なし	バッチ	なし	いいえ	なし
オディア/オリヤ語	or-IN	バッチ	なし	バッチ	なし	いいえ	なし
パシュト語	ps-AF	バッチ	なし	バッチ	なし	いいえ	なし
ポーランド語	pl-PL	バッチ	なし	バッチ	なし	いいえ	なし

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
ポルトガル語	pt-PT	バッチ	なし	バッチ	なし	なし	post-call
ポルトガル語 ポルトガル語 (ブラジル)	pt-BR	バッチ、ストリーミング	なし	バッチ、ストリーミング	なし	なし	post-call, real-time
パンジャブ語	pa-IN	バッチ	なし	バッチ	なし	いいえ	なし
ルーマニア語	ro-RO	バッチ	なし	バッチ	なし	いいえ	なし
ロシア語	ru-RU	バッチ	なし	いいえ	いいえ	いいえ	なし
セルビア語	sr-RS	バッチ	なし	バッチ	なし	いいえ	なし
シンハラ語	si-LK	バッチ	なし	バッチ	なし	いいえ	なし
スロバキア語	sk-SK	バッチ	なし	バッチ	なし	いいえ	なし
スロベニア語	sl-SI	バッチ	なし	バッチ	なし	いいえ	なし
ソマリ語	so-SO	バッチ	なし	バッチ	なし	いいえ	なし
スペイン語	es-ES	バッチ	なし	バッチ	なし	なし	post-call

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
スペイン語、アメリカ	es-US	バッチ、ストリーミング	なし	バッチ、ストリーミング	バッチ、ストリーミング	バッチ、ストリーミング	post-call, real-time
スンダ語	su-ID	バッチ	なし	バッチ	なし	いいえ	なし
スワヒリ語 (ケニア)	sw-KE	バッチ	なし	バッチ	なし	いいえ	なし
スワヒリ語 (ブルンジ)	sw-BI	バッチ	なし	バッチ	なし	いいえ	なし
スワヒリ語 (ルワンダ)	sw-RW	バッチ	なし	バッチ	なし	いいえ	なし
スワヒリ語 (タンザニア)	sw-TZ	バッチ	なし	バッチ	なし	いいえ	なし
スワヒリ語 (ウガンダ)	sw-UG	バッチ	なし	バッチ	なし	いいえ	なし
スウェーデン語	sv-SE	バッチ	なし	バッチ	なし	いいえ	なし
タガログ語/フィリピン語	tl-PH	バッチ	なし	バッチ	なし	いいえ	なし
タミル語	ta-IN	バッチ	なし	いいえ	いいえ	いいえ	なし

[言語]	言語コード	データ入力	文字起こし番号	頭字語	カスタム言語モデル	リダクション	コール分析*
タタール語	tt-RU	バッチ	なし	バッチ	なし	いいえ	なし
テルグ語	te-IN	バッチ	なし	いいえ	いいえ	いいえ	なし
タイ語	th-TH	バッチ、ストリーミング	なし	バッチ、ストリーミング	なし	いいえ	なし
トルコ語	tr-TR	バッチ	なし	バッチ	なし	いいえ	なし
ウクライナ語	uk-UA	バッチ	なし	バッチ	なし	いいえ	なし
ウイグル語	ug-CN	バッチ	なし	バッチ	なし	いいえ	なし
ウズベク語	uz-UZ	バッチ	なし	バッチ	なし	いいえ	なし
ベトナム語	vi-VN	バッチ	なし	バッチ	なし	いいえ	なし
ウェールズ語	cy-WL	バッチ	なし	バッチ	なし	いいえ	なし
ウォロフ語	wo-SN	バッチ	なし	バッチ	なし	いいえ	なし
ズールー語	zu-ZA	バッチ	なし	バッチ	なし	いいえ	なし

*コール分析に関する以下のインサイトは、一部の英語の方言でのみサポートされています。

- コールサマリー: en-* (すべて英語の方言)

- 問題検出: en-AU、en-GB、en-US

サポートされているプログラミング言語

Amazon Transcribe は、次の AWS SDKsをサポートしています。

バッチ文字起こし	ストリーミング文字起こし
.NET	.NET はストリーミングをサポートしません。
AWS コマンドラインインターフェイス (CLI)	CLI はストリーミングをサポートしません。
C++	C++
Go	Go
Java V2	Java V2
JavaScript	JavaScript V3
PHP v3	PHP v3
Python Boto3	用の Python ストリーミング SDK Amazon Transcribe
Ruby v3	Ruby v3
Rust	Rust

で SDKs「」を参照してくださいAWS SDKs を使用した文字起こし。 Amazon Transcribe

使用可能なすべての AWS SDKs「 で構築するツール AWS」を参照してください。

> **Tip**
>
> SDK コードサンプルは、次の GitHub リポジトリにあります。
>
> - AWS コードの例
> - Amazon Transcribe 例

カスタム語彙および語彙フィルターの文字セット

Amazon Transcribe がサポートする言語ごとに、Amazon Transcribe は認識できる特定の文字セットがあります。カスタム語彙または語彙フィルターを作成する場合は、その言語の文字セットに記載されている文字のみを使用してください。サポートされていない文字を使用すると、カスタム語彙または語彙フィルターが失敗します。

> **Important**
>
> カスタム語彙ファイルが、対応された Unicode コードポイントと、次の文字セット内にリスト化されたコードポイントシーケンスのみを使用していることを必ず確認してください。

多くの Unicode 文字は、異なるコードポイントを使用していても、一般的なフォントでは同じように表示されることがあります。このガイドに記載されているコードポイントのみに対応しています。例えば、フランス語の単語 déjà は 合成済み 文字 (1 つの Unicode 値がアクセント付き文字を表す) または 分解 文字 (2 つの Unicode 値はアクセント付き文字を表し、1 つは基本文字、もう 1 つはアクセントを表す) を使用してレンダリングできます。

- 合成済みバージョン 0064 **00E9** 006A **00E0** (déjà としてレンダリングされます)。
- 分解バージョン 0064 **0065 0301** 006A **0061 0300** (déjà としてレンダリングされます)。

トピック

- ブルガリア語の文字セット
- カタロニア語の文字セット
- 中部クルド語の文字セット
- 中国語、北京語 (中国本土) 、簡体字の文字セット
- 中国語、北京語 (台湾)、繁体字の文字セット
- クロアチア語の文字セット
- チェコ語の文字セット
- デンマーク語の文字セット
- オランダ語の文字セット
- 英語の文字セット
- エストニア語の文字セット
- ペルシャ語の文字セット
- フィンランド語の文字セット
- フランス語の文字セット
- ガリシア語の文字セット
- グルジア語の文字セット
- ドイツ語の文字セット
- ギリシャ語の文字セット
- グジャラート語の文字セット
- ハウサ語の文字セット
- ヘブライ語の文字セット
- ヒンディー語の文字セット
- ハンガリー語の文字セット
- アイスランド語の文字セット
- インドネシア語の文字セット
- イタリア語の文字セット
- 日本語の文字セット
- カビル語の文字セット
- カンナダ語の文字セット
- カザフ語の「文字セット

- キニヤルワンダ語の文字セット
- 韓国語の文字セット
- キルギス語の文字セット
- ラトビア語の文字セット
- リトアニア語の文字セット
- ルガンダ語の文字セット
- マケドニア語の文字セット
- マレー語の文字セット
- マラヤーラム語の文字セット
- マルタ語の文字セット
- マラーティー語の文字セット
- 牧地マリ語の文字セット
- モンゴル語の文字セット
- ノルウェーブークモール語の文字セット
- オディア/オリヤ語の文字セット
- パシュト語の文字セット
- ポーランド語の文字セット
- ポルトガル語の文字セット
- パンジャブ語の文字セット
- ルーマニア語の文字セット
- ロシア語の文字セット
- セルビア語の文字セット
- シンハラ語の文字セット
- スロバキア語の文字セット
- スロベニア語の文字セット
- ソマリ語の文字セット
- スペイン語の文字セット
- スンダ語の文字セット
- スワヒリ語の文字セット
- スウェーデン語の文字セット

- タガログ語/フィリピン語の文字セット
- タミル語の文字セット
- タタール語の文字セット
- テルグ語の文字セット
- タイ語の文字セット
- トルコ語の文字セット
- ウクライナ語の文字セット
- ウイグル語の文字セット
- ウズベク語の文字セット
- ベトナム語の文字セット
- ウェールズ語の文字セット
- ウォロフ語の文字セット
- ズールー語の文字セット

アブハズ語の文字セット

アブハズ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
а	0430	лъ	0459
б	0431	нь	045A
в	0432	ҧ	045B
г	0433	ќ	045C

文字	Code	文字	Code
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	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

文字	Code	文字	Code
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		

アフリカーンス語の文字セット

アフリカーンス語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a~z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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

アラビア語の文字セット

アラビア語のカスタムボキャブラリーでは、次の Unicode 文字を Phrase フィールドで使用できます。ハイフン (-) を使用して単語を区切ることもできます。

文字	Code	文字	Code
ء	0621	س	0633
آ	0622	ش	0634

文字	Code	文字	Code
أ	0623	ص	0635
ؤ	0624	ض	0636
إ	0625	ط	0637
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

アストゥリアス語の文字セット

アストゥリアス語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)

- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
á	00E1	ñ	00F1
é	00E9	ó	00F3
í	00ED	ú	00FA
ü	00FC		

アゼルバイジャン語の文字セット

アゼルバイジャン語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ä	00E4	ğ	011F
ç	00E7	ı	0131
ö	00F6	ş	015F
ü	00FC	ə	0259
̇	0307		

アルメニア語の文字セット

アルメニア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ա	0561	ն	0574
բ	0562	շ	0575
գ	0563	ո	0576
դ	0564	չ	0577
ե	0565	պ	0578
զ	0566	ջ	0579
է	0567	ռ	057A
ը	0568	ս	057B
թ	0569	վ	057C
ժ	056A	տ	057D
ի	056B	ր	057E
լ	056C	ւ	057F
խ	056D	փ	0580
ծ	056E	ք	0581

文字	Code	文字	Code
կ	056F	ւ	0582
հ	0570	փ	0583
ձ	0571	ք	0584
ղ	0572	օ	0585
ճ	0573	ֆ	0586

バシキール語の文字セット

バシキール語のカスタムボキャブラリーでは、次の文字を `Phrase` フィールドで使用できます。

- a～z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を `Phrase` フィールドで使用することもできます。

文字	Code	文字	Code
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

文字	Code	文字	Code
и	0438	ҧ	0493
й	0439	ҙ	0497
к	043A	ҙ	0499
л	043B	ҟ	049B
м	043C	ҡ	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

文字	Code	文字	Code
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
i	0456	ӳ	04F3
ї	0457	#	04F7
j	0458	ӹ	04F9

バスク語の文字セット

バスク語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
á	00E1	ñ	00F1
é	00E9	ó	00F3
í	00ED	ú	00FA
ü	00FC		

ベラルーシ語の文字セット

ベラルーシ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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

文字	Code	文字	Code
й	0439	ы	044B
к	043A	ь	044C
л	043B	э	044D
м	043C	ю	044E
н	043D	я	044F
о	043E	ё	0451
п	043F	і	0456
р	0440	ў	045E

ベンガル語の文字セット

ベンガル語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ঁ	0981	দ	09A6
ং	0982	ধ	09A7
ঃ	0983	ন	09A8
অ	0985	প	09AA
আ	0986	ফ	09AB

文字	Code	文字	Code
ই	0987	ব	09AC
ঈ	0988	ভ	09AD
উ	0989	ম	09AE
ঊ	098A	য	09AF
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	ঽ	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

文字	Code	文字	Code
ঀ	09A0	ৈ	09C8
ঁ	09A1	ো	09CB
ং	09A2	ৌ	09CC
ঃ	09A3	্	09CD
঄	09A4	#	09CE
অ	09A5	ৗ	09D7

ボスニア語の文字セット

ボスニア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ć	0107	đ	0111
č	010D	š	0161
ž	017E		

ブルガリア語の文字セット

ブルガリア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z

- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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

カタロニア語の文字セット

カタロニア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

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

中部クルド語の文字セット

中部クルド語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ئ	0626	م	0645
ا	0627	ن	0646
ب	0628	و	0648

文字	Code	文字	Code
ت	062A	پ	067E
ج	062C	چ	0686
ح	062D	ڕ	0695
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	ه	06D5

中国語、北京語 (中国本土) 、簡体字の文字セット

中国語 (簡体字) のカスタム語彙の場合、Phrase フィールドには次のファイルに表示されている任意の文字が使用できます。

- zh-cn-character-set

SoundsLike フィールドには、以下のファイルに一覧表示されているピンイン音節を含めることができます。

- ピンイン文字セット

SoundsLike フィールドにピンイン音節を使用する場合、音節をハイフン (-) で区切ります。

Amazon Transcribe は、数字を使用して中国語 (簡体字) の 4 つの声調を表します。次の表では、「ma」という単語に対応する声調記号を示しています。

声調	声調記号	声調番号
Tone 1	māi	ma1
Tone 2	má	ma2
Tone 3	mǎ	ma3
Tone 4	mà	ma4

> **Note**
>
> 5 番目 (二ユートラル) トーンの場合、トーン 1 を使用できます。ただし、「er」を除き、トーン 2 にマッピングする必要があります。例えば、「打转儿」は「da3-zhuan4-er2」として表されます。

中国語 (簡体字) のカスタム語彙の場合、IPA フィールドを使用しませんが、カスタム語彙テーブルに IPA ヘッダーを含める必要があります。

テキスト形式の入力ファイルの例を以下に示します。この例では、スペースを使用して列を揃えています。入力ファイルでは、TAB 文字を使用して列を区切る必要があります。DisplayAs 列にのみスペースを含めます。

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

中国語、北京語 (台湾)、繁体字の文字セット

中国語 (繁体字) のカスタム語彙の場合、Phrase フィールドには次のファイルに表示されている任意の文字が使用できます。

- zh-tw-character-set

SoundsLike フィールドには、以下のファイルに一覧表示されているピンイン音節を含めることができます。

- zhuyin-character-set

SoundsLike フィールドにピンイン音節を使用する場合、音節をハイフン (-) で区切ります。

Amazon Transcribe は、数字を使用して中国語 (繁体字) の 4 つの声調を表します。次の表では、「ㄇㄚ」という単語に対応する声調記号を示しています。

声調	声調記号
Tone 1	ㄇㄚ
Tone 2	ㄇㄚˊ
Tone 3	ㄇㄚˇ
Tone 4	ㄇㄚˋ

中国語 (繁体字) のカスタム語彙の場合、IPA フィールドを使用しませんが、カスタム語彙テーブルに IPA ヘッダーを含める必要があります。

テキスト形式の入力ファイルの例を以下に示します。この例では、スペースを使用して列を揃えています。入力ファイルでは、TAB 文字を使用して列を区切る必要があります。DisplayAs 列にのみスペースを含めます。

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

```
#####     #`-###`-##´-##ˇ-###`          ###
```

クロアチア語の文字セット

クロアチア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ć	0107	đ	0111
č	010D	š	0161
ž	017E		

チェコ語の文字セット

チェコ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
á	00E1	ď	010F
é	00E9	ě	011B
í	00ED	ň	0148

文字	Code	文字	Code
ó	00F3	ř	0159
ú	00FA	š	0161
ý	00FD	ť	0165
č	010D	ů	016F
ž	017E		

デンマーク語の文字セット

デンマーク語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
Å	00C5	æ	00E6
Æ	00C6	é	00E9
Ø	00D8	ø	00F8
å	00E5		

オランダ語の文字セット

オランダ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
à	00E0	î	00EE
á	00E1	ï	00EF
â	00E2	ñ	00F1
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

英語の文字セット

英語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z

- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

エストニア語の文字セット

エストニア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ä	00E4	ü	00FC
õ	00F5	š	0161
ö	00F6	ž	017E

ペルシャ語の文字セット

ペルシャ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

文字	Code	文字	Code
ء	0621	ظ	0638
آ	0622	ع	0639
أ	0623	غ	063A
ؤ	0624	ف	0641
ئ	0626	ق	0642

文字	Code	文字	Code
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
ذ	0630	ـّ	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		

フィンランド語の文字セット

フィンランド語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a~z
- - (ハイフン)

- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ä	00E4	ö	00F6
å	00E5	š	0161
ž	017E		

フランス語の文字セット

フランス語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
À	00C0	à	00E0
Â	00C2	â	00E2
Ç	00C7	ç	00E7
È	00C8	è	00E8
É	00C9	é	00E9
Ê	00CA	ê	00EA

文字	Code	文字	Code
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
Ü	00DC	ü	00FC

ガリシア語の文字セット

ガリシア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
á	00E1	ñ	00F1
é	00E9	ó	00F3
í	00ED	ú	00FA
ü	00FC		

グルジア語の文字セット

グルジア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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

文字	Code	文字	Code
პ	10DE	ხ	10EE
ჟ	10DF	ჯ	10EF
ჰ	10F0		

ドイツ語の文字セット

ドイツ語のカスタムボキャブラリーでは、次の文字を `Phrase` フィールドで使用できます。

- a～z
- A～Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を `Phrase` フィールドで使用することもできます。

文字	Code	文字	Code
ä	00E4	Ä	00C4
ö	00F6	Ö	00D6
ü	00FC	Ü	00DC
ß	00DF		

ギリシャ語の文字セット

ギリシャ語のカスタムボキャブラリーでは、次の文字を `Phrase` フィールドで使用できます。

- a～z
- - (ハイフン)

- . (ピリオド)

以下の Unicode 文字を `Phrase` フィールドで使用することもできます。

文字	Code	文字	Code
ά	03AC	ν	03BD
έ	03AD	ξ	03BE
ή	03AE	ο	03BF
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		

グジャラート語の文字セット

グジャラート語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ં	0A81	૬	0AA6
ઃ	0A82	૧	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

文字	Code	文字	Code
ઓ	0A93	ષ	0AB7
ઔ	0A94	સ	0AB8
ક	0A95	હ	0AB9
ખ	0A96	઼	0ABC
ગ	0A97	ા	0ABE
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

ハウサ語の文字セット

ハウサ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ƙ	0199	ɓ	0253
ƴ	01B4	ɗ	0257
̃	0303		

ヘブライ語の文字セット

ヘブライ語のカスタムボキャブラリーでは、次の Unicode 文字を Phrase フィールドで使用できます。

文字	Code	文字	Code
-	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

文字	Code	文字	Code
ז	05D6	פ	05E4
ח	05D7	ץ	05E5
ט	05D8	צ	05E6
י	05D9	ק	05E7
ך	05DA	ר	05E8
כ	05DB	ש	05E9
ל	05DC	ת	05EA

ヒンディー語の文字セット

ヒンディー語のカスタムボキャブラリーでは、次の Unicode 文字を `Phrase` フィールドで使用できます。

文字	Code	文字	Code
-	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

文字	Code	文字	Code
उ	0909	य	092F
ऊ	090A	र	0930
ऋ	090B	ल	0932
ए	090F	व	0935
ऐ	0910	श	0936
ऑ	0911	ष	0937
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	ॏ	0949
ट	091F	ो	094B
ठ	0920	ौ	094C

文字	Code	文字	Code
ड	0921	़	094D
ढ	0922	ज़	095B
ण	0923	ड़	095C
त	0924	ढ़	095D

Amazon Transcribe は以下の文字をマッピングします。

文字	マッピング先
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)

ハンガリー語の文字セット

ハンガリー語のカスタムボキャブラリーでは、次の文字を `Phrase` フィールドで使用できます。

- a~z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を `Phrase` フィールドで使用することもできます。

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

アイスランド語の文字セット

アイスランド語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

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

インドネシア語の文字セット

インドネシア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

イタリア語の文字セット

イタリア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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

文字	Code	文字	Code
Ù	00D9	ù	00F9
Ü	00DC	ü	00FC

日本語の文字セット

日本語のカスタム語彙の場合、DisplayAs フィールドはすべてのひらがな、カタカナ、漢字と全角ローマ字の大文字をサポートします。

Phrase フィールドは、次のファイルに一覧表示されている文字をサポートします。

- ja-jp-character-set

カビル語の文字セット

カビル語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

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

カンナダ語の文字セット

カンナダ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ಂ	0C82	ಧ	0CA7
ಃ	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

文字	Code	文字	Code
ಔ	0C94	ಸ	0CB8
ಕ	0C95	ಹ	0CB9
ಖ	0C96	#	0CBC
ಗ	0C97	#	0CBD
ಘ	0C98	ಾ	0CBE
ಙ	0C99	ಿ	0CBF
ಚ	0C9A	ೀ	0CC0
ಛ	0C9B	ು	0CC1
ಜ	0C9C	ೂ	0CC2
ಝ	0C9D	ೃ	0CC3
ಞ	0C9E	ೆ	0CC6
ಟ	0C9F	ೇ	0CC7
ಠ	0CA0	ೈ	0CC8
ಡ	0CA1	ೊ	0CCA
ಢ	0CA2	ೋ	0CCB
ಣ	0CA3	ೌ	0CCC
ತ	0CA4	್	0CCD
ಥ	0CA5	ೕ	0CD5
ದ	0CA6	ೖ	0CD6
ೠ	0CE0		

カザフ語の「文字セット

カザフ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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

文字	Code	文字	Code
з	0437	ф	0444
х	0445	к	043A
ц	0446	у	0443
ү	04AF	ж	0436
м	043C	ғ	0493
ь	044C	а	0430
ъ	044A	ұ	04B1

キニヤルワンダ語の文字セット

キニヤルワンダ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
á	00E1	ó	00F3
â	00E2	ô	00F4
ã	00E3	ú	00FA
ç	00E7	ü	00FC
è	00E8	ā	0101

文字	Code	文字	Code
é	00E9	ē	0113
ê	00EA	ī	012B
ë	00EB	ō	014D
í	00ED	ū	016B
ï	00EF	´	0301

韓国語の文字セット

韓国語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。詳細については、Wikipedia の「ハングル音節文字」を参照してください。

キルギス語の文字セット

キルギス語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

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

文字	Code	文字	Code
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	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	һ	04BB

文字	Code	文字	Code
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

ラトビア語の文字セット

ラトビア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)

- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ā	0101	ķ	0137
č	010D	ļ	013C
ē	0113	ņ	0146
ģ	0123	š	0161
ī	012B	ū	016B
ž	017E		

リトアニア語の文字セット

リトアニア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a~z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ą	0105	į	012F
č	010D	š	0161
ę	0119	ų	0173
ė	0117	ū	016B

文字	Code	文字	Code
ž	017E		

ルガンダ語の文字セット

ルガンダ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ÿ	00FF	ŋ	014B

マケドニア語の文字セット

マケドニア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B

文字	Code	文字	Code
г	0433	ќ	045C
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	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

文字	Code	文字	Code
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	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

マレー語の文字セット

マレー語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

マラヤーラム語の文字セット

マラヤーラム語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ം	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
ഏ	0D0E	ല	0D32

文字	Code	文字	Code
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
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

文字	Code	文字	Code
ത	0D24	#	0D7A
ഥ	0D25	#	0D7B
ദ	0D26	#	0D7C
ധ	0D27	#	0D7D

マルタ語の文字セット

マルタ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

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

マラーティー語の文字セット

マラーティー語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z

- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ँ	0901	थ	0925
ं	0902	द	0926
ः	0903	ध	0927
अ	0905	न	0928
आ	0906	प	092A
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

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

牧地マリ語の文字セット

牧地マリ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)

- . (ピリオド)

以下の Unicode 文字を `Phrase` フィールドで使用することもできます。

文字	Code	文字	Code
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

文字	Code	文字	Code
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

文字	Code	文字	Code
і	0456	ӯ	04F3
ї	0457	#	04F7
ј	0458	ӹ	04F9

モンゴル語の文字セット

モンゴル語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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

文字	Code	文字	Code
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	ұ	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

文字	Code	文字	Code
ю	044E	ĕ	04D7
я	044F	ə	04D9
#	0450	ʒ	04E1
ё	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	ӹ	04F9

ノルウェーブークモール語の文字セット

ノルウェーブークモール語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
å	00E5	æ	00E6

文字	Code	文字	Code
ø	00F8		

オディア/オリヤ語の文字セット

オディア/オリヤ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ଁ	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

文字	Code	文字	Code
ଏ	0B10	ଳ	0B33
ଓ	0B13	ଗ	0B36
ଔ	0B14	ଷ	0B37
କ	0B15	ସ	0B38
ଖ	0B16	ହ	0B39
ଗ	0B17	଼	0B3C
ଘ	0B18	ଽ	0B3E
ଙ	0B19	ା	0B3F
ଚ	0B1A	ି	0B40
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

文字	Code	文字	Code
#	0B71		

パシュト語の文字セット

パシュト語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a~z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
آ	0622	و	0648
أ	0623	ي	064A
ؤ	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	ٗ	0670

文字	Code	文字	Code
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
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

ポーランド語の文字セット

ポーランド語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z

- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

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

ポルトガル語の文字セット

ポルトガル語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
À	00C0	à	00E0
Á	00C1	á	00E1
Â	00C2	â	00E2

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

パンジャブ語の文字セット

パンジャブ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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
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

文字	Code	文字	Code
ਵ	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
ੳ	0A73		

ルーマニア語の文字セット

ルーマニア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
ă	0103	#	0219
â	00E2	#	021B
î	00EE	ș	015F

文字	Code	文字	Code
ț	0163		

ロシア語の文字セット

ロシア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

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

文字	Code	文字	Code
м	043C	ю	044E
н	043D	я	044F
о	043E	ё	0451

セルビア語の文字セット

セルビア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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

文字	Code	文字	Code
е	0435	ґ	0491
ж	0436	ғ	0493
з	0437	ҙ	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
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

文字	Code	文字	Code
щ	0449	#	04CA
ъ	044A	ӑ	04D1
ы	044B	ӓ	04D3
ь	044C	ӗ	04D7
э	044D	ә	04D9
ю	044E	ӡ	04E1
я	044F	ӣ	04E3
#	0450	ӧ	04E7
ё	0451	ө	04E9
ђ	0452	ӯ	04EF
ѓ	0453	ӱ	04F1
є	0454	ӳ	04F3
ѕ	0455	#	04F7
ӹ	04F9		

シンハラ語の文字セット

シンハラ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
#	0D82	#	0DAF
#	0D83	#	0DB0
#	0D85	#	0DB1
#	0D86	#	0DB3
#	0D87	#	0DB4
#	0D88	#	0DB5
#	0D89	#	0DB6
#	0D8A	#	0DB7
#	0D8B	#	0DB8
#	0D8C	#	0DB9
#	0D8D	#	0DBA
#	0D91	#	0DBB
#	0D92	#	0DBD
#	0D93	#	0DC0
#	0D94	#	0DC1
#	0D95	#	0DC2
#	0D96	#	0DC3
#	0D9A	#	0DC4
#	0D9B	#	0DC5
#	0D9C	#	0DC6

文字	Code	文字	Code
#	0D9D	#	0DCA
#	0D9E	#	0DCF
#	0D9F	#	0DD0
#	0DA0	#	0DD1
#	0DA1	#	0DD2
#	0DA2	#	0DD3
#	0DA3	#	0DD4
#	0DA4	#	0DD6
#	0DA5	#	0DD8
#	0DA7	#	0DD9
#	0DA8	##	0DDA
#	0DA9	#	0DDB
#	0DAA	##	0DDC
#	0DAB	###	0DDD
#	0DAC	##	0DDE
#	0DAD	#	0DDF
#	0DAE	#	0DF2

スロバキア語の文字セット

スロバキア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z

- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
á	00E1	ň	0148
ä	00E4	ó	00F3
č	010D	ô	00F4
ď	010F	ŕ	0155
é	00E9	š	0161
í	00ED	ť	0165
Í	013A	ú	00FA
Ľ	013E	ý	00FD
ž	017E		

スロベニア語の文字セット

スロベニア語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
č	010D	š	0161
ž	017E		

ソマリ語の文字セット

ソマリ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
s	0073	d	0064
t	0074	a	0061
a	0061	r	0072
n	006E	d	0064

スペイン語の文字セット

スペイン語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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		

スンダ語の文字セット

スンダ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
s	0073	d	0064
t	0074	a	0061
a	0061	r	0072
n	006E	d	0064

スワヒリ語の文字セット

スワヒリ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
s	0073	d	0064
t	0074	a	0061
a	0061	r	0072
n	006E	d	0064

スウェーデン語の文字セット

スウェーデン語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
Ä	00C4	ä	00E4

文字	Code	文字	Code
Å	00C5	å	00E5
Ö	00D6	ö	00F6

タガログ語/フィリピン語の文字セット

タガログ語/フィリピン語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code		
ñ	00F1		

タミル語の文字セット

タミル語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

文字	Code	文字	Code
அ	0B85	ர	0BB0
ஆ	0B86	ல	0BB2
இ	0B87	வ	0BB5
ஈ	0B88	ழ	0BB4
உ	0B89	ள	0BB3

文字	Code	文字	Code
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
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

タタール語の文字セット

タタール語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C
д	0434	ѝ	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	ҝ	049F
н	043D	ҡ	04A1

文字	Code	文字	Code
о	043E	ӈ	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
я	044F	ә	04D9
#	0450	ӡ	04E1
ё	0451	ӣ	04E3

文字	Code	文字	Code
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
ѕ	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	#	04F7
ј	0458	ӹ	04F9

テルグ語の文字セット

テルグ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

文字	Code	文字	Code
-	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

文字	Code	文字	Code
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	ౌ	0C14
ష	0C37	క	0C15
స	0C38	ఖ	0C16
హ	0C39	గ	0C17
ా	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

文字	Code	文字	Code
ొ	0C4A	ౡ	0C21
ో	0C4B	ౢ	0C22
ౌ	0C4C	ౣ	0C23
్	0C4D		

タイ語の文字セット

タイ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
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

文字	Code	文字	Code
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
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

文字	Code	文字	Code
พ	0E1E	็	0E48
ฟ	0E1F	่	0E49
ภ	0E20	้	0E4A
ม	0E21	๊	0E4B
ย	0E22	๋	0E4C
ร	0E23	์	0E4D
ฤ	0E24		

トルコ語の文字セット

トルコ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
Ç	00C7	ö	00F6
Ö	00D6	û	00FB
Ü	00DC	ü	00FC
â	00E2	Ğ	011E

文字	Code	文字	Code
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		

ウクライナ語の文字セット

ウクライナ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
а	0430	р	0440
б	0431	с	0441
в	0432	т	0442
г	0433	у	0443
д	0434	ф	0444

文字	Code	文字	Code
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		

ウイグル語の文字セット

ウイグル語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
#	0611	ۋ	0648

文字	Code	文字	Code
#	0613	ى	0649
#	0614	ي	064A
ء	0621	ً	064
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
خ	062E	ٮ	067A
د	062F	پ	067B
ذ	0630	ټ	067C
ر	0631	ٽ	067D
ز	0632	پ	067E

文字	Code	文字	Code
س	0633	ٹ	067F
ش	0634	پ	0680
ص	0635	څ	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	ڏ	068F
ن	0646	ڑ	0691
ه	0647	ړ	0693
ہ	0695		

ウズベク語の文字セット

ウズベク語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z

- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

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

文字	Code	文字	Code
a	0430		

ベトナム語の文字セット

Amazon Transcribe はベトナム語の 6 つの声調を数字で表します。次の表では、「ma」という単語に対応する声調記号を示しています。

声調名	声調記号	声調番号
ngang	ma	ma1
sắc	má	ma2
huyền	mà	ma3
hỏi	mả	ma4
ngã	mã	ma5
nặng	mạ	ma6

ベトナム語のカスタムボキャブラリーでは、次の文字を `Phrase` フィールドで使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)
- & (アンパサンド)
- ; (セミコロン)
- _ (ローライン)

以下の Unicode 文字を `Phrase` フィールドで使用することもできます。

文字	Code	文字	Code
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
ý	00FD	Ý	00DD
ă	0103	Ă	0102
đ	0111	Đ	0110
ĩ	0129	Ĩ	0128
ũ	0169	Ũ	0168

文字	Code	文字	Code
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	1EA6	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
ẻ	1EBB	Ẻ	1EBA
ẽ	1EBD	Ẽ	1EBC
ế	1EBF	Ế	1EBE
ề	1EC1	Ề	1EC0
ể	1EC3	Ể	1EC2

文字	Code	文字	Code
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
ợ	1EE3	Ợ	1EE2
ụ	1EE5	Ụ	1EE4
ủ	1EE7	Ủ	1EE6
ứ	1EE9	Ứ	1EE8
ừ	1EEB	Ừ	1EEA

文字	Code	文字	Code
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

ウェールズ語の文字セット

ウェールズ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
à	00E0	ò	00F2
á	00E1	ó	00F3
â	00E2	ô	00F4
ä	00E4	ö	00F6
è	00E8	ù	00F9
é	00E9	ú	00FA

文字	Code	文字	Code
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		

ウォロフ語の文字セット

ウォロフ語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
à	00E0	ê	00EA
ã	00E3	ë	00EB
ç	00E7	ñ	00F1
è	00E8	ó	00F3
é	00E9	ô	00F4
ŋ	014B		

ズールー語の文字セット

ズールー語のカスタムボキャブラリーでは、次の文字を Phrase フィールドで使用できます。

- a ~ z
- - (ハイフン)
- . (ピリオド)

以下の Unicode 文字を Phrase フィールドで使用することもできます。

文字	Code	文字	Code
s	0073	d	0064
t	0074	a	0061
a	0061	r	0072
n	006E	d	0064

Amazon Transcribe の仕組み

Amazon Transcribe機械学習モデルを使用して音声をテキストに変換する機械学習モデルを使用して音声をテキストに変換する機械学習モデルです。

トランスクリプトには、文字起こしされたテキストに加えて、各単語または句読点の信頼スコアやタイムスタンプなど、文字起こしされたコンテンツに関するデータが含まれています。出力例については、「データの入力と出力」セクションを参照してください。文字起こしに適用できる機能の一覧については、機能の概要を参照してください。

文字起こし、次の 2 つの主要なカテゴリに分類できます。

- Batch Transcribe:Amazon S3 バケットにアップロードされたメディアファイルを転記します。、AWS CLIAWS Management Console、およびさまざまな AWSSDK を使用してバッチ文字起こしを行うことができます。
- ストリーミングTranscribe: メディアストリームをリアルタイムで文字起こしします。AWS Management Console、HTTP/2、、およびさまざまな AWSSDK を使用してWebSockets、文字起こしをストリーミングできます。

バッチ文字起こしとストリーミング文字変換では、機能と言語のサポートが異なることに注意してください。詳細については、Amazon Transcribe featuresおよび「サポートされる言語」を参照してください。

トピック

- データ入力との出力
- 数字と句読点の文字起こし

> **開始するための API オペレーション**
>
> Batch: `StartTranscriptionJob`
> ストリーミング:`StartStreamTranscription`, StartStreamTranscriptionWebSocket

データ入力との出力

Amazon TranscribeAmazon S3オーディオデータをバケットまたはメディアストリーム内のメディアファイルとして取得し、テキストデータに変換します。

Amazon S3バケットに保存されているメディアファイルを文字起こしする場合は、バッチ文字起こしを実行することになります。メディアストリームを文字起こしする場合は、ストリーミング文字起こしを実行することになります。これら 2 つのプロセスには、異なるルールと要件があります。

バッチ文字起こしでは、Job キューイングすべての文字起こしジョブを同時に処理する必要がない場合に使用できます。これによりAmazon Transcribe、トランスクリプションジョブを追跡し、スロットが使用可能になったときに処理することができます。

> **Note**
>
> Amazon Transcribe分析モデルの品質を継続的に向上させるため、コンテンツを一時的に保存する場合があります。詳細については、Amazon Transcribeよくある質問を参照してください。Media Media コンテンツの削除をリクエストするにはAmazon Transcribe、でケースを開いてくださいAWS Support。

トピック

メディア形式

サポートされるメディアタイプは、バッチトランスクリプションとストリーミングトランスクリプションで異なりますが、どちらにも口スレスフォーマットが推奨されます。詳細については、次の表を参照してください。

	バッチ	ストリーミング
サポートされる形式	• アーム	• FLAC

	バッチ	ストリーミング
	• FLAC • M4A • MP3 • MP4 • Ogg • WebM • WAV	• オッグ・オーパス • PCM エンコード
推奨フォーマット	• FLAC • PCM 16 のいずれかを、ビットエンコードと共に使用します。	• FLAC • PCM 署名付き 16 ビットリトルエンディアンオーディオ (これには WAV は含まれないことに注意してください)

最良の結果を得るには、PCM 16 のいずれかを、ビットエンコードなどの可逆形式を使用します。

> **Note**
>
> ストリーミング文字変換は、すべての言語でサポートされているわけではありません。詳細については、サポート言語表の「データ入力」列を参照してください。

オーディオチャンネル

Amazon Transcribeシングルチャンネルメディアとデュアルチャンネルメディアをサポートします。現在、3 つ以上のチャンネルを含むメディアはサポートされていません。

オーディオの 1 つのチャンネルに複数のスピーカーが含まれていて、トランスクリプション出力の各スピーカーを分割してラベルを付ける場合は、スピーカーパーティショニング (ダイアライゼーション) を使用できます。

オーディオに 2 つの異なるチャンネルの音声が含まれている場合は、チャンネル識別機能を使用して、トランスクリプト内の各チャンネルを個別に文字起こしできます。

これらのオプションは両方とも 1 つのトランスクリプトファイルを生成します。

> **Note**
>
> スピーカー分割またはチャンネル識別を有効にしない場合、トランスクリプトテキストは 1 つの連続したセクションとして提供されます。

サンプルレート

バッチトランスクリプションジョブでは、サンプルレートを指定することもできますが、このパラメータはオプションです。リクエストに含める場合は、入力する値がオーディオの実際のサンプルレートと一致することを確認してください。オーディオと一致しないサンプルレートを指定すると、ジョブが失敗する可能性があります。

ストリーミング文字変換では、リクエストにサンプルレートを含める必要があります。バッチトランスクリプションジョブと同様に、入力する値がオーディオの実際のサンプルレートと一致していることを確認してください。

電話録音などの低忠実度オーディオのサンプルレートは、通常 8,000 Hz を使用します。ハイファイオーディオの場合、Amazon Transcribe 16,000 Hz ~ 48,000 Hz のいずれかを、ビットエンコードなどの値を使用します。

出力

トランスクリプションの出力は JSON 形式にあります。トランスクリプトの最初の部分には、トランスクリプト自体が段落形式で含まれ、その後に各単語と句読点に関する追加データが続きます。提供されるデータは、リクエストに含める機能によって異なります。トランスクリプトには、少なくとも各単語の開始時刻、終了時刻、および信頼スコアが含まれます。次のセクションでは、追加のオプションや機能が含まれていない基本的な文字起こしリクエストの出力例を示します。

Amazon S3バッチトランスクリプトはすべてバケットに保存されます。Amazon S3トランスクリプトを自分のバケットに保存するか、Amazon Transcribe安全なデフォルトバケットを使用するかを選択できます。バケットの作成と使用の詳細については、「Amazon S3バケットの使用」を参照してください。

Amazon S3自分の所有するバケットにトランスクリプトを保存したい場合は、トランスクリプションリクエストでバケットの URI を指定します。バッチ文字起こしジョブを開始する前に、Amazon Transcribe必ずこのバケットに書き込み権限を与えてください。独自のバケットを指定した場合、トランスクリプトは削除するまでそのバケットに残ります。

Amazon S3バケットを指定しない場合は、Amazon Transcribe安全なサービス管理バケットを使用し、トランスクリプトをダウンロードするために使用できる一時的な URI を提供します。テンポラリ URI は 15 分間有効であることに注意してください。提供された URIAccessDenied を使用してエラーが発生した場合は、トランスクリプト用の新しい一時的な URIGetTranscriptionJob の取得をリクエストしてください。

デフォルトのバケットを選択した場合、ジョブの有効期限が切れると（90 日）、履歴書は削除されます。この有効期限を過ぎてもトランスクリプトを保存したい場合は、ダウンロードする必要があります。

ストリーミングのトランスクリプトは、ストリームに使用しているのと同じ方法で返されます。

> **Tip**
>
> JSON 出力を Word turn-by-turn 形式のトランスクリプトに変換する場合は、GitHub この例 (Python3 用) を参照してください。このスクリプトは、通話後の分析トランスクリプトとダイアライゼーションを有効にした標準バッチトランスクリプトで動作します。

出力例

トランスクリプトには、段落形式で完全な文字起こしが行われ、 word-for-word その後に分類が続き、すべての単語と句読点のデータが表示されます。これには、開始時間、終了時間、信頼度スコア、タイプ (pronunciationまたはpunctuation) が含まれます。

次の例は、追加機能が含まれていない単純なバッチトランスクリプションジョブのものです。文字起こしリクエストに追加機能を適用するたびに、書き起こし出力ファイルに追加データが追加されます。

基本的なバッチトランスクリプトには、主に次の 2 つのセクションがあります。

1. transcripts: トランスクリプト全体が 1 つのテキストブロックに含まれます。
2. items:transcripts セクションの各単語と句読点に関する情報が含まれています。

文字起こしリクエストに追加機能を追加するたびに、文字起こしに追加情報が表示されます。

```
{
    "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"
}
```


数字と句読点の文字起こし

Amazon Transcribeすべての言語では、文章システムで大文字小文字の区別を使用する言語に対して、適切な単語を大文字にします。

ほとんどの言語では、数字は単語形式で文字起こしされます。ただし、メディアが英語またはドイツ語の場合は、Amazon Transcribe数字が使用される文脈によって数字の扱いが異なります。

たとえば、話者が「」と言うと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、これは次のように書き起こされます。

- 英語とドイツ語の方言:Meet me at 8:30 a.m. on June 1st at 100 Main Street with $3.50 and 1.5 chocolate bars
- その他すべての言語: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

英語とドイツ語の音声番号に関連するすべてのルールを確認するには、次の表を参照してください。

ルール	英語の方言 (音声入力 → テキスト出力)	ドイツ語の方言 (音声入力 → テキスト出力)
10 を超える基数を数値に変換します。	• "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"
「million」または「billion」の後に数字が続かない場合、「million」または「billion」が後に続く基数を、単語が後に続く数字に変換します。	• "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
10 を超える序数を数値に変換します。	• "Forty third" → 43rd • "twenty sixth avenue" → 26th avenue	• "dreiundzwanzigste" → 23 • "vierzigster" → 40 • "ich war Erster" → "ich war Erster"
分数は数値形式に変換	• "a quarter" → 1/4 • "three sixteenths" → 3/16 • "a half" → 1/2 • "a hundredth" → 1/100	分数は数値は単語形式で文字起こしされます。 • "ein Drittel" → "ein Drittel"
1 行に複数ある場合は、10 未満の数字を数字に変換します。	• "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

ルール	英語の方言 (音声入力 → テキスト出力)	ドイツ語の方言 (音声入力 → テキスト出力)
「ドット」または「ポイント」という単語は小数点として表示されます。	• "three hundred and three dot five" → 303.5 • "three point twenty three" → 3.23 • "zero point four" → 0.4 • "point three" → 0.3	小数は「,」で示します。 • "zweiundzwanzig komma drei" → 22,3
数値の後の「percent」という単語をパーセント記号 (%) に変換します。	• "twenty three percent" → 23% • "twenty three point four five percent" → 23.45%	• "fünf Prozent Hürde" → 5% Hürde • "dreiundzwanzig komma vier Prozent" → 23,4%

ルール	英語の方言 (音声入力 → テキスト出力)	ドイツ語の方言 (音声入力 → テキスト出力)
通貨の単語を記号に変換します。	数字の後の「ドル」、「米ドル」、「オーストラリアドル」、「AUD」、または「USD」という単語を、数字の前のドル記号 ($) に変換します。 • "one dollar and fifteen cents" → $1.15 • "twenty three USD" → $23 • "twenty three Australian dollars" → $23 数字の後の「pounds」、「British pounds」、または「GDB」という単語を、数字の前のポンド記号 (£) に変換します。 • "twenty three pounds" → £23 • "I have two thousand pounds" → I have £2,000 • "five pounds thirty three pence" → £5.33 数字の後の「rupees」、「Indian rupees」、または「INR」という単語を、数字の前のルピー記号 (#) に変換します。 • "twenty three rupees" → #23	「ユーロ」という単語をユーロ記号に変換します。 • "ein euro" → 1 € • "ein Euro vierzig" → 1,40 € • "ein Euro vierzig Cent" → 1,40 €

ルール	英語の方言 (音声入力 → テキスト出力)	ドイツ語の方言 (音声入力 → テキスト出力)
	• "fifty rupees thirty paise" → #50.30	
時刻は数字に変換	• "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
日付は数字に変換	• "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
複数の数字の間を「to」という単語で区切ります。	• "twenty three to thirty seven" → 23 to 37	該当しない
年は 4 桁で表されます。これは 20 世紀、21 世紀、22 世紀の年にのみ有効です。	• "nineteen sixty two" → 1962 • 「the year is twenty twelve」→ 年は2012 • "twenty nineteen" → 2019 • "twenty one thirty" → 2130	該当しない
スラッシュとダッシュを表示します。	• "fifty-five dash thirteen" → 55-13 スラッシュは表示されません。 • "fifty-five slash thirteen" → 55 slash 13	• "fünfundfünfzig Schrägstrich dreizehn" → 55/13 • "fünfundfünfzig Strich dreizehn" → 55-13

ルール	英語の方言 (音声入力 → テキスト出力)	ドイツ語の方言 (音声入力 → テキスト出力)
番号の表示	番号付きの段落は、段落記号 (§) を使用して表示されません。 • "paragraph seventeen" → paragraph 17	• "Paragraf siebzehn" → § 17

Amazon Transcribe の開始方法

文字起こしを作成する前に、いくつかの前提条件があります。

- サインアップしてAWS アカウント
- AWS CLIと SDK をインストールします（AWS Management Console文字起こしにを使用している場合は、このステップを省略できます）
- IAM認証情報の設定
- Amazon S3バケットを設定する
- IAMポリシーの作成

これらの前提条件を満たしたら、文字起こしを行う準備が整います。開始するには、次のリストからお好みの文字起こし方法を選択してください。

- AWS CLI
- AWS Management Console
- AWS SDK
- HTTP
- WebSockets

> **Tip**
>
> 当社の機能を初めて使用する場合、Amazon Transcribeまたは機能を試してみたい場合は、を使用することをお勧めしますAWS Management Console。パソコンのマイクを使ってストリーミングを開始したい場合も、これが一番簡単な方法です。

HTTP/2 を使用したストリーミングは他の文字起こし方法よりも複雑なため WebSockets 、ストリーミング文字起こしの設定これらの方法を使い始める前にセクションを確認することをお勧めします。なお、音声文字変換のストリーミングには SDK を使用することを強くお勧めします。

にサインアップAWS アカウント

無料利用枠アカウントまたは有料アカウントにサインアップできます。どちらのオプションでも、すべてにアクセスできますAWS のサービス。無料利用枠には試用期間があり、AWS のサービスその

間に使用状況を確認して見積もることができます。試用期間が終了すると、有料アカウントに移行できます。 pay-as-you-use 料金はベースで発生します。詳細については、「Amazon Transcribe料金」を参照してください。

> **Tip**
>
> アカウントを設定するときは、ID をメモしておいてください。AWS アカウント IDIAM はエンティティの作成に必要だからです。

AWS CLIと SDK のインストール

Amazon TranscribeAPI を使用するには、最初にをインストールする必要がありますAWS CLI。AWS CLI現在のバージョンはバージョン 2 です。Linux、Mac、Windows、および Docker のインストール手順については、AWS Command Line Interfaceユーザーガイドに記載されています。

AWS CLIをインストールしたら、セキュリティ認証情報と合わせて設定する必要がありますAWS リージョン。

SDKAmazon Transcribe と一緒に使用する場合は、インストール手順で希望する言語を選択してください。

- .NET
- C++
- Go
- Java V2
- JavaScript
- SDK VP VP V
- AWS SDK for Python (Boto3)(バッチ転写)
- Python (ストリーミング文字起こし)
- Ruby Vuby Vuby
- Rust (バッチトランスクリプション)
- Rust (ストリーミング文字起こし)

IAM 認証情報の設定

を作成する際はAWS アカウント、お客様のAWSアカウントのすべてのサービスやリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。このアイデンティティはAWS アカウントルートユーザーと呼ばれ、アカウントの作成に使用したメールアドレスとパスワードでサインインすることでアクセスされます。

日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報を保護し、それらを使用してルートユーザーのみが実行できるタスクを実行します。

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含むユーザーに対し、ID プロバイダーとのフェデレーションを使用して、一時的な認証情報の使用により、AWSにアクセスすることを要求します。

フェデレーティッドアイデンティティとは、ID ソースから提供された認証情報を使用して、AWS サービスにアクセスする際は、ID ソースから提供された認証情報の使用により、サービスにアクセスすることを要求します。フェデレーティッド ID が AWS アカウント にアクセスすると、ロールが継承され、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Center を使用することをお勧めします。でユーザーやグループを作成できますIAM Identity Center。または、お客様の IDAWS アカウント ソースで一連のユーザーやグループに接続して同期することもできます。詳細については、「Amazon Transcribe 向けの Identity and Access Management」を参照してください。

ベストプラクティスの詳細については、の「セキュリティのベストプラクティス」を参照してくださいIAM。IAM

Amazon S3バケットの作成

Amazon S3は安全なオブジェクトストレージサービスです。 Amazon S3ファイル (オブジェクトと呼ばれる) をコンテナ (バケットと呼ばれる) に保存します。

バッチトランスクリプションを実行するには、Amazon S3まずメディアファイルをバケットにアップロードする必要があります。Amazon S3トランスクリプション出力用のバケットを指定しない場合は、Amazon TranscribeAWSAmazon S3トランスクリプトを一時的に管理されたバケットに入れます。AWS-managed バケットの文字起こし出力は 90 日後に自動的に削除されます。

最初の S3 バケットを作成し、バケットにオブジェクトをアップロードする方法について説明します。

IAM ポリシーの作成

でアクセスを管理するにはAWS、ポリシーを作成し、IAM ID (ユーザー、グループ、ロール)AWS またはリソースにアタッチする必要があります。ポリシーは、アタッチされているエンティティの権限を定義します。たとえば、Amazon S3ロールがバケットにあるメディアファイルにアクセスできるのは、そのロールにアクセス権を付与するポリシーをそのロールにアタッチした場合のみです。そのロールをさらに制限したい場合は、Amazon S3代わりにバケット内の特定のファイルへのアクセスを制限できます。

AWSポリシーの使用の詳細については、以下を参照してください。

- のポリシーと権限IAM
- IAMポリシーの作成
- Amazon Transcribe と IAM の連携方法

使用できるポリシーの例についてはAmazon Transcribe、を参照してくださいAmazon Transcribe アイデンティティベースポリシーの例。カスタムポリシーを生成する場合は、AWSポリシージェネレーターの使用を検討してください。

、AWS Management ConsoleAWS CLI、またはAWS SDK を使用してポリシーを追加できます。手順については、「IAMID のアクセス許可の追加および削除」を参照してください。

ポリシーの形式は次のとおりです。

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

Amazon リソースネーム (ARN) は、AWSAmazon S3バケットなどのリソースを一意に識別します。ポリシーで ARN を使用して、特定のリソースを使用する特定のアクションの権限を付与できます。たとえば、Amazon S3バケットとそのサブフォルダーへの読み取りアクセスを許可する場合は、Statementトラストポリシーのセクションに次のコードを追加できます。

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

Amazon S3バケットとそのサブフォルダーにAmazon Transcribe read (GetObject,ListBucket) と write (PutObject) のアクセス権限を付与するポリシーの例を次に示します。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/*"
            ]
        }
    ]
}
```

による文字起こしAWS Management Console

AWSコンソールを使用して、文字起こしをバッチ処理およびストリーミングできます。Amazon S3バケットにあるメディアファイルを文字起こしする場合は、バッチ文字起こしを実行することになります。オーディオデータのリアルタイムストリームをトランスクリプションする場合は、ストリーミングトランスクリプションを実行することになります。

バッチトランスクリプションを開始する前に、Amazon S3まずメディアファイルをバケットにアップロードする必要があります。を使用して文字起こしをストリーミングするにはAWS Management Console、コンピューターのマイクを使用する必要があります。

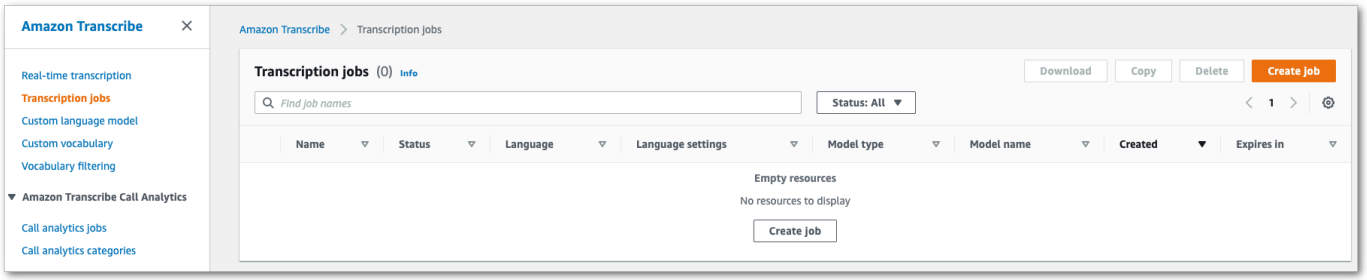
サポートされているメディアフォーマットやその他のメディア要件と制約については、を参照してくださいデータ入力との出力。

各転写方法の簡単な説明については、以下のセクションを拡張してください。

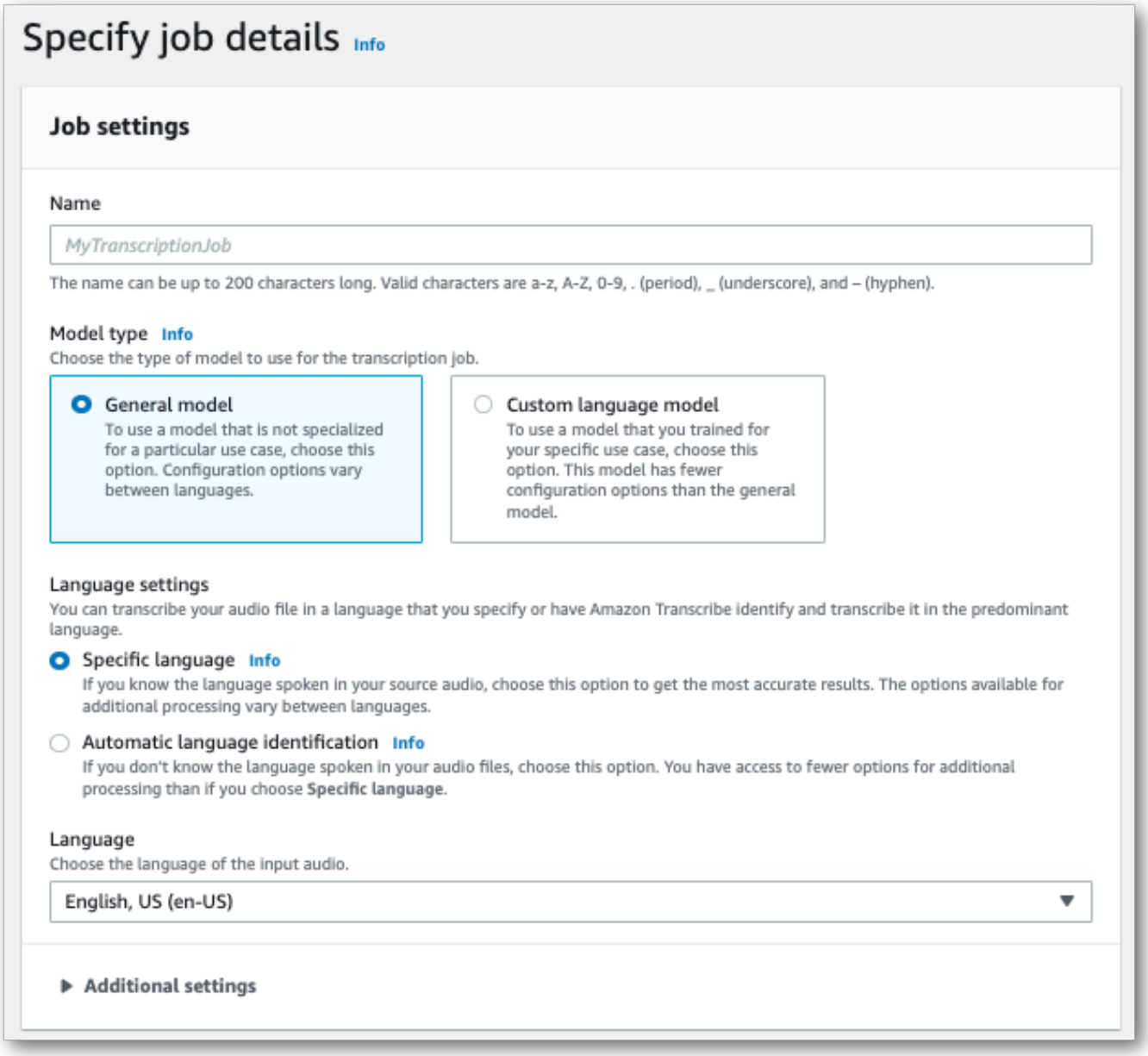
Batch 起こしのしのしのしのリング

まず、Amazon S3文字起こししたいメディアファイルがバケットにアップロードされていることを確認します。方法がわからない場合は、「Amazon S3ユーザーガイド:バケットにオブジェクトをアップロードする」を参照してください。

1. からAWS Management Console、左側のナビゲーションペインで [文字起こしジョブ] を選択します。これにより、トランスクリプションジョブのリストが表示されます。



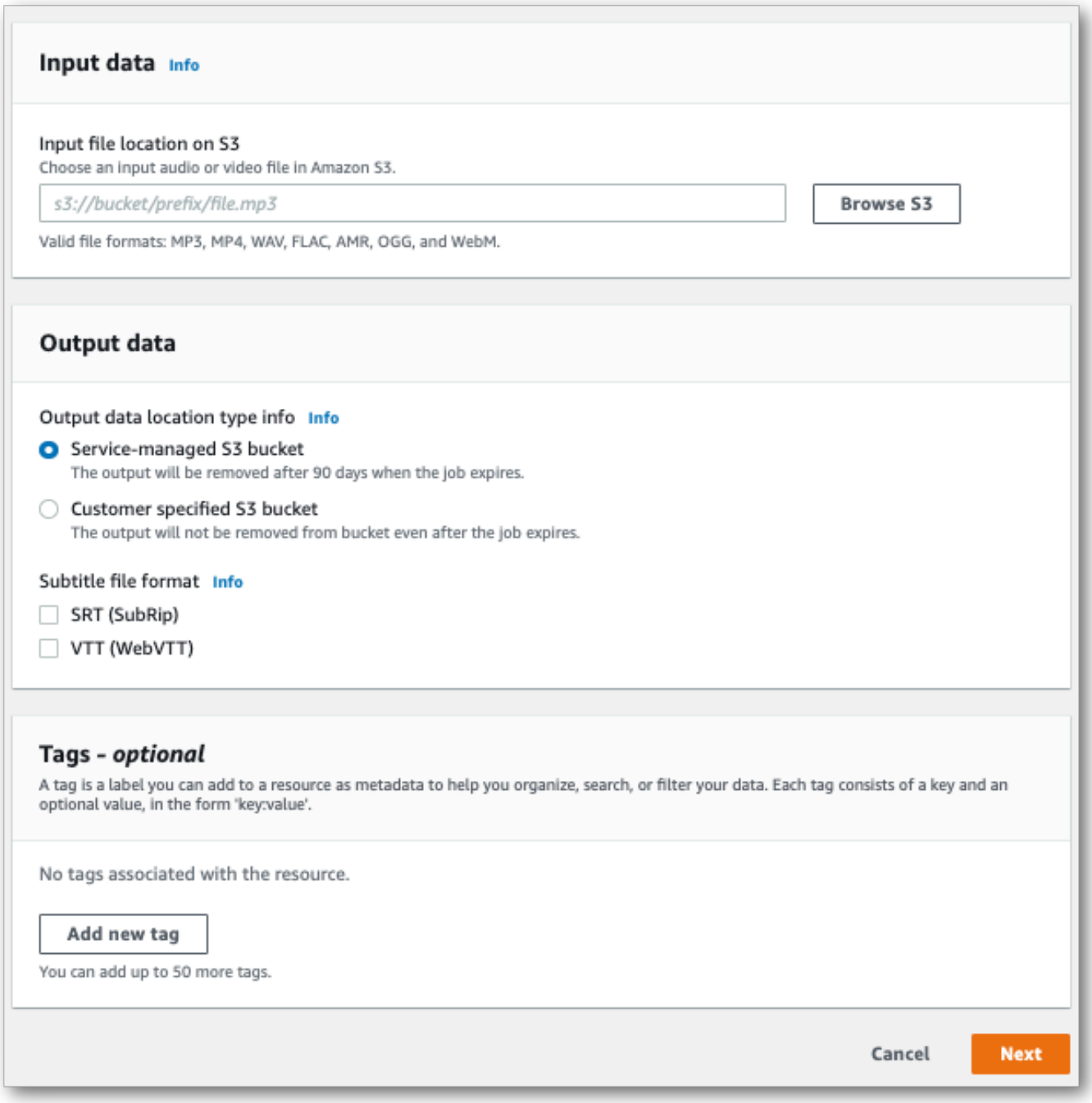
   [ジョブを作成] を選択します。
2. 「ジョブの詳細を指定」ページのフィールドに入力します。



Amazon S3入力場所はバケット内のオブジェクトでなければなりません。出力場所には、Amazon S3安全なサービス管理バケットを選択するか、Amazon S3独自のバケットを指定できます。

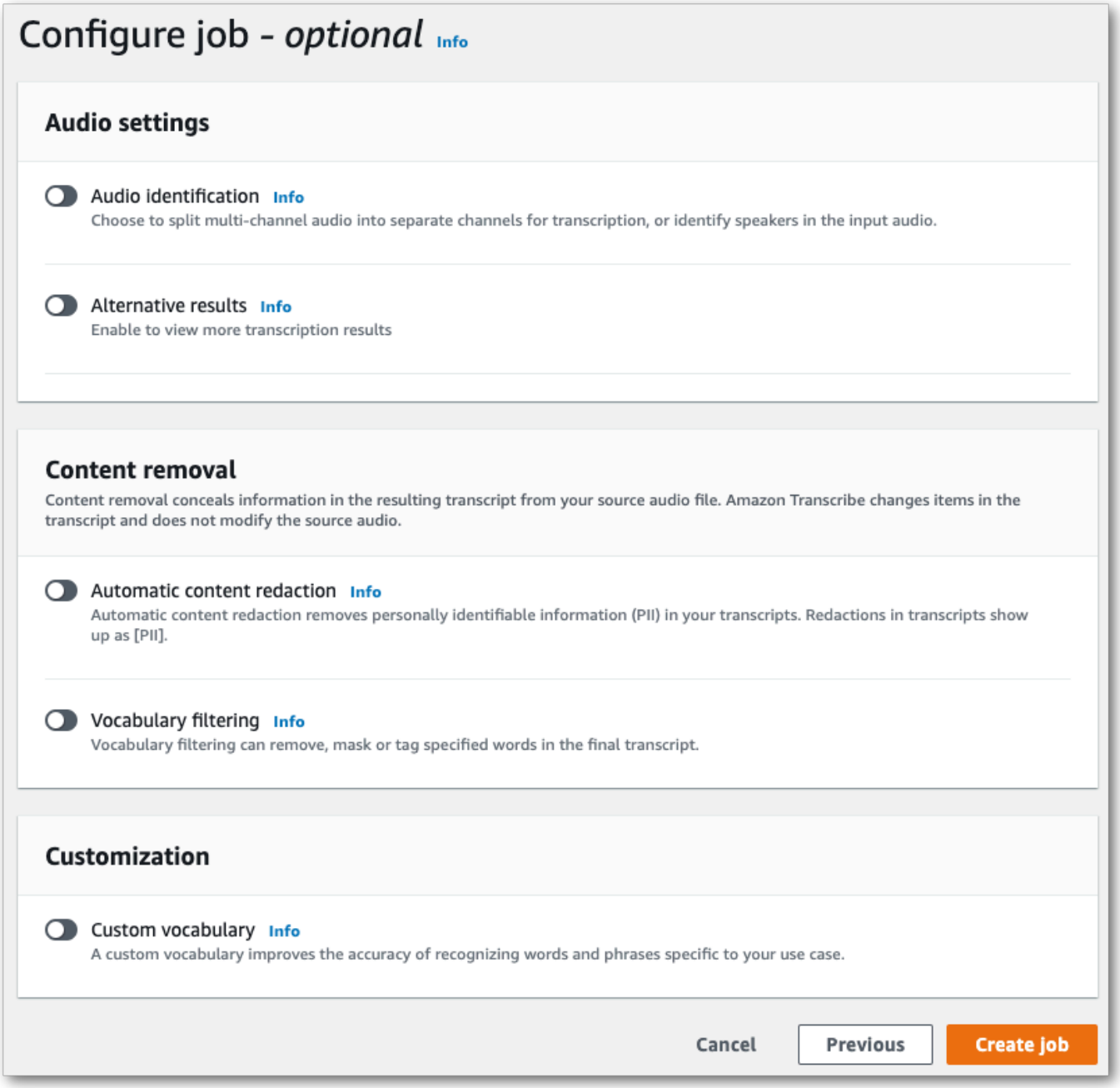
サービス管理バケットを選択すると、でトランスクリプトのプレビューを表示したりAWS Management Console、ジョブの詳細ページ（以下を参照）からトランスクリプトをダウンロードしたりできます。

Amazon S3独自のバケットを選択した場合、にプレビューが表示されないため、Amazon S3バケットに移動してトランスクリプトをダウンロードする必要があります。AWS Management Console



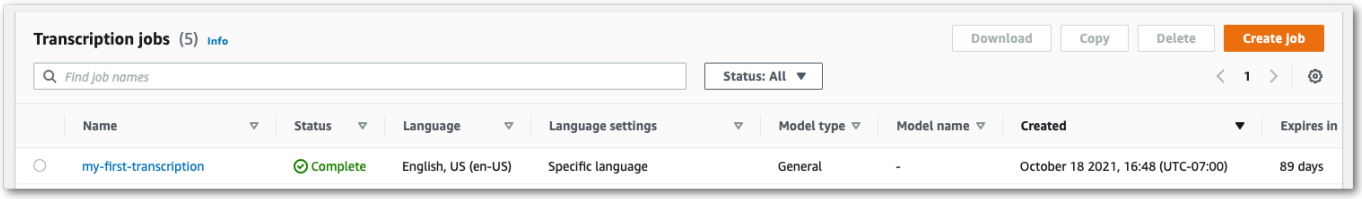
[Next] (次へ) を選択します。

3. 「ジョブの設定」ページで任意のオプションを選択します。カスタム語彙カスタム言語モデルまたはを文字起こしと一緒に使用する場合は、文字起こし作業を開始する前にこれらを作成する必要があります。



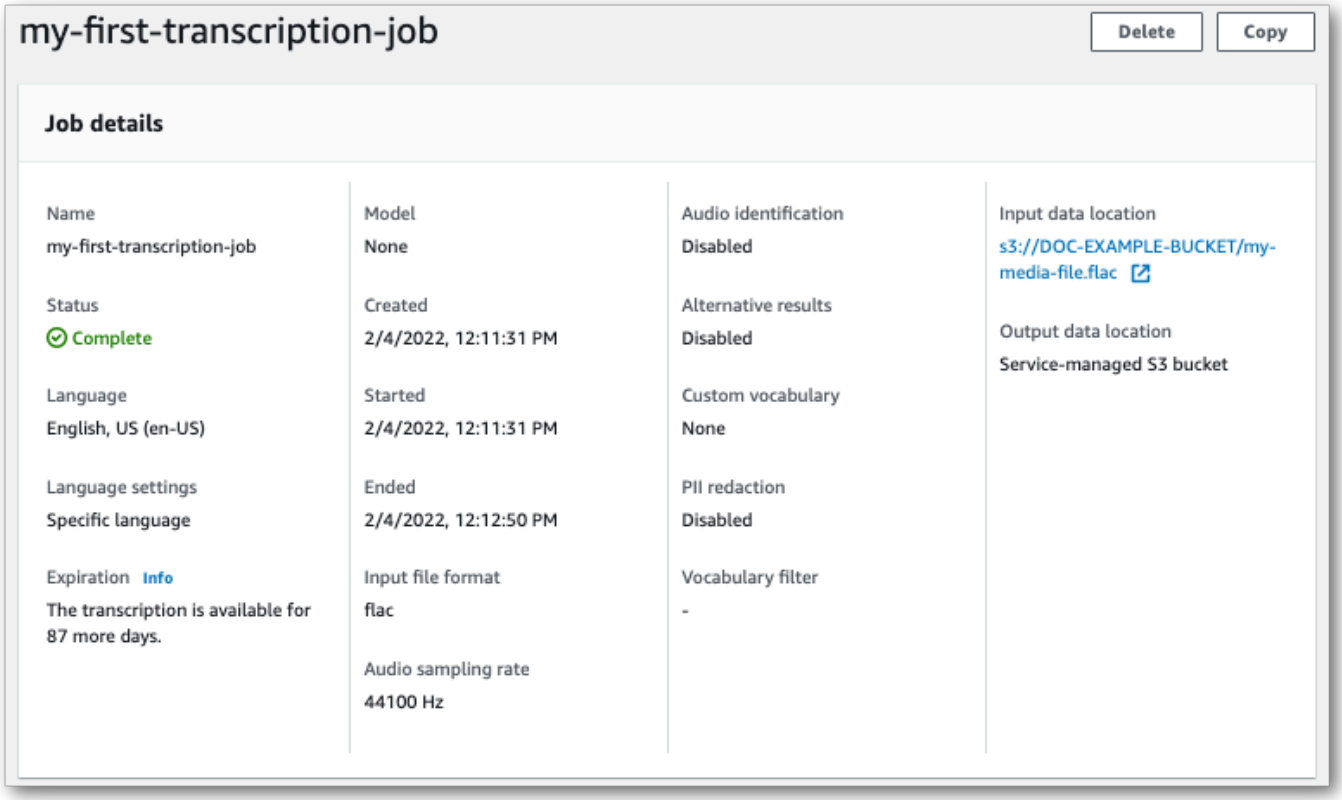
[ジョブを作成] を選択します。

4. これで、文字起こしジョブのページが表示されます。文字起こしジョブの新しい状態。完了したら、文字起こしを選択します。



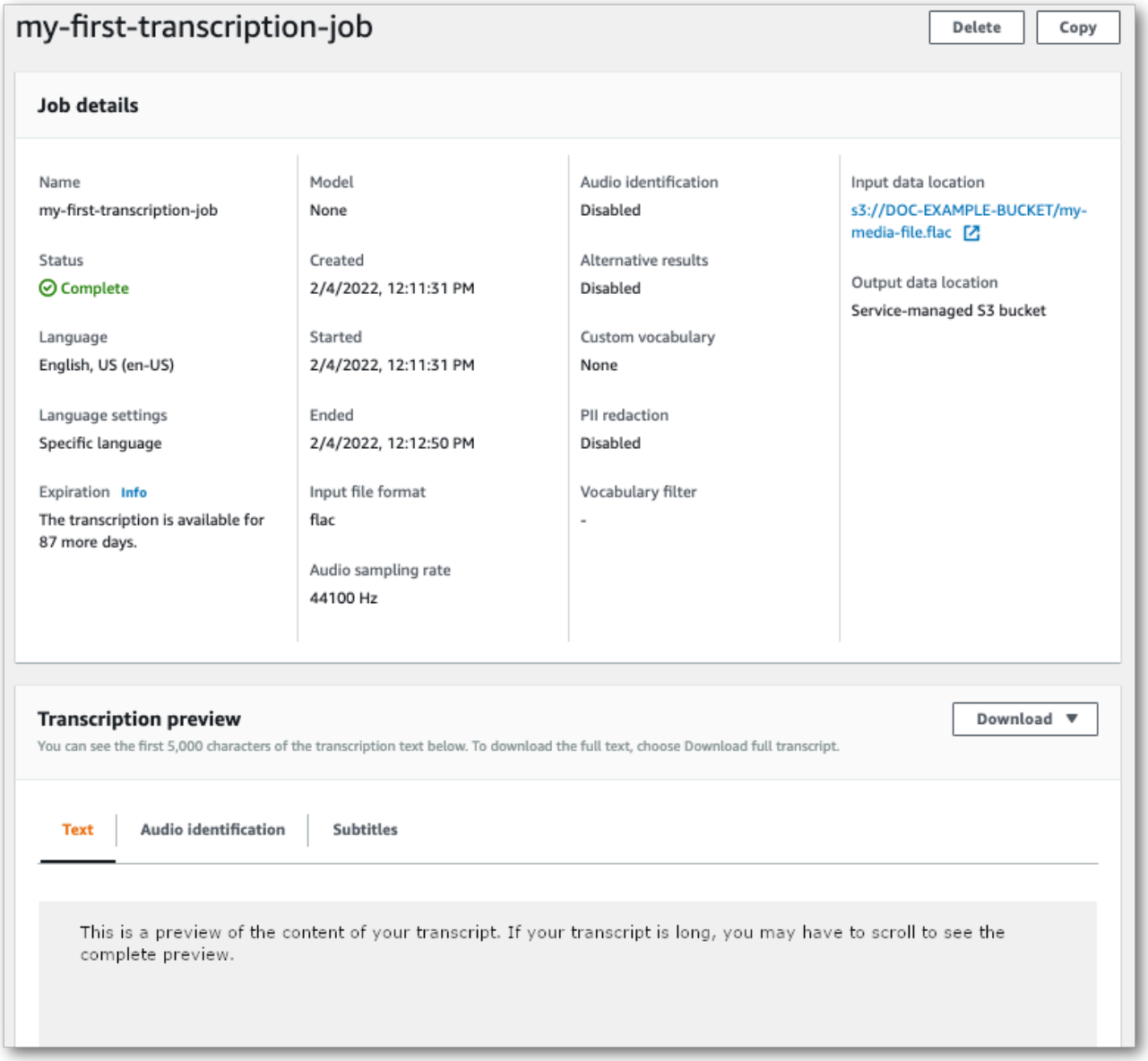
5. これで、文字起こしのJob 詳細ページが表示されています。ここでは、文字起こしジョブの設定時に指定したすべてのオプションを表示できます。

   トランスクリプトを表示するには、右側の列の [出力データの場所] でリンクされているファイルパスを選択します。これにより、Amazon S3指定した出力フォルダーに移動します。.json 拡張子が付いた出力ファイルを選択します。



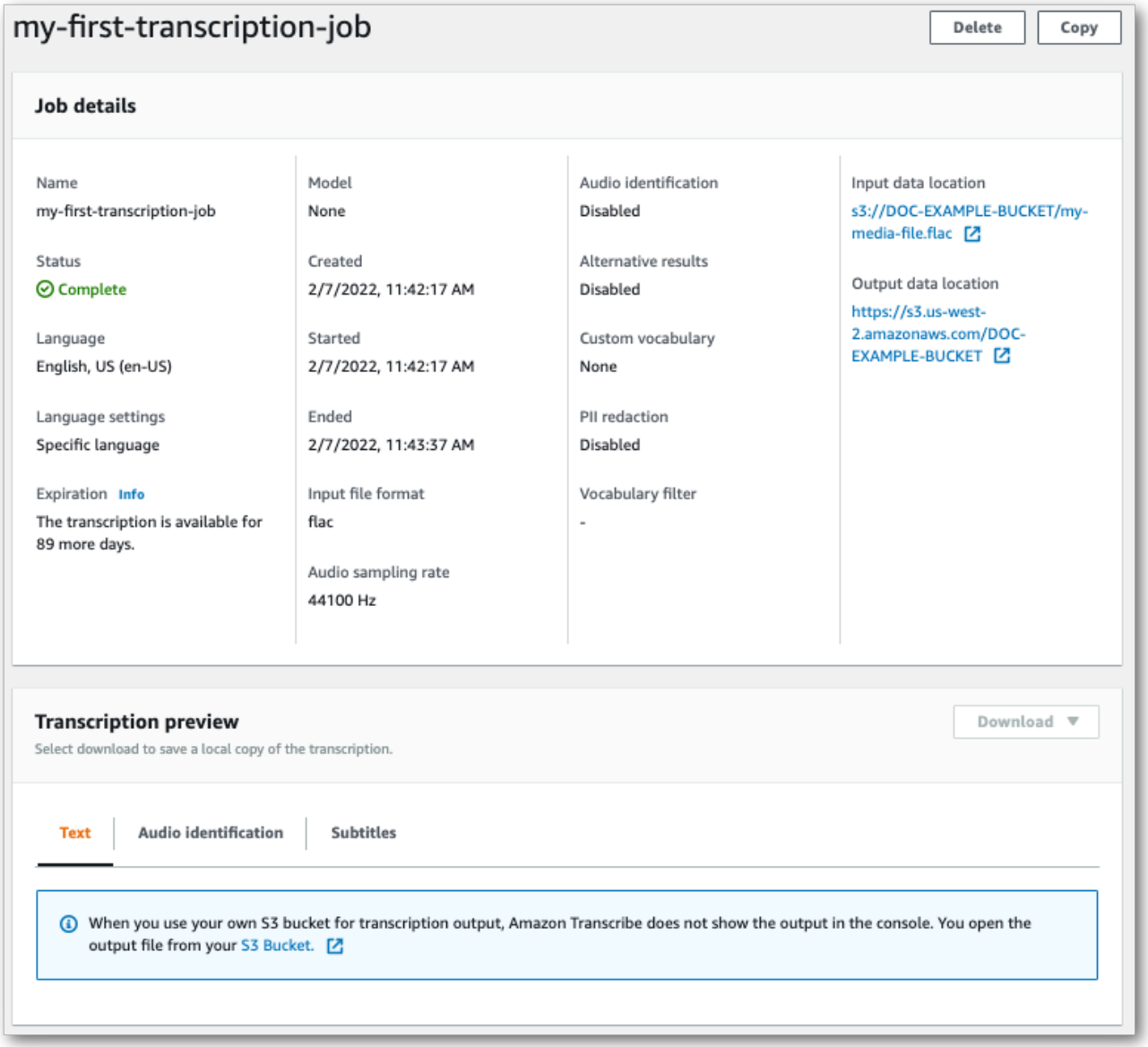
6. トランスクリプトのダウンロード方法は、Amazon S3サービス管理バケットを選択したか、Amazon S3独自のバケットを選択したかによって異なります。

   a. サービス管理バケットを選択した場合、トランスクリプションジョブの情報ページに [トランスクリプション] プレビューペインと [ダウンロード] ボタンが表示されます。



[ダウンロード] を選択し、[トランスクリプトをダウンロード] を選択します。

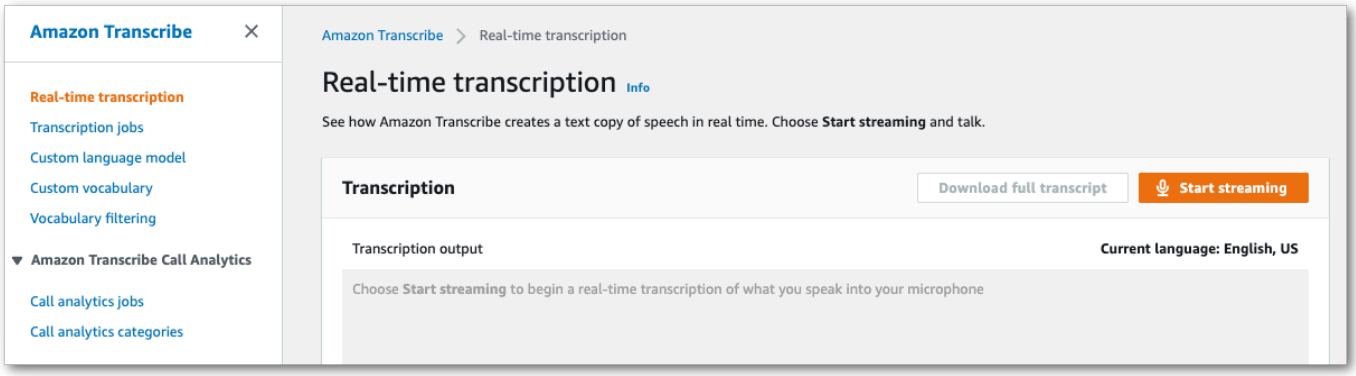
b. Amazon S3独自のバケットを選択した場合、文字起こしジョブの情報ページの文字起こしプレビューペインにテキストは表示されません。代わりに、Amazon S3選択したバケットへのリンクが記載された青い情報ボックスが表示されます。



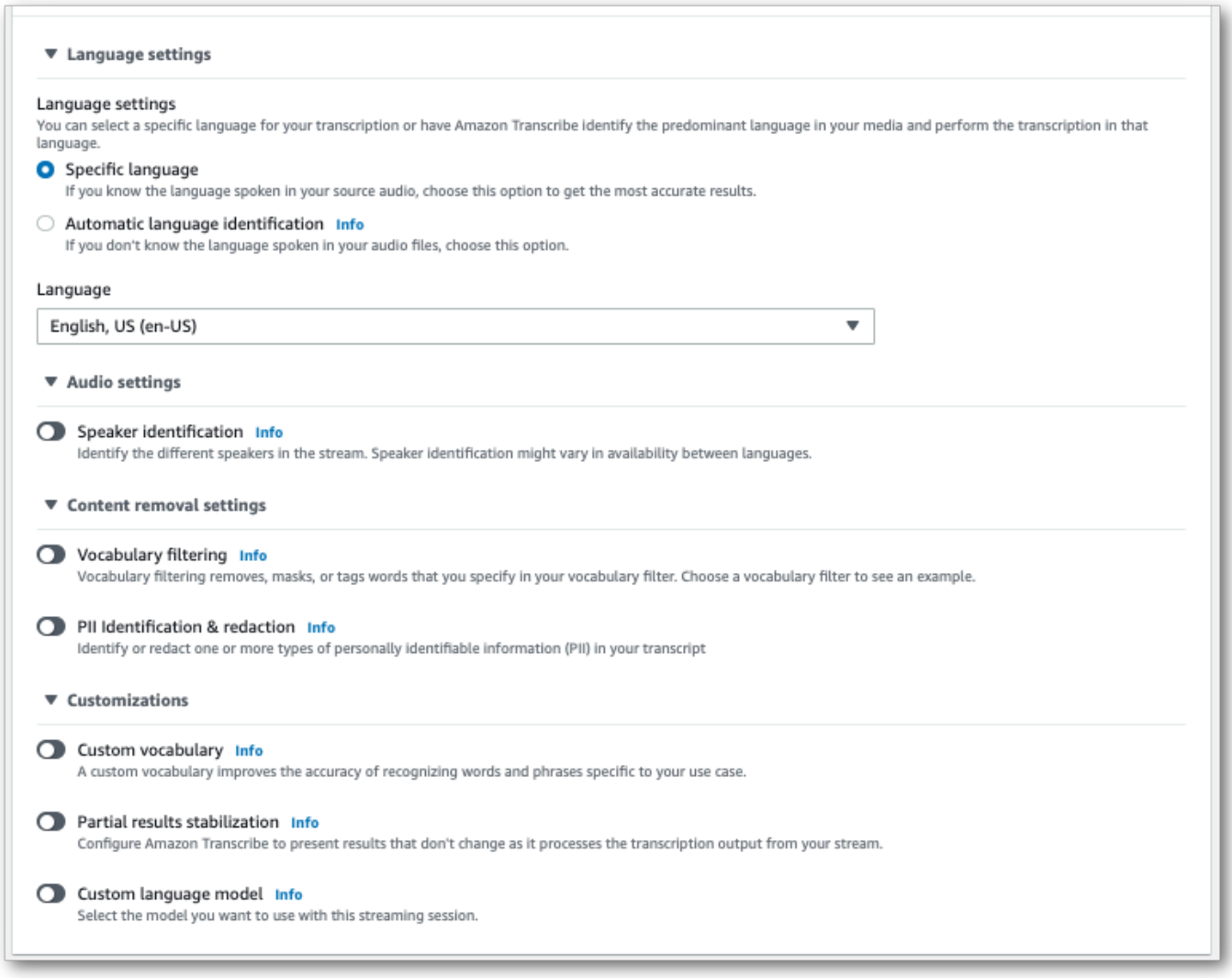
トランスクリプトにアクセスするには、Job の詳細ペインの [出力データの場所] の下のリンク、または文字起こしプレビューペインの青い情報ボックス内の [S3 Bucket] リンクを使用して、Amazon S3指定したバケットに移動します。

ストリーミングトランスクリプション

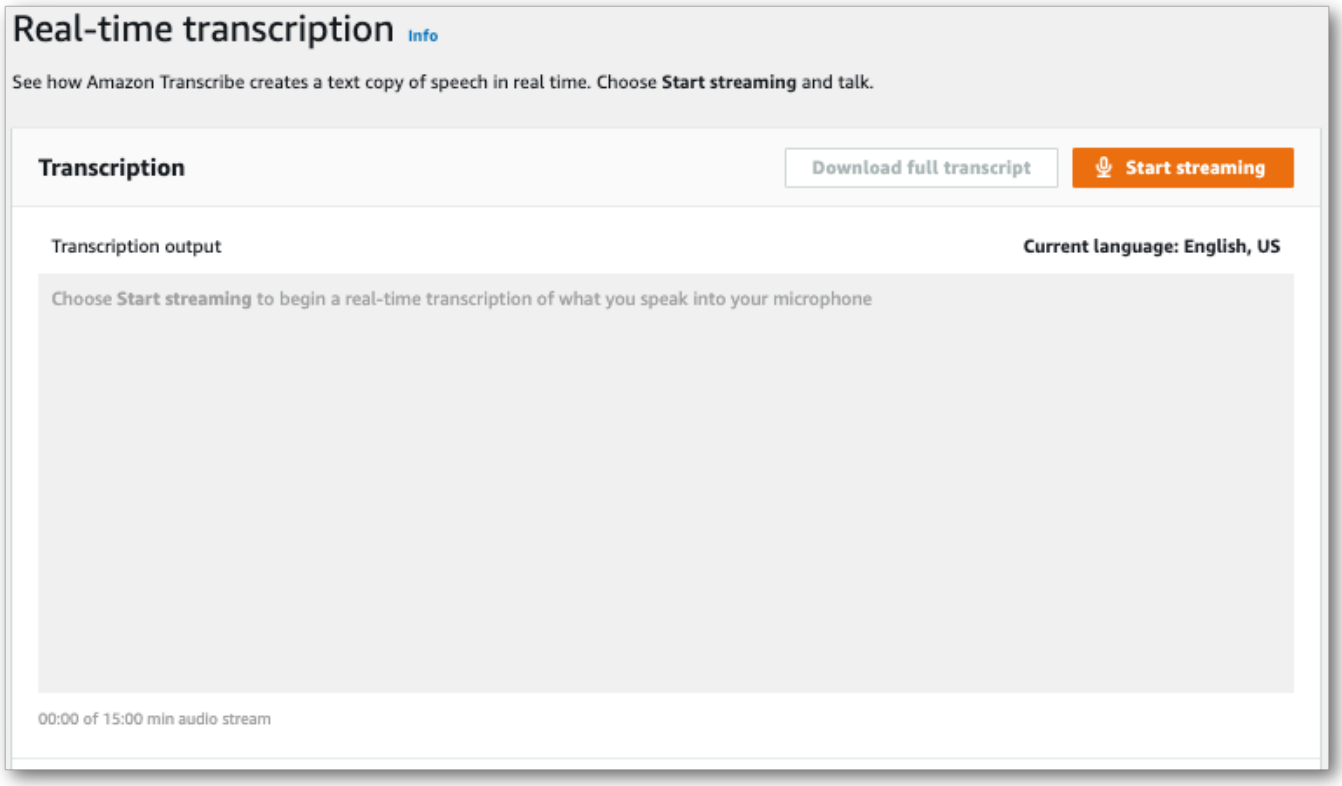
1. からAWS Management Console、左側のナビゲーションペインで [リアルタイム文字起こし] を選択します。これにより、メインのストリーミングページに移動し、ストリームを開始する前にオプションを選択できます。



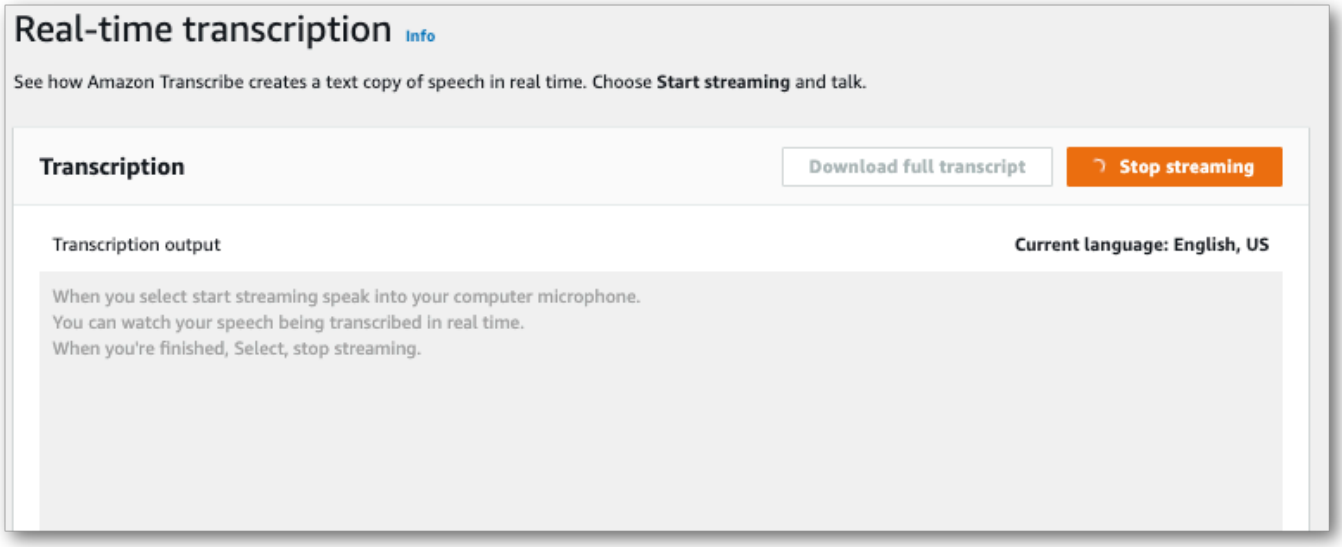
2. トランスクリプション出力ボックスの下には、さまざまな言語とオーディオ設定を選択するオプションがあります。



3. 適切な設定を選択したら、ページの一番上までスクロールして [ストリーミングを開始] を選択し、コンピューターのマイクに向かって話し始めます。スピーチの文字起こしをリアルタイムで見ることができます。



4. 終了したら、[ストリーミングを停止] を選択します。



これで、「受講実績をすべてダウンロード」を選択して成績証明書をダウンロードできます。

による文字起こしAWS CLI

を使用して文字変換を開始すると、すべてのコマンドを CLI レベルで実行できます。AWS CLIまたは、使用するコマンドを実行してから、リクエスト本文を含む JSONAWS リージョン ファイルの場所とを入力することもできます。このガイドの例では、両方の方法を示していますが、このセクションでは前者の方法に焦点を当てます。

AWS CLIはストリーミング文字変換をサポートしていません。

次に進む前に、次のことを確認してください。

- Amazon S3メディアファイルをバケットにアップロードしました。バケットの作成方法やファイルのアップロード方法がわからない場合は、「Amazon S3最初のバケットを作成する」と「Amazon S3バケットにオブジェクトをアップロードする」を参照してください。
- をインストールしましたAWS CLI。

のすべてのコマンドは、「AWS CLIAWS CLIコマンドリファレンス」Amazon Transcribe に記載されています。

新しい文字起こしジョブの開始

新しい文字起こしを開始するには、`start-transcription-job`コマンドを使用します。

1. のターミナルウィンドウに、以下のコマンドを入力します。

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

   次の行に>「」が表示され、次のステップで説明するように、必要なパラメータを追加し続けることができます。

   また、'\' を省略してすべてのパラメータをスペースで区切って追加することもできます。

2. `start-transcription-job`コマンドには、、、`region`、`transcription-job-name`、`media`、`language-code`およびまたはのいずれかを含める必要がありますidentify-language。

   `output-bucket-name`出力場所を指定する場合はリクエストに含めてください。指定した出力バケットのサブフォルダーを指定する場合は、それも含めてください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
```

すべてのパラメータを追加すると、このリクエストは次のようになります。

```
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
```

を使用して出力バケットを指定しない場合はoutput-bucket-name、Amazon Transcribe文字起こし出力をサービス管理のバケットに入れます。サービス管理バケットに保存されたトランスクリプトは、90 日後に期限切れになります。

Amazon Transcribe次のように応答します。

```
{
    "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"
    }
}
```

TranscriptionJobStatusIN_PROGRESSからに変更すると、COMPLETED文字起こし作業は正常に完了します。更新された内容を確認するにはTranscriptionJobStatus、次のセクションに示すようにget-transcription-job orlist-transcription-job コマンドを使用します。

文字起こしジョブのステータスの取得

文字起こしジョブの取得には、get-transcription-job以下のコマンドを使用します。

このコマンドに必要なパラメータは、AWS リージョンジョブの場所とジョブの名前のみです。

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

Amazon Transcribe次のように応答します。

```
{
    "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
        }
    }
}
```

Amazon S3文字起こし出力用に独自のバケットを選択した場合、このバケットにはが表示されますTranscriptFileUri。サービス管理バケットを選択した場合、一時的な URI が提供されます。この URI を使用してトランスクリプトをダウンロードしてください。

> **Note**
>
> Amazon S3サービス管理バケットの一時的な URI は 15 分間のみ有効です。URIAccesDenied の使用中にエラーが発生した場合は、get-transcription-jobリクエストを再実行して新しい一時的な URI を取得します。

文字起こしジョブの表示

特定の文字起こしジョブをすべて一覧表示するにはAWS リージョン、`list-transcription-jobs`コマンドを使用します。

このコマンドに必要なパラメータは、文字起こしジョブが置かれている場所だけです。AWS リージョン

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

Amazon Transcribe次のように応答します。

```
{
    "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"
        }
    ]
}
```

文字起こしジョブの削除

文字起こしジョブを削除するには、`delete-transcription-job`コマンドを使用します。

このコマンドに必要なパラメータは、AWS リージョンジョブの場所とジョブの名前のみです。

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

削除リクエストが成功したことを確認するには、`list-transcription-jobs`コマンドを実行します。ジョブがリストにないことがわかります。

AWS SDKs を使用した文字起こし

SDK は、バッチおよびストリーミング文字起こしの両方に使用できます。 Amazon S3 バケットにあるメディアファイルを文字起こしする場合は、バッチ文字起こしを実行します。音声データのリアルタイムストリームを文字起こしする場合は、ストリーミング文字起こしを実行していることになります。

で使用できるプログラミング言語のリストについては、 Amazon Transcribe「」を参照してくださいサポートされているプログラミング言語。ストリーミング文字起こしは、すべての AWS SDKsでサポートされているわけではないことに注意してください。サポートされているメディア形式とその他のメディア要件と制約については、「データ入力との出力」を参照してください。

使用可能なすべての AWS SDKs「 で構築するツール AWS」を参照してください。

> **Tip**
>
> 機能固有の例、シナリオ例、クロスサービス例など、 AWS SDKs「」のSDK を使用したAmazon Transcribe のコード例 AWS SDKs章を参照してください。
> SDK コードサンプルは、次の GitHub リポジトリにも記載されています。
>
> - AWS コードの例
> - Amazon Transcribe 例

バッチ文字起こし

Amazon S3 バケットにあるメディアファイルの URI を使用して、バッチ文字起こしを作成できます。 Amazon S3 バケットの作成方法やファイルのアップロード方法がわからない場合は、「最初のS3 バケットの作成」および「バケットへのオブジェクトのアップロード」を参照してください。

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()
```

ストリーミング文字起こし

ストリーミングされたメディアファイルまたはライブメディアストリームを使用して、ストリーミング文字起こしを作成できます。

標準 AWS SDK for Python (Boto3) はストリーミングでは Amazon Transcribe サポートされていないことに注意してください。Python を使用してストリーミング文字起こしを開始するには、この非同期 Python SDK for Amazon Transcribeを使用します。

Java

次の例は、音声ストリーミングを書き起こす Java プログラムです。

この例を実行するには、以下の点に注意します。

- AWS SDK for Java 2.x を使用する必要があります。
- クライアントは AWS for Java 2.x との互換性を保つために、Java 1.8 を使用する必要があります。
- 指定するサンプルレートは、音声ストリームの実際のサンプルレートと一致する必要があります。

Amazon Transcribe 「ストリーミング用のクライアントを再試行する (Java SDK)」も参照してください。このコードは Amazon Transcribe への接続を管理し、接続にエラーがあるとデータの送信を再試行します。たとえば、ネットワーク上で一時的なエラーが発生した場合、このクライアントは失敗したリクエストを再送信します。

```
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

次の例は、音声ストリーミングを書き起こす Python プログラムです。

この例を実行するには、以下の点に注意します。

- この SDK for Python を使用する必要があります。
- 指定するサンプルレートは、音声ストリームの実際のサンプルレートと一致する必要があります。

```
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++

ストリーミング C++ SDK の例については、コード例の章を参照してください。

AWS SDK でこのサービスを使用する

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	PowerShell コード例のツール
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

このサービスに固有の例については、「SDK を使用した Amazon Transcribe のコード例 AWS SDKs」を参照してください。

> **可用性の例**
>
> 必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

HTTP による文字起こしまたは WebSockets

Amazon Transcribeバッチ (HTTP/1.1) とストリーミング (HTTP/2) の両方のトランスクリプションで HTTP をサポートします。 WebSockets ストリーミング文字変換がサポートされています。

Amazon S3バケットにあるメディアファイルを文字起こしする場合は、バッチ文字起こしを実行することになります。オーディオデータのリアルタイムストリームをトランスクリプションする場合は、ストリーミングトランスクリプションを実行することになります。

どちらも HTTP で WebSockets 、AWS署名バージョン 4 ヘッダーを使用してリクエストを認証する必要があります。詳細については、「AWSAPI リクエストへの署名」を参照してください。

Batch 文字起こし

次のヘッダーを使用してバッチ HTTP リクエストを作成できます。

- host
- x-amz-target
- コンテンツタイプ
- x-amz-content-sha256
- x-amz-date
- 認可

例を示しますStartTranscriptionJob。

```
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/"
}
```

その他の操作とパラメータは「API リファレンス」に、すべてのAWS API 操作に共通するパラメータは「共通パラメータ」セクションに記載されています。その他の署名要素については、「AWS署名バージョン 4 リクエストの要素」で詳しく説明しています。

ストリーミングトランスクリプション

HTTP/2 WebSockets を使用して文字起こしをストリーミングしますが、SDK を使用するよりも複雑です。最初のストリームを設定する前に、ストリーミング文字起こしの設定このセクションを確認することをお勧めします。

これらの方法の詳細については、HTTP/2 ストリームの設定またはを参照してください WebSocket ストリームのセットアップ。

> **Note**
>
> 音声文字変換のストリーミングには SDK を使用することを強くお勧めします。サポートされている SDK のリストについては、を参照してくださいサポートされているプログラミング言語。

音声ストリーミングの文字起こし

Amazon Transcribe ストリーミングを使用すると、メディアコンテンツのリアルタイム文字起こしを生成できます。メディアファイルのアップロードを伴うバッチ文字起こしとは異なり、ストリーミングメディアはリアルタイムで Amazon Transcribe に配信されます。 Amazon Transcribe その後、 はトランスクリプトもリアルタイムで返します。

ストリーミングには、事前に録画されたメディア (映画、音楽、ポッドキャスト) とリアルタイムメディア (ライブニュース放送) が含まれます。の一般的なストリーミングのユースケース Amazon Transcribe には、スポーツイベントのライブクローズドキャプションや、コールセンターの音声のリアルタイムモニタリングなどがあります。

ストリーミングコンテンツは、 Amazon Transcribe が瞬時に文字起こししした一連の連続したデータパケット、つまり「チャンク」として配信されます。バッチ経由のストリーミングを使用する利点には、アプリケーションのリアルタイム speech-to-text 機能や文字起こし時間の短縮などがあります。ただし、この速度の向上により、場合によっては精度に制限が生じることがあります。

Amazon Transcribe では、ストリーミングに次のオプションが用意されています。

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

でストリーミング音声を文字起こしするには AWS Management Console、コンピュータのマイクに向かって話します。

> **Tip**
>
> SDK コードの例については、「」のAWS 「サンプルリポジトリ」を参照してください GitHub。

ストリーミング文字起こしでサポートされている音声形式は以下のとおりです。

- FLAC
- Ogg コンテナ内の OPUS エンコードされた音声

- PCM (16 ビット符号付き リトルエンディアンの音声形式のみ。WAV は含まない)

可逆形式 (FLAC または PCM) が推奨されます。

> **Note**
>
> ストリーミング文字起こしは、すべての言語でサポートされているわけではありません。詳細については、サポートされている言語の表の「データ入力」列を参照してください。

ストリーミング文字起こしの Amazon Transcribe リージョンの可用性を確認するには、Amazon Transcribe 「エンドポイントとクォータ」を参照してください。

ベストプラクティス

ストリーミング文字起こしの効率を高めるには、以下のことを推奨します。

- 可能であれば、PCM でエンコードされた音声を使用します。
- ストリーミングは、できる限りリアルタイムに近いことを確認します。
- レイテンシーは、音声チャンクのサイズによって異なります。音声タイプ (PCM など) でチャンクサイズを指定できる場合は、各チャンクを 50 ミリ秒から 200 ミリ秒の間で設定します。音声チャンクサイズは次の式で計算できます。

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

- チャンクのサイズを統一します。
- 音声チャネル数は正しく指定してください。
- シングルチャネルの PCM 音声では、各サンプルは 2 バイトで構成されるため、各チャンクは偶数のバイトで構成されている必要があります。
- デュアルチャネル の PCM 音声では、各サンプルは 4 バイトで構成されるため、各チャンクは 4 バイトの倍数である必要があります。
- 音声ストリームに音声が含まれていない場合は、同じ量の無音部分をエンコードして送信します。たとえば、PCM の無音は 0 バイトのストリームです。
- 音声には必ず正しいサンプリングレートを指定します。可能であれば、16,000 Hz のサンプリングレートで録音します。これにより、ネットワーク経由で送信される品質とデータ量の最適な妥協点が得られます。ほとんどのハイエンドマイクは 44,100 Hz または 48,000 Hz で録音されます。

ストリーミングと部分的な結果

ストリーミングはリアルタイムで機能するため、トランスクリプトは部分的な結果 で生成されます。 は、スピーカーの変更や音声の一時停止などの自然な音声セグメントに基づいて受信音声ストリームを Amazon Transcribe 分割します。文字起こしは文字起こしイベントのストリームとしてアプリケーションに返されます。セグメント全体の文字起こしが行われるまで、各レスポンスにはさらに多くの文字起こしされた音声が含まれます。

これの近似値は、次のコードブロックに示されています。AWS Management Console にサインインして [リアルタイム文字起こし] を選択し、マイクに向かって話すと、このプロセスが実際に動作していることを確認できます。話している間は、文字起こし出力ペインを見ます。

この例では、各行は音声セグメントの部分的な結果です。

```
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.
```

これらの部分的な結果は、Results オブジェクト内の文字起こし出力に含まれています。また、このオブジェクトブロックには IsPartialフィールドがあります。このフィールドが true の場合、文字起こしセグメントはまだ完成していません。不完全なセグメントと完全なセグメントの違いは以下で確認できます。

```
"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
```

完全なセグメント内に含まれる各単語には、0 と 1 の間の値である信頼スコアが関連付けられています。値が大きいほど、その単語が正しく文字起こしされる可能性が高くなります。

> **Tip**
>
> 音声セグメントの StartTime と EndTime は、文字起こし出力とビデオダイアログを同期させることができます。

低レイテンシーを必要とするアプリケーションを実行している場合は、部分的な結果の安定化を使用するとよいでしょう。

部分的な結果の安定化

Amazon Transcribe は、音声のストリーミングを開始するとすぐに文字起こし結果を返します。これらの部分的な結果は、自然な音声セグメントのレベルで最終的な結果が生成されるまで、段階的に返されます。自然な音声セグメントは、一時停止やスピーカーの変更を含む連続音声です。

Amazon Transcribe は、音声セグメントの最終文字起こし結果を生成するまで、部分的な結果を出力し続けます。音声認識では、コンテキストが増えるにつれて単語が修正されることがあるため、新しい部分的な結果が出力されるたびに、ストリーミング文字起こしはわずかに変化する可能性があります。

このプロセスでは、各音声セグメントに対して 2 つのオプションがあります。

- セグメントが完成するまで待つ
- セグメントの部分的な結果を使用する

部分的な結果の安定化により、 が完全なセグメントごとに最終的な文字起こし結果 Amazon Transcribe を生成する方法が変わります。有効にすると、部分的な結果から最後の数単語だけを変更することができます。このため、文字起こしの精度に影響が出る可能性があります。ただし、文字起

こしは部分的な結果の安定化を行わない場合よりも早く返されます。このレイテンシーの短縮は、動画の字幕やライブストリームのキャプションの生成に役立ちます。

以下の例は、部分的な結果の安定化が有効になっていない場合と有効になっている場合に、同じ音声ストリームがどのように処理されるかを示しています。安定性レベルは、低、中、高に設定できることに注意してください。安定性が低いと、最高の精度を実現します。安定性が高いと文字起こしは速くなりますが、精度はやや低下します。

「文字起こし」:	"EndTime":	"IsPartial":
部分的な結果の安定化は有効化されていません		
The 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. The Amazon is the largest rainforest on the planet. The Amazon is the largest rainforest on the planet.	0.545 1.045 1.545 2.045 2.545 3.045 3.545 4.045 4.545 5.045 5.545 6.025 6.025	true true true true true true true true true true true true false
部分的な結果の安定化が有効です (安定性が高い)		
The	0.515	true

「文字起こし」:	"EndTime":	"IsPartial":
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.	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 false

部分結果の安定化を有効にすると、 は Stableフィールド Amazon Transcribe を使用して項目が安定しているかどうかを示します。ここで、「item」は文字起こしされた単語または句読点を指します。Stable の値は true または false です。false (安定していない) とフラグが立てられたアイテムは、セグメントが文字起こしされるにつれて変化する可能性が高くなります。逆に、true (安定している) とフラグが付けられたアイテムは変わりません。

字幕が音声と一致するように、安定しない単語をレンダリングすることを選択できます。コンテキストが追加されるにつれて字幕が少し変わっても、音声と一致するかどうか分からないテキスを定期的に表示するよりか、こちらのほうがユーザーエクスペリエンスは向上します。

また、安定しない単語を斜体などの別の形式で表示して、これらの単語が変わる可能性があることを視聴者に伝えることもできます。また、部分的な結果を表示することで、一度に表示されるテキスト量を制限することができます。これは、動画キャプションのようにスペースに制約がある場合に重要になることがあります。

AWS Machine Learningブログで詳しく知る

リアルタイム文字起こしによる精度の向上について詳しくは、以下をご覧ください。

- Amazon Transcribe 部分的な結果の安定化によるストリーミング文字起こしエクスペリエンスの向上
- 「あれは何だったの?」 Amazon Transcribe を使用してライブ放送の字幕の精度を高める

部分的な結果の安定化の出力例

次の出力例は、不完全なセグメント (`"IsPartial": true`) に対する `Stable` のフラグを示しています。「to」と「Amazon」という単語は安定していないため、セグメントが確定される前に変更される可能性があることがわかります。

```
"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
        }
    ]
}
```


ストリーミング文字起こしの設定

このセクションでは、メインのストリーミングセクションをさらに詳しく説明します。これは、AWS SDK ではなく HTTP/2 または WebSockets 直接ストリームをセットアップするユーザー向けの情報を提供することを目的としています。このセクションの情報は、独自の SDK の構築にも使用できます。

> **Important**
>
> HTTP/2 と WebSockets を直接使用するのではなく、 SDKs を使用することを強くお勧めします。SDK は、データストリームを文字起こしするための最も簡単で信頼性の高い方法で

す。 AWS SDK を使用してストリーミングを開始するには、「」を参照してくださいAWS SDKs を使用した文字起こし。

HTTP/2 ストリームの設定

を使用して文字起こしリクエストをストリーミングするための HTTP/2 プロトコルの主なコンポーネント Amazon Transcribe は次のとおりです。

- ヘッダーフレーム。これには、リクエストの HTTP/2 ヘッダーと、データフレームに署名するためのシード署名として Amazon Transcribe を使用する認証ヘッダーの署名が含まれます。
- メタデータと未加工の音声バイトを含む、イベントストリームエンコーディングの 1 つ以上のメッセージフレーム。
- 終了フレーム。空の本文を持つイベントストリームエンコーディングの署名付きメッセージです。

> **Note**
>
> Amazon Transcribe は、HTTP/2 セッションごとに 1 つのストリームのみをサポートします。複数のストリーミングを使用しようとすると、文字起こしリクエストは失敗します。

1. リクエストを行う IAM ロールに次のポリシーをアタッチします。詳細については、IAM 「ポリシーの追加」を参照してください。

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

2. セッションを開始するには、HTTP/2 リクエストを Amazon Transcribeに送信します。

```
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
```

その他のオペレーションとパラメータは API リファレンスに記載されています。すべての AWS API オペレーションに共通するパラメータは「共通パラメータ」セクションをご覧ください。

Amazon Transcribe は、次のレスポンスを送信します。

```
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. 音声データを含む音声イベントを作成します。以下の表に示すヘッダを、イベントエンコードされたメッセージの音声バイトのチャンクと組み合わせます。イベントメッセージのペイロードを作成するには、raw バイト形式のバッファを使用します。

ヘッダー名のバイト長	ヘッダー名 (文字列)	ヘッダー値の型	値の文字列のバイト長	値の文字列 (UTF-8)
13	:content-type	7	24	application/octet-stream
11	:event-type	7	10	AudioEvent
13	:message-type	7	5	イベント

このリクエスト例のバイナリデータは base64 でエンコードされています。実際のリクエストでは、データはローバイトです。

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

4. 音声データを含む音声メッセージを作成します。

   a. 音声メッセージデータフレームには、現在の日付と、音声チャンクと音声ベントの署名を含むイベントエンコードのヘッダーが含まれます。

ヘッダー名のバイト長	ヘッダー名 (文字列)	ヘッダー値の型	値の文字列のバイト長	値
16	:チャンク署名	6	可変	生成された署名
5	:日付	8	8	タイムスタンプ

   このリクエストのバイナリデータは base64 でエンコードされています。実際のリクエストでは、データはローバイトです。

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

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

   b. 「署名バージョン 4 の署名文字列を作成する」に従って、署名文字列を作成します。文字列は次の形式に従います。

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

- DateTime: 署名が作成された日時。形式は YYYYMMDDTHHMMSSZ で、YYY = 年、MM = 月、DD = 日、HH = 時間、MM = 分、SS = 秒、「T」と「Z」は固定文字です。詳細については、「署名バージョン 4 で日付を扱う」を参照してください。
- Keypath: `date/region/service/aws4_request` 形式の署名範囲。例えば `20220127/us-west-2/transcribe/aws4_request` です。
- Hex: 入力を 16 進数にエンコードする関数。
- priorSignature: 前回のフレームの署名。最初のデータフレームで、ヘッダーフレームの署名を使用します。
- HexHash: 最初に入力の SHA-256 ハッシュを作成し、次に Hex 関数を使用してハッシュをエンコードする関数。
- nonSignatureHeaders: 文字列としてエンコードされた DateTime ヘッダー。
- payload: 音声イベントデータを含むバイトバッファ。

c. AWS シークレットアクセスキーから署名キーを取得し、それを使用して に署名しますstringToSign。保護レベルを高めるために、派生キーは日付、サービス、 AWS リージョンに固有になっています。詳細については、「AWS署名バージョン 4 の署名を計算する」を参照してください。

GetSignatureKey 関数を実装して署名キーを取得します。まだ署名キーを取得していない場合は、「署名バージョン 4 の署名キーを取得する方法の例」を参照してください。

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

- HMACSHA256: SHA-256 ハッシュ関数を使って署名を作成する関数。

- derivedSigningKey: 署名バージョン 4 の署名キー。
- stringToSign: データフレームに対して計算した文字列。

データフレームの署名を計算したら、日付、署名、および音声イベントペイロードを含むバイトバッファを構築します。文字起こし用にバイト配列を Amazon Transcribe に送信します。

5. 音声ストリームが完了したことを示すには、日付と署名のみを含む空のデータ終了フレームを送信します。データフレームを構築するのと同じ方法で、この終了フレームを構築します。

Amazon Transcribe は、アプリケーションに送信される文字起こしイベントのストリームで応答します。このレスポンスはイベントストリームでエンコードされます。また、標準の prelude と以下のヘッダーが含まれます。

ヘッダー名のバイト長	ヘッダー名 (文字列)	ヘッダー値の型	値の文字列のバイト長	値の文字列 (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	15	TranscriptEvent
13	:message-type	7	5	イベント

イベントは、ローバイトの形式で送信されます。この例では、バイトは base64 でエンコードされています。

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

文字起こし結果を表示するには、イベントストリームのエンコードを使用してローバイトをデコードします。

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

{
```

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

6. ストリームを終了するには、空の音声イベントを Amazon Transcribeに送信します。他の音声イベントとまったく同じように音声イベントを作成します (空のペイロードは除く)。イベントに署名し、その署名を :chunk-signature ヘッダーに含めます。以下に手順を示します。

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

HTTP/2 ストリーミングエラーの処理

メディアストリームの処理中にエラーが発生した場合、 は例外レスポンス Amazon Transcribe を送信します。レスポンスはイベントストリームでエンコードされます。

レスポンスには、標準の prelude と以下のヘッダーが含まれます。

ヘッダー名のバイト長	ヘッダー名 (文字列)	ヘッダー値の型	値の文字列のバイト長	値の文字列 (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	19	BadRequestException
13	:message-type	7	9	例外

デコードされた例外レスポンスには、以下の情報が含まれます。

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

```
Exception message
```

WebSocket ストリームのセットアップ

で文字起こしリクエストをストリーミングするためのWebSocketプロトコルの主なコンポーネント Amazon Transcribe は次のとおりです。

- アップグレードリクエスト。これには、リクエストのクエリパラメータと、 がデータフレームに署名するためのシード署名 Amazon Transcribe として使用する署名が含まれます。
- メタデータと未加工の音声バイトを含む、イベントストリームエンコーディングの 1 つ以上のメッセージフレーム。
- 終了フレーム。空の本文を持つイベントストリームエンコーディングの署名付きメッセージです。

> **Note**
>
> Amazon Transcribe は WebSocket 、セッションごとに 1 つのストリームのみをサポートします。複数のストリーミングを使用しようとすると、文字起こしリクエストは失敗します。

1. リクエストを行う IAM ロールに次のポリシーをアタッチします。詳細については、IAM 「ポリシーの追加」を参照してください。

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

2. セッションを開始するには、以下の形式で署名付き URL を作成します。読みやすくするために、改行が追加されています。

```
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 の最大値は 300 (5 分) です。

その他のオペレーションとパラメータは API リファレンスに記載されています。すべての AWS API オペレーションに共通するパラメータは「共通パラメータ」セクションをご覧ください。

リクエストの URL を作成し、署名バージョン 4 の署名を作成するには、以下の手順を参照してください。例は擬似コードで示しています。

a. 正規リクエストを作成します。正規は、リクエストからの情報を標準化された形式で含む文字列です。これにより、 がリクエスト AWS を受け取ると、URL に対して作成したのと同じ署名を計算できます。詳細については、「署名バージョン 4 の正規リクエストを作成する」を参照してください。

```
# 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. ドメインとクエリ文字列との間の URI の一部である正規 URI を作成します。

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

c. 正規ヘッダーと署名付きヘッダーを作成します。正規ヘッダーの末尾の \n に注意してください。

- 小文字のヘッダー名とそれに続くコロン (:) を追加します。
- このヘッダーの値のカンマ区切りリストを追加します。複数の値を持つヘッダーの値はソートしません。
- 改行 (\n) を追加します。

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

d. アルゴリズムをハッシュアルゴリズムに一致させます。SHA-256 を使用します。

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

e. 派生キーの範囲を日付、 AWS リージョン、サービスに絞るための認証情報スコープを作成します。例えば *20220127*/*us-west-2*/transcribe/aws4_request です。

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

f. 正規クエリ文字列を作成します。クエリ文字列値は、URI エンコードされ、名前順にソートされている必要があります。

- パラメータ名を文字コードポイントで昇順にソートします。名前が重複しているパラメータは、値でソートする必要があります。たとえば、大文字 F で始まるパラメータ名は、小文字 b で始まるパラメータ名より前に置きます。
- RFC 3986 が定義する非予約文字である A ~ Z、a ~ z、0 ~ 9、ハイフン (-)、アンダースコア (_)、ピリオド (.)、およびチルド (~) を、URI がエンコードすることはありません。
- 他のすべての文字についても、%XY によるパーセントエンコードが必要です。X および Y には 16 進数文字 (0 ~ 9 および大文字の A ~ F) が入ります。例えば、スペース文字は %20 として (一部のエンコードスキームのように '+' を含めずに) エンコードする必要があり、拡張 UTF-8 文字は %XY%ZA%BC 形式でエンコードする必要があります。

- パラメータ値の等号 (=) 文字を二重エンコードします。

```
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. ペイロードのハッシュを作成します。GET リクエストの場合、ペイロードは空の文字列です。

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

h. 以下の要素を組み合わせて正規リクエストを作成します。

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

3. リクエストについてのメタ情報が含まれる署名する文字列を作成します。次のステップでリクエストの署名を計算するときに、文字列を使用してサインインします。詳細については、「署名バージョン 4 の署名文字列を作成する」を参照してください。

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

4. 署名を計算します。これを行うには、 AWS シークレットアクセスキーから署名キーを取得します。保護レベルを高めるために、派生キーは日付、サービス、 AWS リージョンに固有になっています。派生キーを使用して、リクエストに署名します。詳細については、「署名バージョン 4 AWS の署名を計算する」を参照してください。

GetSignatureKey 関数を実装して署名キーを取得します。まだ署名キーを取得していない場合は、「署名バージョン 4 の署名キーを取得する方法の例」を参照してください。

```
#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
```

関数 HMAC(key, data) は、バイナリ形式で結果を返す HMAC-SHA 256 関数を表します。

5. 署名情報をリクエストに追加し、リクエスト URL を作成する

   署名を計算したら、クエリ文字列に追加します。詳細については、「署名をリクエストに追加する」を参照してください。

   まず、クエリ文字列に認証情報を追加します。

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

   次に、リクエストの URL を作成します。

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

   WebSocket ライブラリでリクエスト URL を使用して、 にリクエストを行います Amazon Transcribe。

6. へのリクエストには、次のヘッダーを含める Amazon Transcribe 必要があります。通常、これらのヘッダーは WebSocket クライアントライブラリによって管理されます。

   ```
   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 が WebSocket リクエストを受信すると、アップグレードレスポンスで WebSocket 応答します。通常、 WebSocket ライブラリはこのレスポンスを管理し、 との通信用のソケットを設定します Amazon Transcribe。

   からのレスポンスを次に示します Amazon Transcribe。読みやすくするために、改行が追加されています。

```
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. WebSocket ストリーミングリクエストを行います。

   WebSocket 接続が確立されると、クライアントは一連のオーディオフレームの送信を開始できます。各オーディオフレームは、イベントストリームエンコード を使用してエンコードされます。

   各データフレームには、未加工の音声バイトのチャンクと組み合わせた 3 つのヘッダーが含まれています。以下の表はこれらのヘッダーについて説明したものです。

ヘッダー名のバイト長	ヘッダー名 (文字列)	ヘッダー値の型	値の文字列のバイト長	値の文字列 (UTF-8)
13	:content-type	7	24	application/octet-stream
11	:event-type	7	10	AudioEvent
13	:message-type	7	5	イベント

9. データストリームを終了するには、空の音声チャンクをイベントストリームでエンコードされたメッセージで送信します。

   レスポンスには、イベントストリームでエンコードされた raw バイトがペイロードに含まれています。また、標準の prelude と以下のヘッダーが含まれます。

ヘッダー名のバイト長	ヘッダー名 (文字列)	ヘッダー値の型	値の文字列のバイト長	値の文字列 (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	15	TranscriptEvent
13	:message-type	7	5	イベント

   バイナリレスポンスをデコードすると、文字起こしの結果を含む JSON 構造になります。

WebSocket ストリーミングエラーの処理

リクエストの処理中に例外が発生した場合、 はイベントストリームでエンコードされたレスポンスを含むターミナル WebSocket フレームに Amazon Transcribe 応答します。このレスポンスには以下の表で示しているヘッダーがあり、レスポンスの本文には説明のエラーメッセージが含まれます。例外レスポンスを送信すると、 Amazon Transcribe はクローズフレームを送信します。

ヘッダー名のバイト長	ヘッダー名 (文字列)	ヘッダー値の型	値の文字列のバイト長	値の文字列 (UTF-8)
13	:content-type	7	16	application/json
15	:exception-type	7	varies	varies、以下を参照してください
13	:message-type	7	9	例外

`exception-type` ヘッダーには、次のいずれかの値が含まれます。

- **`BadRequestException`**: ストリームの作成時にクライアントエラーが発生したか、データのストリーミング中にエラーが発生しました。クライアントでデータの受け入れ準備ができていることを確認し、リクエストを再試行してください。
- **`InternalFailureException`**: クライアントとのハンドシェイク中に Amazon Transcribe 問題が発生しました。リクエストを再試行してください。
- **`LimitExceededException`**: クライアントで同時ストリームの制限を超えました。詳細については、「Amazon Transcribe 制限」を参照してください。文字起こしするストリームの数を減らしてください。
- **`UnrecognizedClientException`**: WebSocket アップグレードリクエストが不正なアクセスキーまたはシークレットキーで署名されました。アクセスキーを正しく作成していることを確認し、リクエストを再試行してください。

Amazon Transcribe は、一般的なサービスエラーを返すこともできます。リストについては、「共通エラー」を参照してください。

イベントストリームエンコード

Amazon Transcribe は、ストリーミング文字起こしにイベントストリームエンコーディングと呼ばれる形式を使用します。

イベントストリームエンコードは、クライアントとサーバーの間の双方向通信を提供します。Amazon Transcribe ストリーミングサービスに送信されるデータフレームは、この形式でエンコードされます。からのレスポンス Amazon Transcribe もこのエンコードを使用します。

各メッセージは、prelude と data の 2 つのセクションで構成されています。prelude の構成要素は以下のとおりです。

1. メッセージの合計バイト長
2. すべてのヘッダーを組み合わせたバイトの長さ

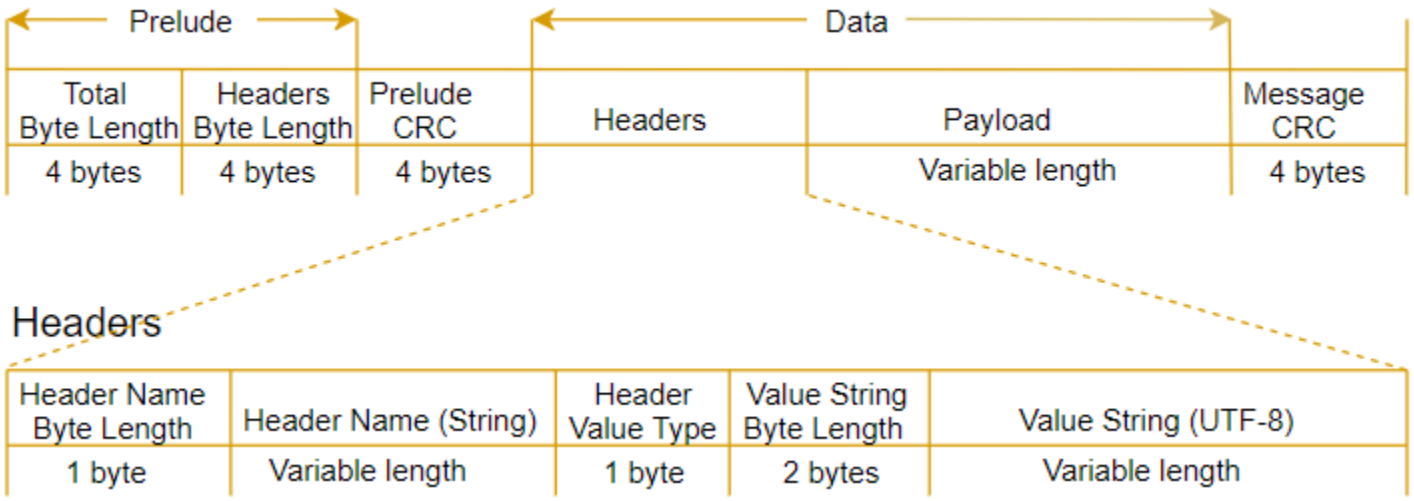
データセクションの構成要素は以下のとおりです。

1. ヘッダー
2. ペイロード

各セクションは、4 バイトのビッグエンディアン整数巡回冗長検査 (CRC) チェックサムで終わります。メッセージ CRC チェックサムは、プレリュードセクションとデータセクションの両方を対象としています。 Amazon Transcribe は、CRC32 (GZIP CRC32 とも呼ばれます) を使用して、両方の CRC を計算します。CRC32 の詳細については、「GZIP ファイル形式仕様バージョン 4.3」を参照してください。

合計メッセージオーバーヘッド (prelude と両方のチェックサムを含む) は 16 バイトです。

次の図は、メッセージとヘッダーを構成するコンポーネントを示します。メッセージごとに複数のヘッダーがあります。



各メッセージには、以下のコンポーネントが含まれます。

- Prelude: 2 つの 4 バイトのフィールドで構成され、合計は 8 バイトに固定されています。
  - 最初の 4 バイト: メッセージ全体のビッグエンディアン整数バイト長です (4 バイト長のフィールド自体を含む)。

  - 次の 4 バイト メッセージのヘッダー部分のビッグエンディアン整数バイト長 (ヘッダー長フィールド自体を除く)。
- Prelude CRC: メッセージの prelude 部分の 4 バイト CRC チェックサム (CRC 自体を除く)。prelude にはメッセージ CRC とは別の CRC があります。これにより、 Amazon Transcribe はバッファオーバーランなどのエラーを発生させることなく、破損したバイト長情報をすぐに検出できます。
- ヘッダー: メッセージタイプ、コンテンツタイプなど、メッセージに注釈を付けるメタデータ。メッセージにはキーと値のペアである複数のヘッダーがあり、キーは UTF-8 文字列です。ヘッダーは、メッセージのヘッダー部分に任意の順序で表示することができ、各ヘッダーは一度だけ表示することができます。
- ペイロード: 書き起こされた音声コンテンツ。
- メッセージ CRC: メッセージの先頭からチェックサムの先頭までの 4 バイトの CRC チェックサム。つまり、CRC を除き、メッセージ内のすべてのものです。

ヘッダーフレームは、ストリーミング transcription の認証フレームです。 は、リクエスト内のデータフレームの認証ヘッダーチェーンを生成するためのシードとして、認証ヘッダーの値 Amazon Transcribe を使用します。

各ヘッダーには以下のコンポーネントが含まれており、フレームごとに複数のヘッダーがあります。

- ヘッダー名のバイト長: ヘッダー名のバイトの長さ。
- ヘッダー名: ヘッダータイプを示すヘッダの名前。有効な値については、次のフレームの説明を参照してください。
- ヘッダー値のタイプ: ヘッダー値を示す数値。以下の表に、ヘッダーに指定できる値と、ヘッダーが示す値を示します。
  - 0 - TRUE
  - 1 - FALSE
  - 2 - BYTE
  - 3 - SHORT
  - 4 - INTEGER
  - 5 - LONG
  - 6 - BYTE ARRAY
  - 7 – STRING
  - 8 - TIMESTAMP

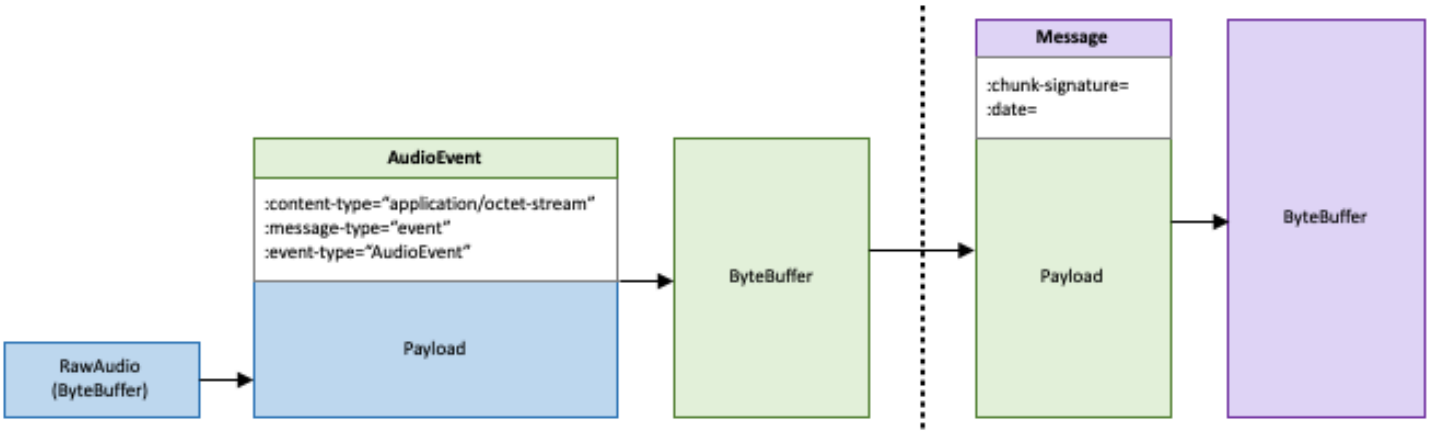
  - 9 - UUID
- 値の文字列のバイト長: ヘッダー値の文字列のバイト長。
- ヘッダー値: ヘッダー文字列の値。このフィールドの有効な値は、ヘッダーのタイプによって異なります。詳細については、「HTTP/2 ストリームの設定」または「 WebSocket ストリームのセットアップ」を参照してください。

データフレーム

各ストリーミングリクエストには、1 つ以上のデータフレームが含まれています。データフレームを作成するには 2 つのステップがあります。

1. 未加工の音声データとメタデータを組み合わせて、リクエストのペイロードを作成します。
2. ペイロードを署名と組み合わせて、 Amazon Transcribeに送信されるイベントメッセージを形成します。

この 仕組み を以下に示します。



Job キューイング

ジョブキューイングを使用すると、同時に処理できる数よりも多くの文字起こしジョブのリクエストを送信できます。ジョブキューを使用しない場合、同時リクエストの許容数に達したら、1 つ以上のリクエストが完了するまで待ってから、新しいリクエストを送信する必要があります。

転記ジョブリクエストでは、ジョJob キューイングはオプションです。コール後の分析リクエストでは、ジョブキューイングが自動的に有効になります。

ジョブキューを有効にすると、Amazon Transcribe制限を超えるすべてのリクエストを含むキューが作成されます。リクエストが完了するとすぐに、新しいリクエストがキューから取り出されて処理されます。キューに入っているリクエストは、FIFO (先入れ先出し) の順序で処理されます。

キューには最大 10,000 ジョブを追加できます。この制限を超えた場合、`LimitExceededConcurrentJobException`エラーが表示されます。最適なパフォーマンスを維持するために、キューに入っているジョブの処理にはクォータの最大 90% (帯域幅比 0.9)Amazon Transcribe しか使用しません。これらはデフォルト値で、リクエストに応じて引き上げることができます。

> **Tip**
>
> Amazon Transcribeリソースのデフォルトの制限とクォータのリストは、「AWS一般リファレンス」にあります。これらのデフォルトの一部は、リクエストに応じて引き上げることができます。

ジョブキューを有効にしても、同時リクエストのクォータを超えない場合、すべてのリクエストが同時に処理されます。

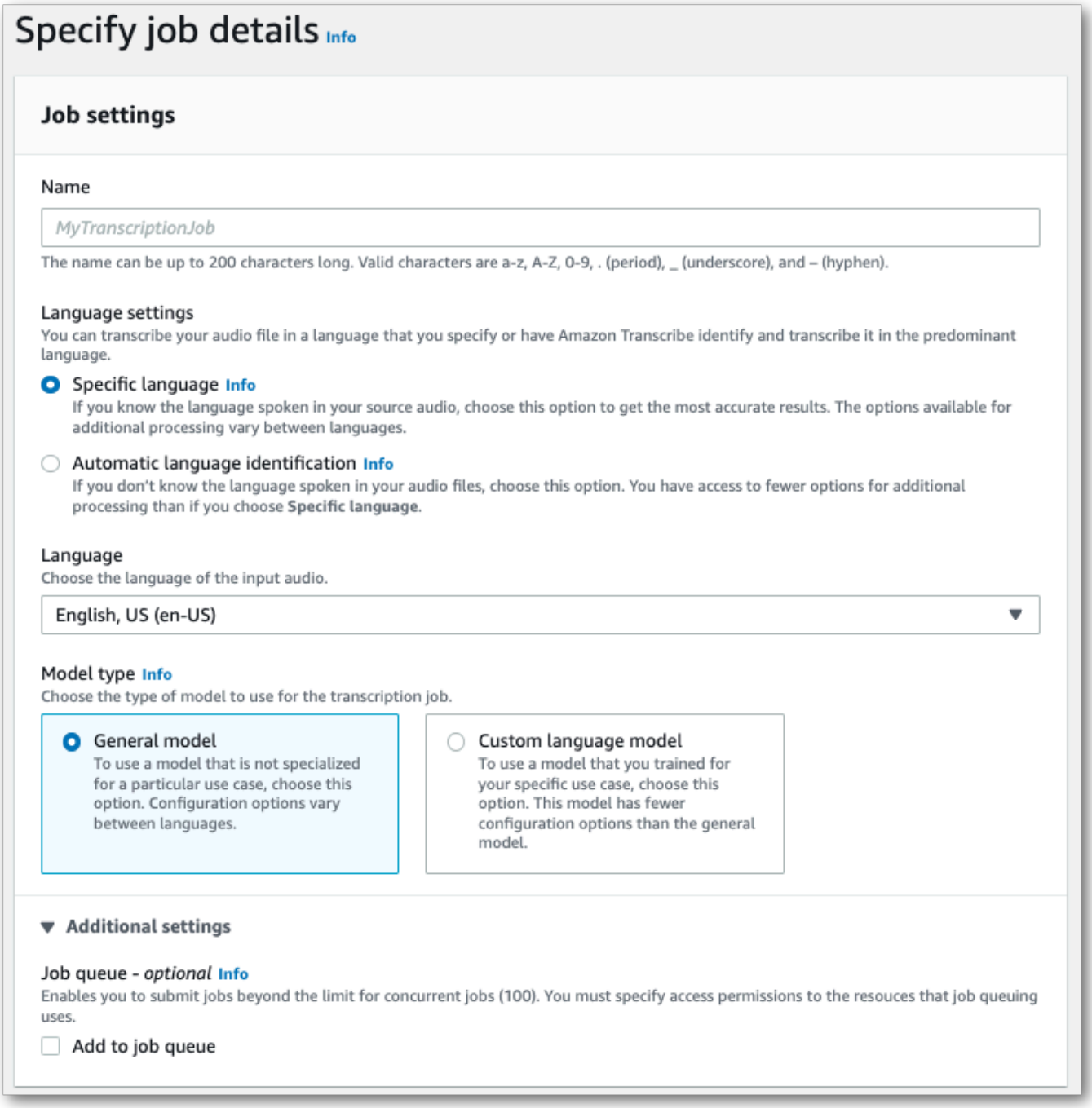
ジョブキューの有効化

、AWS Management Console、AWS CLIまたは AWSSDK を使用してジョブキューを有効にできます。例については以下を参照してください。例については以下を参照してください。

AWS Management Console

1. AWS Management Consoleにサインインします。

2. ナビゲーションペインで、変換ジョブを選択後、ジョブの作成ジョブの作成ジョブの作成ジョブの(右上) を選択します。これにより、ジョブの詳細を指定 ページが開きます。
3. Job 設定ボックスには、追加設定パネルがあります。このパネルを展開すると、「ジョブキューに追加」ボックスを選択してジョブキューを有効にできます。



4. [ジョブの詳細の指定] ページに含めたい他のフィールドを入力し、[次へ] を選択します。これにより、ジョブの設定 : オプション ページ へ移動します。

5. ジョブの作成を選択後、変換ジョブを選択します。

AWS CLI

この例では、start-transcription-jobjob-execution-settingsAllowDeferredExecutionコマンドとパラメーターをサブパラメーターと共に使用しています。AllowDeferredExecutionリクエストに含める場合は、必ず含める必要があることに注意してくださいDataAccessRoleArn。

詳細については、「StartTranscriptionJob」および「JobExecutionSettings」を参照してください。

```
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
```

次に、start-transcription-jobコマンドと、キューイングを有効にするリクエスト本文を使用する別の例を示します。

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

ファイル my-first-queueing-request.json には次のリクエストボディが入ります。

```
{
  "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)

この例では、AWS SDK for Python (Boto3)を使用して start_transcription_jobAllowDeferredExecution メソッドの引数を使用してジョブキューイングを有効にします。AllowDeferredExecutionリクエストに含める場合は、必ず含める必要があることに注意してくださいDataAccessRoleArn。詳細については、「StartTranscriptionJob」および「JobExecutionSettings」を参照してください。

機能固有、シナリオ、サービス間の例など、AWS SDK を使用するその他の例については、SDK を使用した Amazon Transcribe のコード例 AWS SDKsこの章を参照してください。

```
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)
```

キューに入っているジョブの進行状況は、AWS Management ConsoleGetTranscriptionJobまたはリクエストを送信して確認できます。ジョブがキューに登録されている場合、StatusはですQUEUED。ステータスは、IN_PROGRESSジョブの処理が開始されるとに変わり、FAILED処理が終了すると、COMPLETEDまたはに変わります。

リソースのタグ付け

タグは、リソースに追加して識別、整理、検索を容易にするための、カスタムなメタデータラベルです。各タグは、2 つの部分 (タグキーとタグ値) で構成されます。これは、キーと値のペアと呼ばれます。

通常、タグキー はカテゴリの大枠を表し、タグ値 はそのカテゴリのサブセットを表します。例えば、タグキー = Color そしてタグ値 = Blueとすると、キーと値のペアとしては `Color:Blue` が構成されます。タグの値を空の文字列に設定することはできますが、タグの値を null には設定できないことに注意してください。タグ値を省略すると、空の文字列を使用した場合と同じになります。

> **Tip**
>
> AWS Billing and Cost Management では、、請求書を動的なカテゴリで分割するためにタグを利用できます。例えば、会社内のさまざまな部門を表すタグ (`Department:Sales` または `Department:Legal` など) を追加することで、AWS が部門ごとのコスト配分を提供できるようになります。

Amazon Transcribe では、以下のリソースにタグを付けることができます。

- 変換ジョブ
- 医学会話変換ジョブ
- 通話分析通話後の文字起こし求人
- カスタム語彙
- カスタムな医療ボキャブラリー
- カスタム語彙フィルター
- カスタム言語モデル

タグキーには最大 128 文字、タグ値には最大 256 文字を使用でき、どちらも大文字と小文字が区別されます。 Amazon Transcribeリソースごとに最大 50 個のタグがサポートされます。リソースごとに、各タグキーは一意である必要があり、1 つの値のみを与えることができます。注意 : ユーザーは、タグを `aws:` で始めることはできません。このプレフィックスは、AWS によりシステム生成タグ用に予約されています。タグを追加、変更、削除することはできません。`aws:*`このタグは、tags-per-resource 上限タグの数のためのカウントに含まれません。

> リソースのタグ付けに固有の API オペレーション
>
> ListTagsForResource, TagResource, UntagResource
> タグ付け API を使用するには、Amazon リソースネーム (ARN) をリクエストに含める必要があります。ARN は arn:partition:service:region:account-id:resource-type/resource-id という形式です。たとえば、変換ジョブに関連付けられた ARN は arn:*aws*:transcribe:*us-west-2*:*111122223333*:transcription-job/*my-transcription-job-name* のようになります。

タグ付けのベストプラクティスに関するその他情報については、「AWS リソースのタグ付け」を参照してください。

タグベースのアクセスコントロール

タグを使用して、内部のアクセスを制御することができますAWS アカウント。タグベースのアクセス制御に対して、IAMポリシーの条件要素でタグ情報を提供します。次に、タグおよび関連付けられたタグ条件キーを使用して、以下へのアクセスを制御できます。

- リソース:リソースへのアクセスは、Amazon Transcribeリソースに割り当てたタグに基づいて制御します。
  - aws:ResourceTag/*key-name* 条件キーを使用して、リソースにアタッチする必要があるタグキーの値のペアを指定します。
- リクエスト: リクエストで渡すことができるタグを制御します。
  - aws:RequestTag/*key-name*条件キーを使用して、IAMユーザーまたはロールで追加、変更、または削除できるタグを指定します。
- 認証プロセス: 認証プロセスのどの部分でも、タグベースのアクセスを制御できます。
  - aws:TagKeys/ 条件キーを使用して、リソースまたはリクエストで、またはプリンシパルによって特定のタグキーを使用できるかどうかを制御します。この場合、キーバリューは重要ではありません。

タグベースのアクセスコントロールポリシーの例については、「」を参照してくださいタグに基づく字起こしジョブの表示。

タグベースのアクセス制御の詳細については、「タグを使用した AWS リソースへのアクセスの制御」を参照してください。

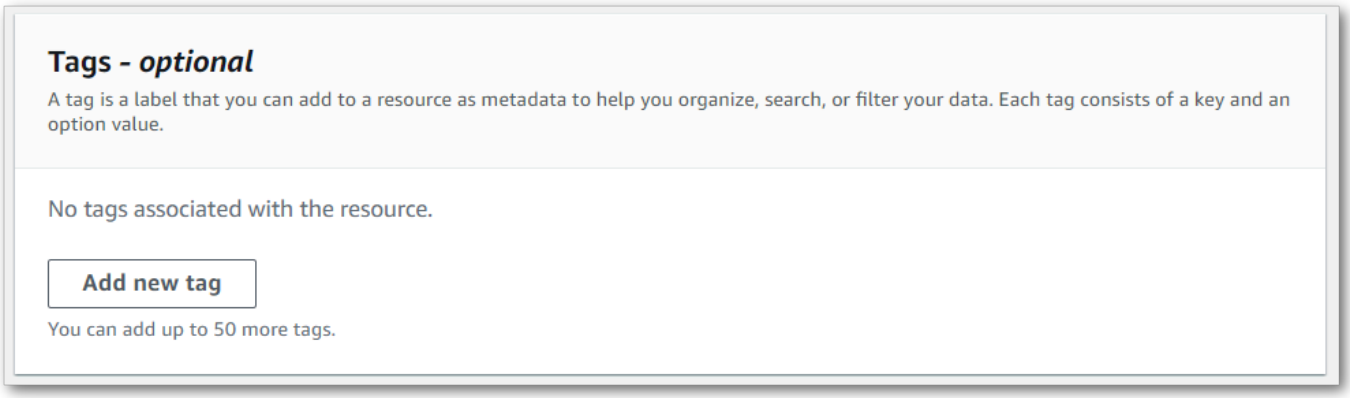
Amazon Transcribeリソースにタグを追加する

Amazon Transcribeジョブの実行前または実行後にタグを追加できます。既存の Create* および Start* API を使用して、文字起こしリクエストにタグを追加できます。

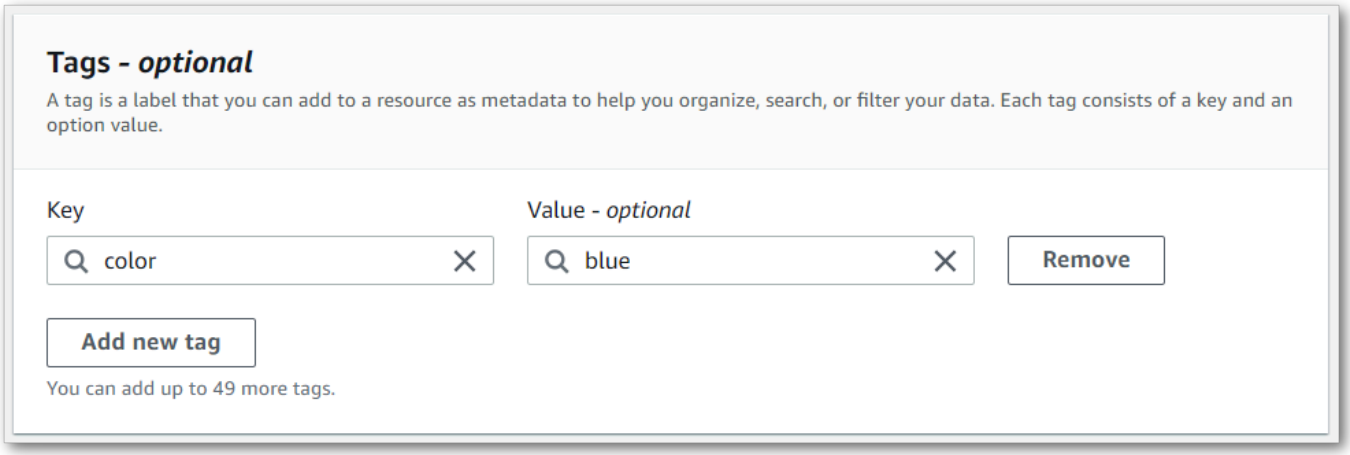
、、または AWSSDK を使用してタグを追加AWS Management Console、変更AWS CLI、または削除できます。例については、以下を参照してください。

AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、変換ジョブを選択後、ジョブの作成 (右上) を選択します。これにより、ジョブの詳細を指定 ページが開きます。
3. [ジョブ詳細の指定] ページの一番下までスクロールして [タグ-オプション] ボックスを見つけ、[新しいタグを追加] を選択します。



4. キー フィールドと、オプションで 値 フィールドに情報を入力します。



5. ジョブの詳細を指定ページで追加したい他のフィールドに入力後、次を選択します。これにより、ジョブの設定 : オプション ページ へ移動します。

   ジョブの作成を選択して、変換ジョブを実行します。

6. 変換ジョブ ページに移動して変換ジョブに関連付けられたタグを表示できますので、変換ジョブを選択し、そのジョブの情報ページの一番下までスクロールします。タグを編集したい場合は、タグの管理を選択すると編集できます。

AWS CLI

この例では、start-transcription-jobTagsコマンドとパラメータを使用しています。詳細については、StartTranscriptionJobおよびTagを参照してください。

```
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
```

別の例として、start-transcription-jobコマンド、およびそのジョブにタグを追加するリクエスト本文を使用します。

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

ファイル my-first-tagging-job.json に次のリクエストボディが入ります。

```
{
  "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)

次の例では、transstart_transcription_job メソッドの AWS SDK for Python (Boto3) 引数で、Tags を使用してタグを追加します。詳細については、StartTranscriptionJobおよびTagを参照してください。

機能固有、シナリオ、サービス間の例など、AWS SDK を使用するその他の例については、SDK を使用した Amazon Transcribe のコード例 AWS SDKsこの章を参照してください。

```
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)
```

スピーカーをパーティション化する (ダイアライゼーション)

話者ダイアライゼーションを使用すると、文字起こし出力で異なる話者を区別できます。 Amazon Transcribe は最大 30 人の一意の話者を区別し、各一意の話者からのテキストに一意の値 (spk_0 から) でラベルを付けることができますspk_9。

スピーカーパーティショニングが有効になっているリクエストには、標準の文字起こしセクション (transcriptsとitems) に加えて、speaker_labels セクションが含まれます。このセクションはスピーカーごとにグループ化されており、話者ラベルやタイムスタンプなど、各発話に関する情報が含まれています。

```
"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"
                },
```

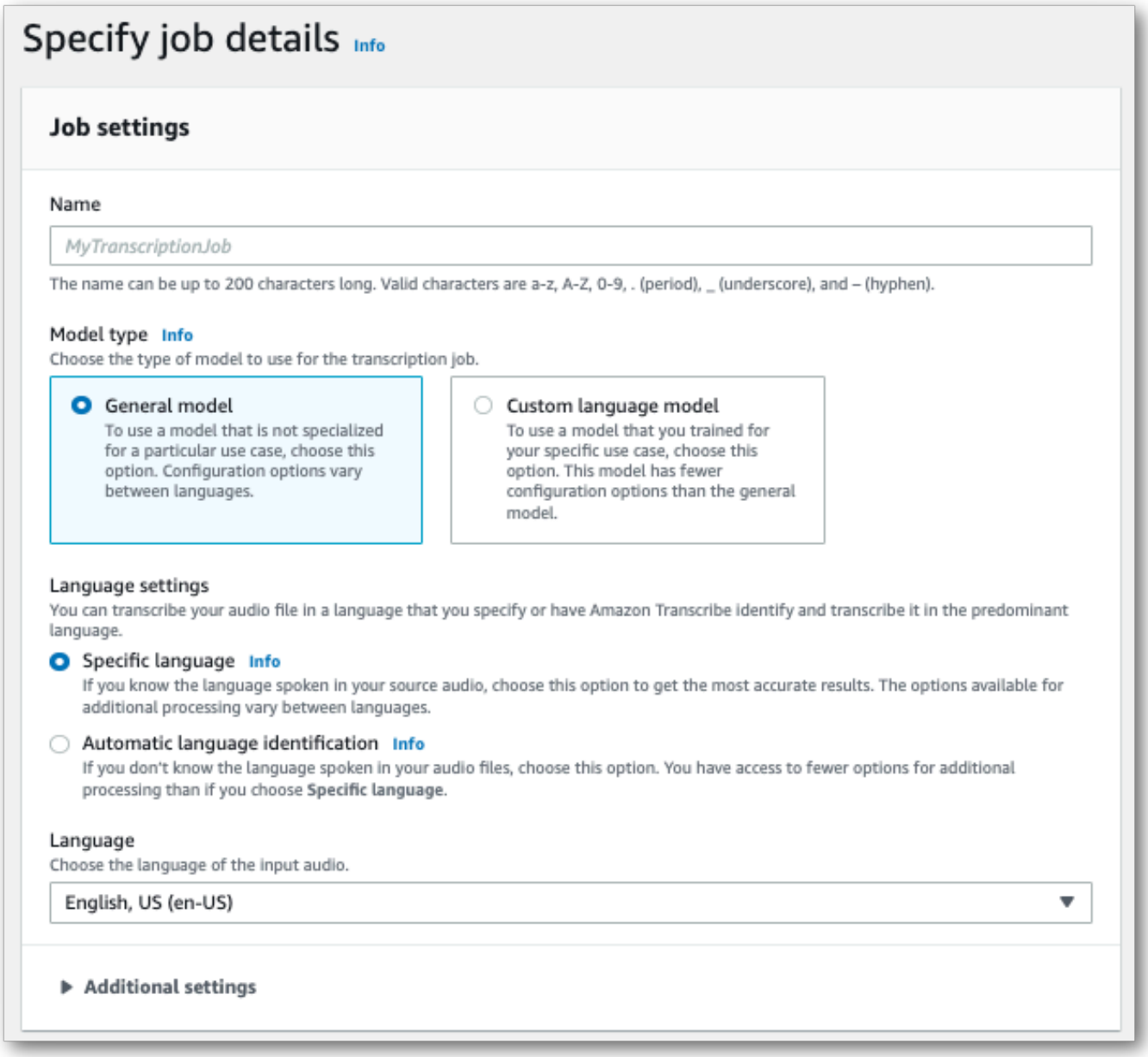
スピーカーパーティショニングを使用した完全な文字起こしの例 (2 人の話者の場合) を見るには、「ダイアライゼーション出力例 (バッチ)」を参照してください。

バッチ文字起こしで、スピーカーをパーティション化する

バッチ文字起こしでスピーカーをパーティション化する方法については、以下の例を参照してください。

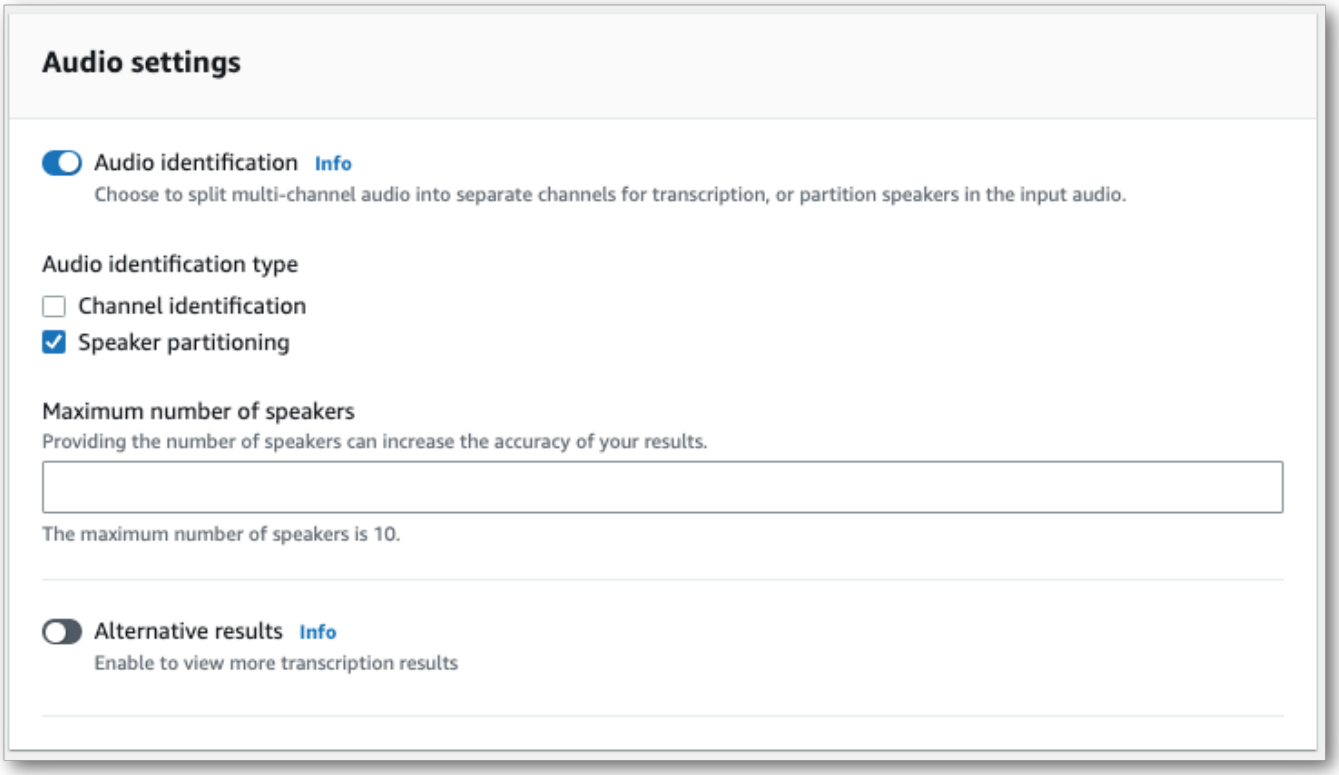
AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、[文字起こしジョブ] を選択後、[ジョブの作成] (右上) を選択します。これにより、「ジョブの詳細を指定」ページが開きます。



3. ジョブの詳細を指定ページで追加したいフィールドに入力後、「次へ」を選択します。これにより、ジョブの設定 - オプションページへ移動します。

   音声設定パネルで、(「音声識別タイプ」の見出しの下にある) [スピーカーパーティショニング] を選択します。オプションで、パーティション化するスピーカーの数を最大 10 個まで指定できます。



4. [ジョブの作成] を選択して、文字起こしジョブを実行します。

AWS CLI

この例では start-transcription-job を使用します。詳細については、「StartTranscriptionJob」を参照してください。

```
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
```

コマンドとstart-transcription-job、そのジョブでスピーカーパーティショニングを有効にするリクエスト本文を使用する別の例を次に示します。

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

ファイル my-first-transcription-job.json には、次のリクエスト本文が含まれています。

```
{
  "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)

この例では AWS SDK for Python (Boto3) 、 を使用して start_transcription_job メソッドを使用してチャネルを識別します。詳細については、「」を参照してください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/',
```

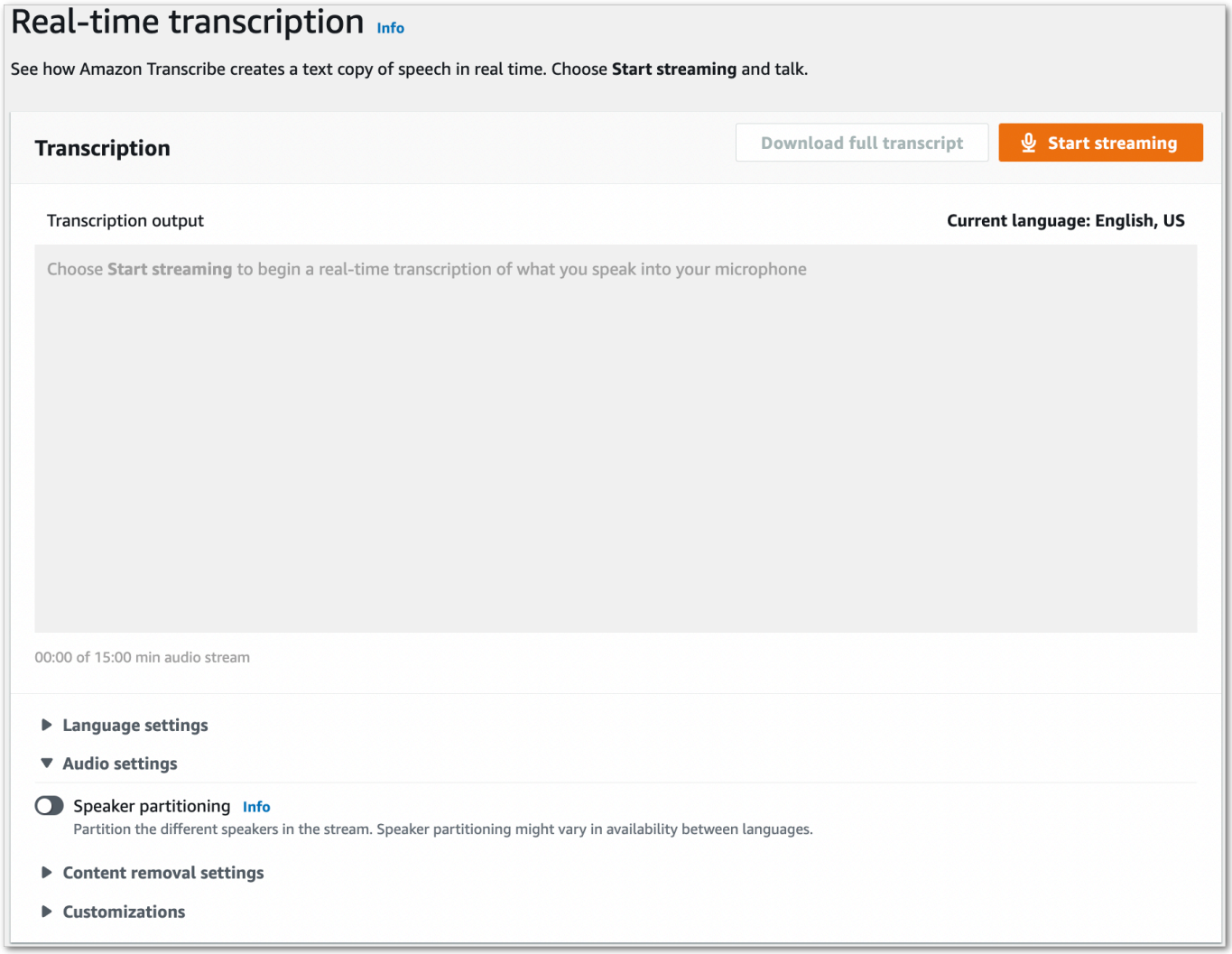
```
    LanguageCode = 'en-US',
    Settings = {
        'ShowSpeakerLabels': True,
        'MaxSpeakerLabels': 3
    }
)

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)
```

ストリーミング文字起こしでスピーカーをパーティション化する

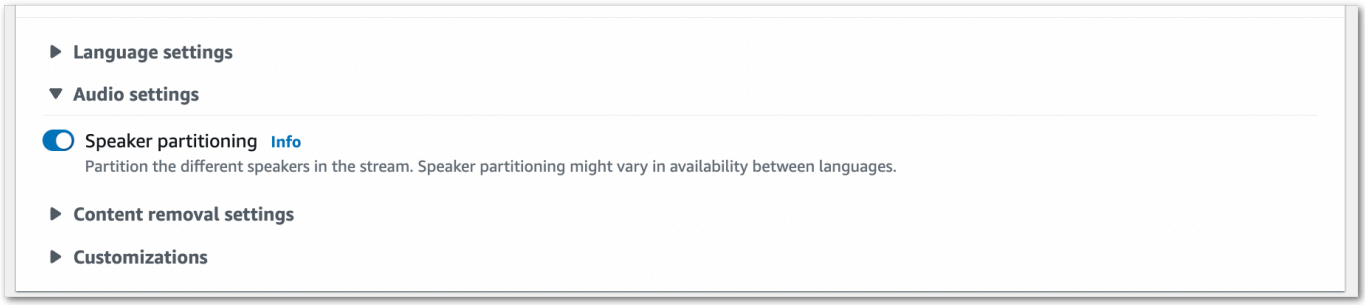
ストリーミング文字起こしでスピーカーをパーティション化するには、次の例を参照してください。

ストリーミング文字起こし

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、[リアルタイム文字起こし] を選択します。音声設定にスクロールして、最小化されている場合はこのフィールドを展開します。



3. スピーカーパーティショニングをオンに切り替えます。



4. これで、ストリームを書き起こす準備ができました。[ストリーミングを開始する] を選択し、話し始めます。ディクテーションを終了するには、[ストリーミングを停止する] を選択します。

HTTP/2 ストリーム

この例では、文字起こし出力でスピーカーをパーティショニングする HTTP/2 リクエストを作成します。での HTTP/2 ストリーミングの使用の詳細については Amazon Transcribe、「」を参照してくださいHTTP/2 ストリームの設定。に固有のパラメータとヘッダーの詳細については、 Amazon Transcribe「」を参照してください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
```

パラメータ定義は API リファレンス にあります。すべての API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。 AWS

WebSocket ストリーム

この例では、文字起こし出力でスピーカーを区切る署名付き URL を作成します。読みやすくするために、改行が追加されています。で WebSocket ストリームを使用する方法の詳細については、Amazon Transcribe「」を参照してください WebSocket ストリームのセットアップ。パラメータの詳細については、「StartStreamTranscription」を参照してください。

```
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
```

パラメータ定義は API リファレンス にあります。すべての API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。 AWS

ダイアライゼーション出力例 (バッチ)

以下は、ダイアライゼーションを有効にしたバッチトランスクリプションの出力例です。

```
{
    "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"
}
```

マルチチャネルの音声文字起こし

オーディオに2つのチャネルがある場合は、チャネル識別で各チャンネルの音声を文字起こしすることができます。 Amazon Transcribe現在、3 チャンネル以上のオーディオはサポートされていません。

トランスクリプトでは、ch_0チャンネルにはラベルとが割り当てられますch_1。

標準のトランスクリプトセクション（transcriptsとitems）に加えて、channel_labelsチャネル識別が有効になっているリクエストにはセクションが含まれます。このセクションには、チャンネルごとにグループ化された各発話または句読点、および関連するチャンネルラベル、タイムスタンプ、および信頼スコアが含まれています。

```
"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
    },
```

あるチャンネルの人が別のチャンネルで話しかけると、各チャンネルの人が重なり、個人が互いに話しかけ合っています。

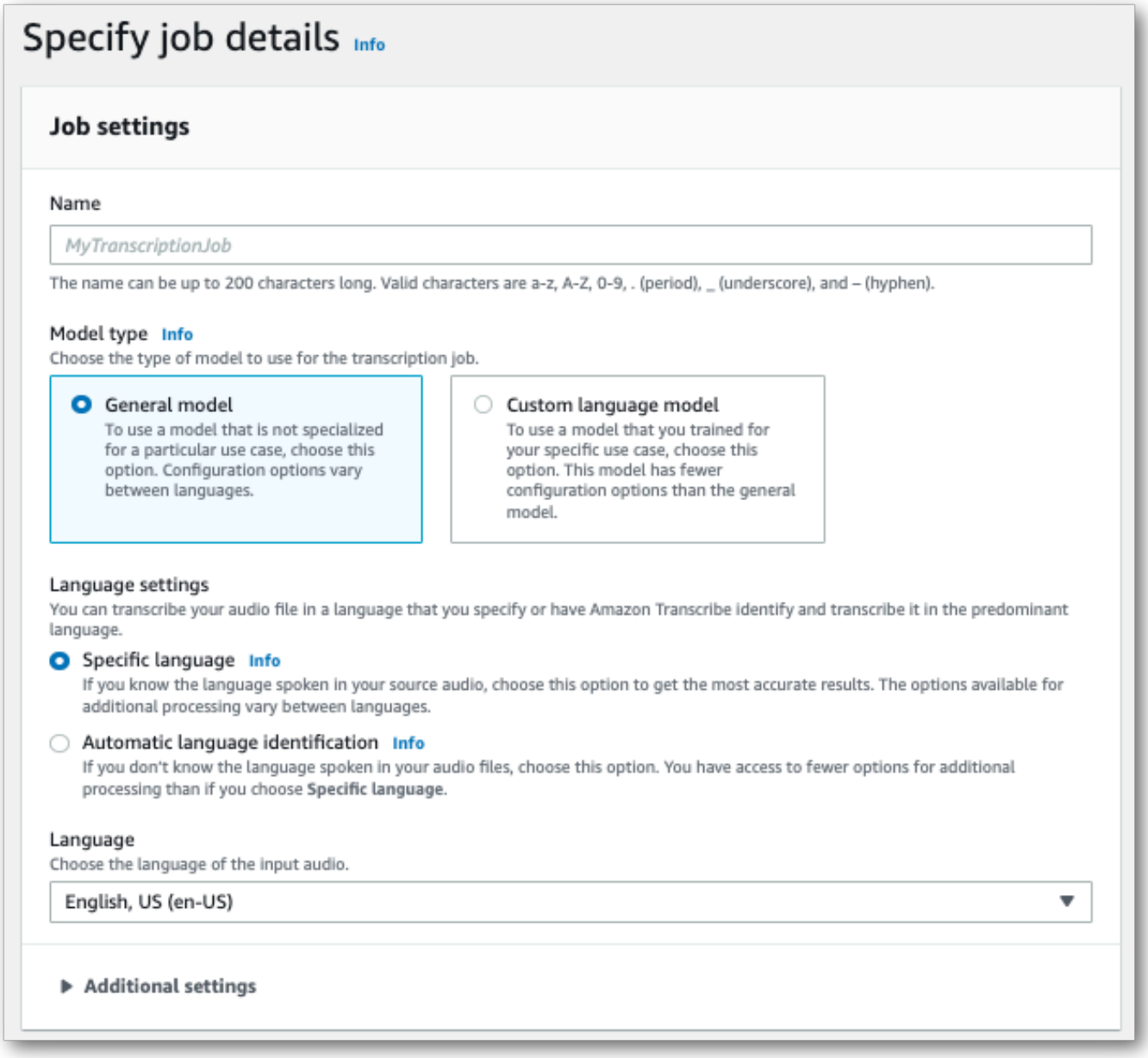
チャンネル識別を含む詳細なトランスクリプト例については、を参照してくださいチャンネル識別出力の例 (バッチ)。

バッチトランスクリプションでのチャンネル識別の使用

バッチトランスクリプション内のチャンネルを識別するには、AWS Management ConsoleAWS CLI、または AWSSDK を使用できます。例については、以下を参照してください。

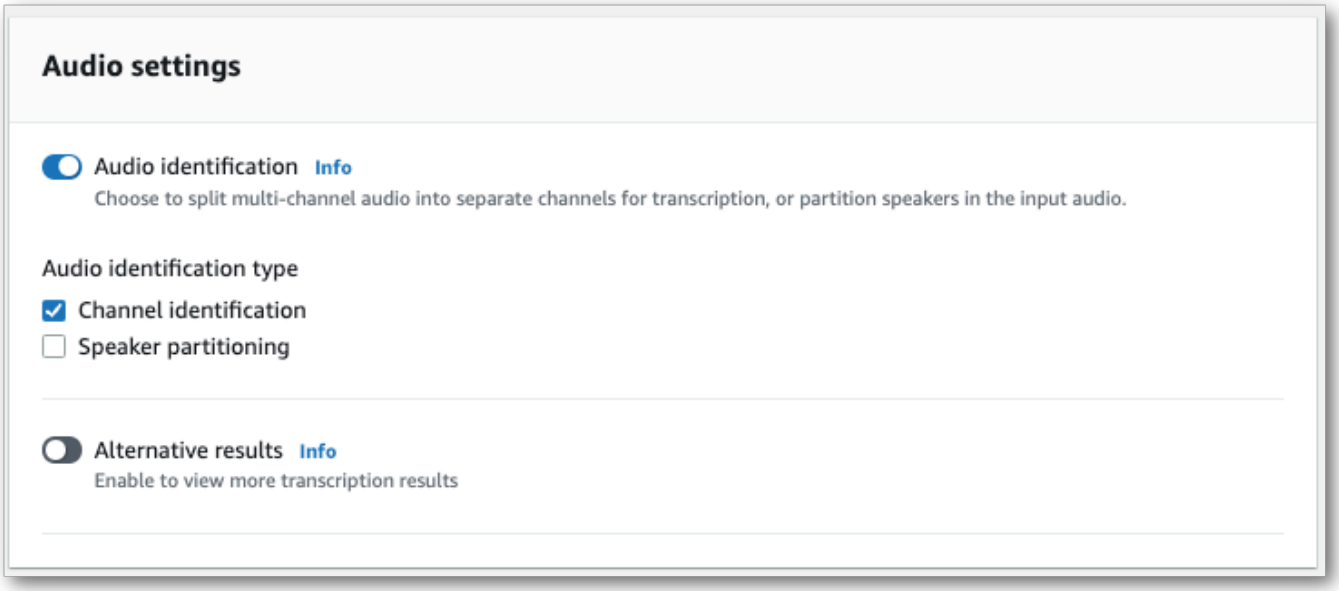
AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、変換ジョブを選択後、ジョブの作成 (右上) を選択します。これにより、ジョブの詳細を指定 ページが開きます。



3. [ジョブの詳細の指定] ページに含めるフィールドを入力し、[次へ] を選択します。これにより、ジョブの設定 : オプション ページ へ移動します。

   オーディオ設定パネルで、チャンネル識別を選択します (「オーディオ識別タイプ」の見出しの下) 。



4. ジョブの作成を選択して、変換ジョブを実行します。

AWS CLI

この例では start-transcription-job を使用します。詳細については、「StartTranscriptionJob」を参照してください。

```
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
```

別の例として、start-transcription-jobコマンド、およびそのジョブでチャネルを識別できるようにするリクエスト本文を使用します。

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

ファイル my-first-transcription-job.json に次のリクエストボディが入ります。

```
{
    "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)

この例では、AWS SDK for Python (Boto3)を使用して start_transcription_job メソッドを使用してチャンネルを識別しています。詳細については、を参照してください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)
```

ストリーミングトランスクリプションでのチャンネル識別の使用

ストリーミング文字変換のチャンネルを識別するには、HTTP/2 またはを使用できますWebSockets。例として以下を参照してください。

HTTP/2 ストリーミング

この例では、音声文字起こし出力のチャネルを分けると、音声文字起こし出力の文字起こし出力のチャネルを分離します。で HTTP/2 ストリーミングを使用する際の詳細についてはAmazon Transcribe、を参照してくださいHTTP/2 ストリームの設定。に固有のパラメータとヘッダーの詳細についてはAmazon Transcribe、を参照してください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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

WebSocket ストリーム

この例では、文字起こし出力のチャンネルを区切る署名付き URL を作成します。読みやすくするために、改行が追加されています。 WebSocket でのストリームの使用の詳細についてはAmazon Transcribe、を参照してください WebSocket ストリームのセットアップ。パラメータの詳細については、「StartStreamTranscription」を参照してください。

```
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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

チャンネル識別出力の例 (バッチ)

チャンネル識別を有効にしたバッチ文字起こしの出力例を次に示します。

```
{
    "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"
}
```

メディア内の主要な言語を特定する

Amazon Transcribe では、言語コードを指定しなくても、メディアで話される言語を自動的に識別できます。

バッチ言語識別では、メディアファイルで話されている主要な言語を識別できます。また、メディアに複数の言語が含まれている場合は、話されているすべての言語を識別できます。言語識別の精度を向上させるために、メディアに含まれていると思われる 2 つ以上の言語のリストをオプションで提供できます。

ストリーミング言語識別は、チャネルごとに 1 つの言語を識別できます (最大 2 つのチャネルがサポートされています)。または、ストリームに複数の言語が含まれている場合は、使用されているすべての言語を識別できます。ストリーミングリクエストには、最低 2 つの追加言語オプションをリクエスト内に含める必要があります。言語オプションを提供すると、言語をすばやく識別できます。Amazon Transcribe が言語を識別するのが早ければ早いほど、ストリームの最初の数秒のデータが失われる可能性は低くなります。

> **Important**
>
> バッチ文字起こしとストリーミング文字起こしは、異なる言語をサポートしています。詳細については、サポートされている言語の表の「データ入力」列を参照してください。現在、ベトナム語とスウェーデン語は言語識別ではサポートされていません。

言語識別によるモニタリングとイベントの詳細については、「言語識別イベント」を参照してください。

バッチ文字起こしジョブを使用した言語の識別

バッチ言語識別を使用して、メディアファイル内の 1 つまたは複数の言語を自動的に識別します。

メディアに 1 つの言語しか含まれていない場合は、単一言語識別を有効にできます。これにより、メディアファイルで使用されている主要言語が識別され、その言語のみを使用してトランスクリプトが作成されます。

メディアに複数の言語が含まれている場合、多言語識別を有効にできます。これにより、メディアファイルで使用されているすべての言語を識別し、識別された各言語を使用してトランスクリプトを

作成できます。多言語のトランスクリプトが作成されることに注意してください。などの他の サービスを使用して Amazon Translate、トランスクリプトを翻訳できます。

サポートされている言語と関連する言語コードの完全なリストについては、「サポートされている言語」の表を参照してください。

最良の結果を得るには、メディアファイルに少なくとも 30 秒の音声が入っていることを確認します。

AWS Management Console、、および AWS Python SDK の使用例については AWS CLI、「」を参照してくださいバッチ文字起こしによる言語識別の使用。

多言語音声の言語識別

多言語識別は多言語のメディアファイルを対象としており、メディア内のすべてのサポートされている言語を反映したトランスクリプトを提供します。つまり、会話の途中でスピーカーが言語を変更したり、各参加者が異なる言語を話したりしても、文字起こし出力は各言語を正しく検出して文字起こしします。たとえば、メディアに米国英語 (en-US) とヒンディー語 (hi-IN) を交互に話すバイリンガルのスピーカーがいる場合、多言語識別により、米国英語を `en-US` とし、ヒンディー語を `hi-IN` として識別して文字起こしすることができます。

これは、1 つの主要言語だけ使ってトランスクリプトを作成する単一言語識別とは異なります。この場合、主要言語ではない話し言葉はすべて正しく文字起こしされません。

> **Note**
>
> 現在、リダクションとカスタム言語モデルは多言語識別ではサポートされていません。

> **Note**
>
> 現在、多言語識別では、en-AB、en-AU、en-GB、en-IE、en-IN、en-NZ、en-US、en-WL、en-ZA、es-ES、es-US、fr-CA、fr-FR、zh-CN、zh-TW、pt-BR、pt-PT、de-CH、de-DE、af-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、th-IN、th-TR-TR がサポートされています。

多言語のトランスクリプトには、検出された言語の概要と、メディア内で各言語が話された合計時間が表示されます。例を示します。

```
"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
            }
        ]
}
```

言語識別の精度の向上

言語識別では、メディアに存在すると思われる言語のリスト含めるオプションがあります。言語オプション (LanguageOptions) を含めると Amazon Transcribe 、音声を正しい言語と一致させるときに指定した言語のみを使用するように制限されます。これにより、言語識別が高速化され、正しい言語ダイアレクトの割り当てに関連する精度が向上します。

言語コードを含める場合は、少なくとも 2 つ含める必要があります。含められる言語コードの数に制限はありませんが、効率と精度を最適化するために、2～5 つの言語コードを使用することをおすすめします。

> **Note**
>
> リクエストに言語コードを含め、指定した言語コードが音声で識別される言語と一致しない場合、 Amazon Transcribe は指定した言語コードから最も近い言語を選択します。次

に、その言語の文字起こしが作成されます。例えば、メディアが米国英語 (en-US) で、Amazon Transcribe 言語コード zh-CN、、および を指定した場合de-DE、メディア Amazon Transcribe をドイツ語 (de-DE) と照合しfr-FR、ドイツ語の文字起こしを生成する可能性があります。言語コードと話し言葉が一致しないと、文字起こしが不正確になる可能性があるため、言語コードを含める際には注意が必要です。

言語識別と他の Amazon Transcribe 機能の組み合わせ

バッチ言語識別は、他の Amazon Transcribe 機能と組み合わせて使用できます。言語識別を他の機能と組み合わせる場合、それらの機能でサポートされている言語に制限されます。例えば、コンテンツ編集で言語識別を使用する場合、秘匿化に使用できる言語は米国英語 (en-US) または米国スペイン語 (es-US) に制限されます。詳細については、「サポートされている言語および言語固有の機能」を参照してください。

> **Important**
>
> コンテンツ編集を有効にして自動言語識別を使用し、音声に米国英語 (en-US) または米国スペイン語 () 以外の言語が含まれている場合es-US、トランスクリプトでは米国英語または米国スペイン語のコンテンツのみが編集されます。他の言語は編集できず、警告やジョブの失敗のお知らせは表示されません。

カスタム言語モデル、カスタム語彙、カスタム語彙フィルター

言語識別リクエストに 1 つ以上のカスタム言語モデル、カスタム語彙、またはカスタム語彙フィルターを追加する場合は、LanguageIdSettings パラメータを含める必要があります。次に、対応するカスタム言語モデル、カスタム語彙、カスタム語彙フィルターを使用して言語コードを指定できます。多言語識別はカスタム言語モデルをサポートしていないことに注意してください。

LanguageIdSettings を使用する際には、正しい言語の方言が確実に識別されるように、LanguageOptions を含めることをおすすめします。例えば、en-USカスタム語彙を指定しても、メディアで話されている言語が Amazon Transcribe であると判断した場合en-AU、カスタム語彙は文字起こしに適用されません。LanguageOptions を含め、en-US を英語の方言のみとして指定すると、カスタム語彙が文字起こしに適用されます。

リクエストの `LanguageIdSettings` の例については、バッチ文字起こしによる言語識別の使用 セクションの AWS CLI および AWS SDK ドロップダウンパネルの「オプション 2」参照してください。

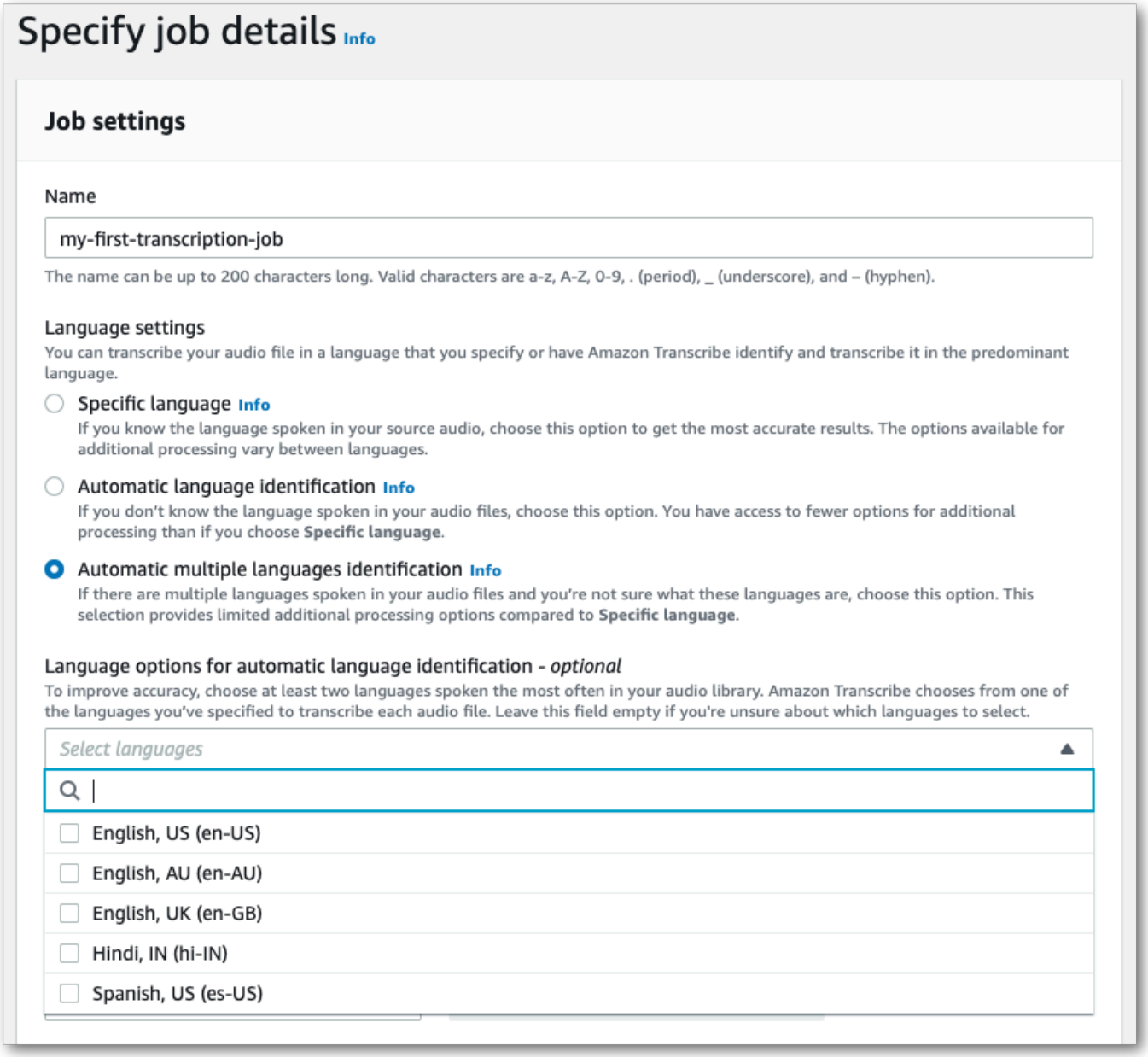
バッチ文字起こしによる言語識別の使用

バッチ文字起こしジョブで、AWS Management Console、AWS CLI、または AWS SDK を使用して、自動言語識別を使用できます。例については、次を参照してください。

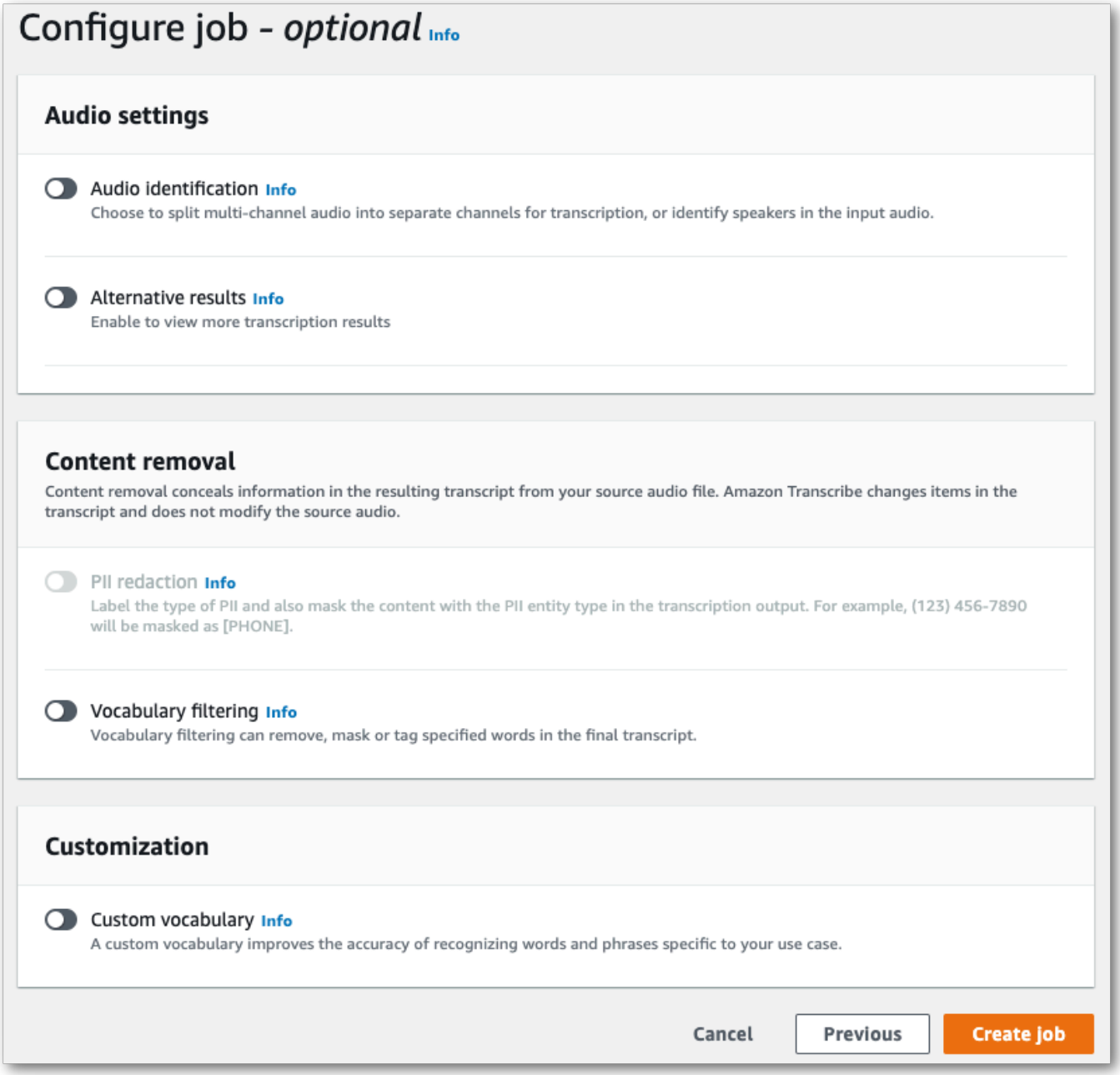
AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、[文字起こしジョブ] を選択後、[ジョブの作成] (右上) を選択します。これにより、「ジョブの詳細を指定」ページが開きます。
3. ジョブの設定パネルで、言語設定セクションを見つけ、[自動言語識別] または [自動多言語識別] を選択します。

   音声ファイルに含まれる言語がわかっている場合は、(「言語を選択」ドロップダウンボックスから) 複数の言語オプションを選択できます。言語オプションを提供することで、精度を向上させることができますが、必須ではありません。



4. ジョブの詳細を指定ページに追加したいその他のフィールドに入力し、「次へ」を選択します。これにより、ジョブの設定 - オプションページへ移動します。



5. [ジョブの作成] を選択して、文字起こしジョブを実行します。

AWS CLI

この例では、start-transcription-job コマンドと IdentifyLanguageパラメータを使用します。詳細については、「StartTranscriptionJob」および「LanguageIdSettings」を参照してください。

オプション 1: language-id-settings パラメータを使用しない場合。リクエストにカスタム言語モデル、カスタム語彙、カスタム語彙フィルターを含めない場合は、このオプションを使用してください。language-options はオプションですが、推奨されます。

```
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"
```

オプション 2: language-id-settings パラメータを使用する場合。リクエストにカスタム言語モデル、カスタム語彙、カスタム語彙フィルターを含める場合は、このオプションを使用してください。

```
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
```

start-transcription-job コマンドと、言語を識別するリクエスト本文を使用する別の例を次に示します。

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

ファイル my-first-language-id-job.json には、次のリクエスト本文が含まれています。

オプション 1: LanguageIdSettings パラメータを使用しない場合。リクエストにカスタム言語モデル、カスタム語彙、カスタム語彙フィルターを含めない場合は、このオプションを使用してください。LanguageOptions はオプションですが、推奨されます。

```
{
  "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"
  ]
}
```

オプション 2: LanguageIdSettings パラメータを使用する場合。リクエストにカスタム言語モデル、カスタム語彙、カスタム語彙フィルターを含める場合は、このオプションを使用してください。

```
{
   "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)

この例では、 を使用して AWS SDK for Python (Boto3) 、start_transcription_job メソッドの IdentifyLanguage引数を使用してファイルの言語を識別します。詳細については、「StartTranscriptionJob」および「LanguageIdSettings」を参照してください。

機能固有の例、シナリオ例、クロスサービス例など、 AWS SDKsSDK を使用した Amazon Transcribe のコード例 AWS SDKs「」の章を参照してください。

オプション 1: LanguageIdSettings パラメータを使用しない場合。リクエストにカスタム言語モデル、カスタム語彙、カスタム語彙フィルターを含めない場合は、このオプションを使用してください。LanguageOptions はオプションですが、推奨されます。

```
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)
```

オプション 2: LanguageIdSettings パラメータを使用する場合。リクエストにカスタム言語モデル、カスタム語彙、カスタム語彙フィルターを含める場合は、このオプションを使用してください。

```
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)
```

ストリーミング文字起こしによる言語識別

ストリーミング言語識別により、メディアストリームで話されている主要な言語を特定できます。Amazon Transcribe は、言語を識別するために最低 3 秒の音声が必要です。

ストリームに 1 つの言語しか含まれていない場合は、単一言語識別を有効にできます。これにより、メディアファイルで使用されている主要言語が識別され、この言語のみを使用してトランスクリプトが作成されます。

ストリームに複数の言語が含まれている場合は、多言語識別を有効にできます。これにより、ストリームで使用されているすべての言語を識別し、識別された各言語を使用してトランスクリプトを作成できます。多言語のトランスクリプトが作成されることに注意してください。Amazon Transcribe など他のサービスを使って、トランスクリプトを翻訳できます。

ストリーミング言語識別には,少なくとも 2 つの言語コードを指定する必要があります。また、ストリームごとに、1 つの言語に対して 1 つの言語方言しか選択できません。つまり、`en-US` と `en-AU` を同じ文字起こしの言語オプションとして選択することはできません。

また、指定した言語コードの一覧から優先言語を選択するオプションもあります。優先言語を追加すると、言語識別プロセスが速くなるため、短い音声クリップの場合に便利です。

> **Important**
>
> 提供された言語コードのいずれかが音声で識別された言語と一致しない場合、Amazon Transcribe は指定した言語コードから最も近い言語を選択します。次に、その言語の文字起こしが作成されます。例えば、メディアが米国英語 (`en-US`) で Amazon Transcribe に言語コード `zh-CN`、`fr-FR`、または `de-DE` を提供した場合、Amazon Transcribe はメディアをドイツ語 (`de-DE`) にマッチさせ、ドイツ語の文字起こしを作成する可能性があります。言語コードと話し言葉が一致しないと、文字起こしが不正確になる可能性があるため、言語コードを含める際には注意が必要です。

メディアに 2 つのチャネルがある場合は、Amazon Transcribe は、各チャネルで話されている主要な言語を特定できます。この場合は、`ChannelIdentification` パラメータを `true` に設定することで各チャネルが別々に文字起こしされます。このパラメータのデフォルトは `false` です。変更しない場合は、最初のチャネルだけが文字起こしされ、1 つの言語だけが識別されます。

ストリーミング言語識別は、カスタム言語モデルやリダクションと組み合わせることはできません。言語識別を他の機能と組み合わせる場合、それらの機能でサポートされている言語、およびストリー

ミング文字起こしでサポートされている言語に制限されます。「サポートされている言語」を参照してください。

> **Note**
>
> PCM と FLAC は、ストリーミング言語識別でサポートされる唯一の音声形式です。

多言語音声の言語識別

多言語識別は、多言語のストリームを対象としており、ストリーム内のすべてのサポートされている言語を反映したトランスクリプトを提供します。つまり、会話の途中でスピーカーが言語を変更したり、各参加者が異なる言語を話したりしても、文字起こし出力は各言語を正しく検出して文字起こしします。

例えば、ストリームに米国英語 (en-US) とヒンディー語 (hi-IN) を交互に話すバイリンガルの話者がいる場合、多言語識別により、米国英語を en-US とし、ヒンディー語を hi-IN として識別して書き起こすことができます。これは、1 つの主要言語だけ使ってトランスクリプトを作成する単一言語識別とは異なります。この場合、主要言語ではない話し言葉はすべて正しく文字起こしされません。

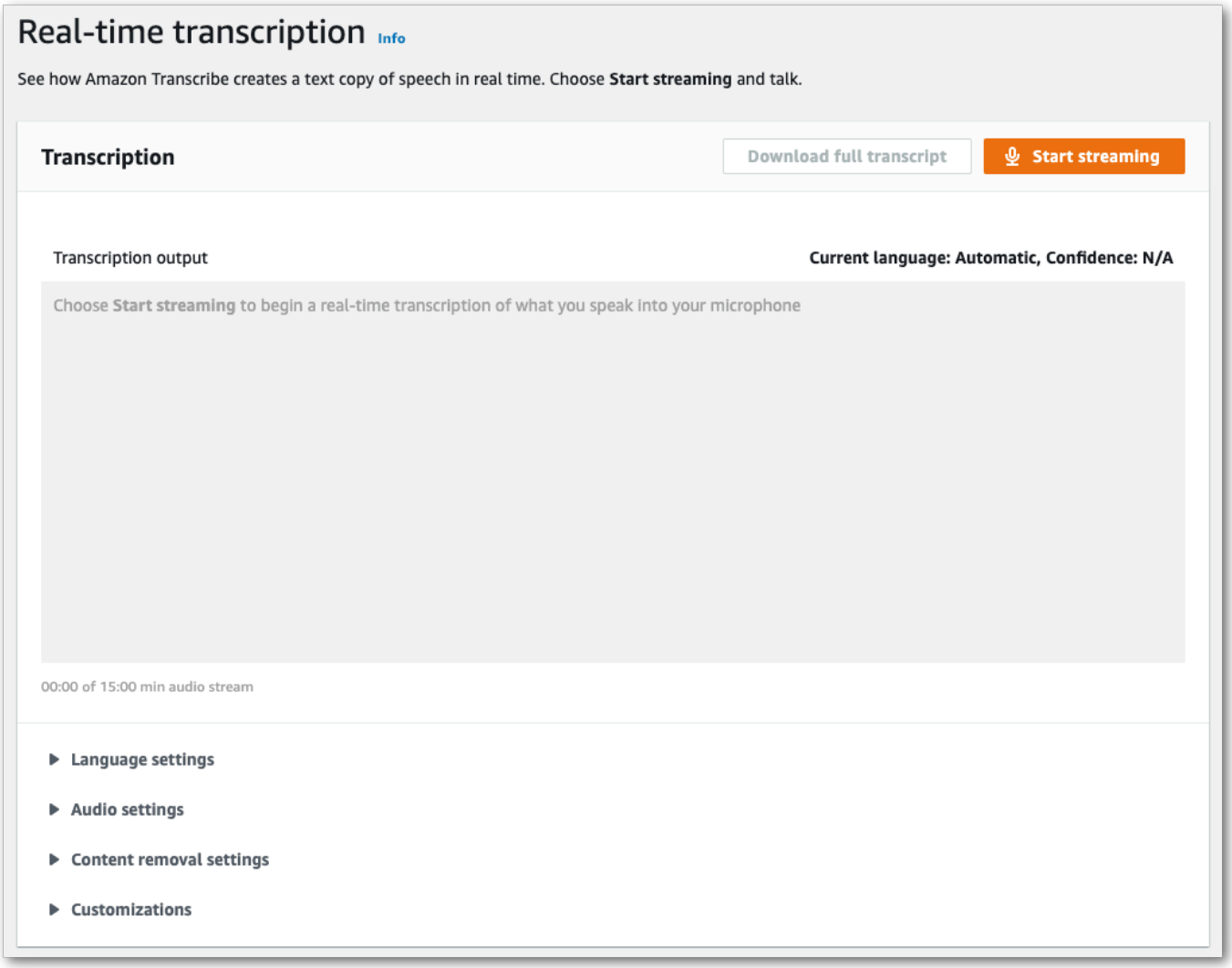
> **Note**
>
> 現在、リダクションとカスタム言語モデルは多言語識別ではサポートされていません。

ストリーミングメディアでの言語識別を使用

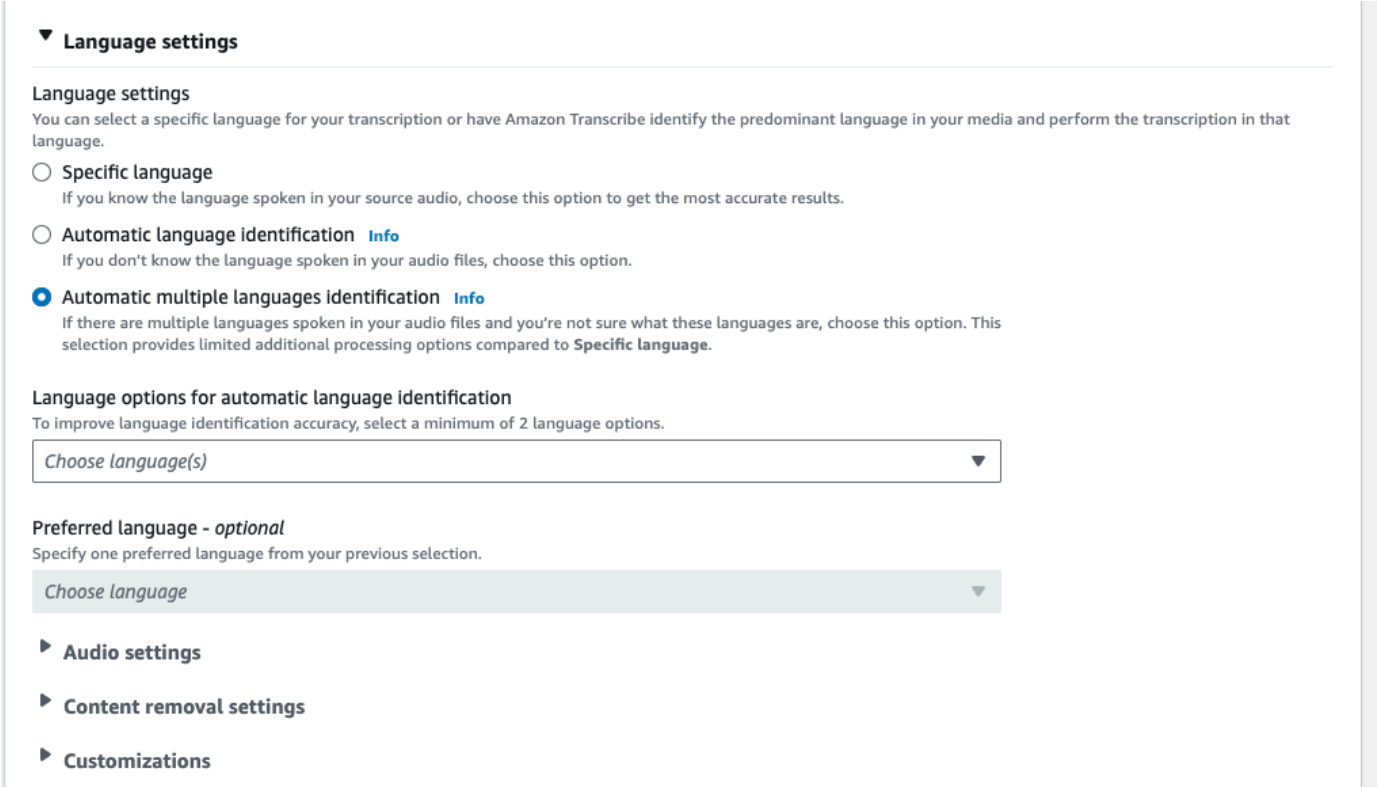
バッチ文字起こしジョブで AWS Management Console、HTTP/2、または WebSocket を使用して、自動言語識別を使用できます: 例については、次を参照してください。

AWS Management Console

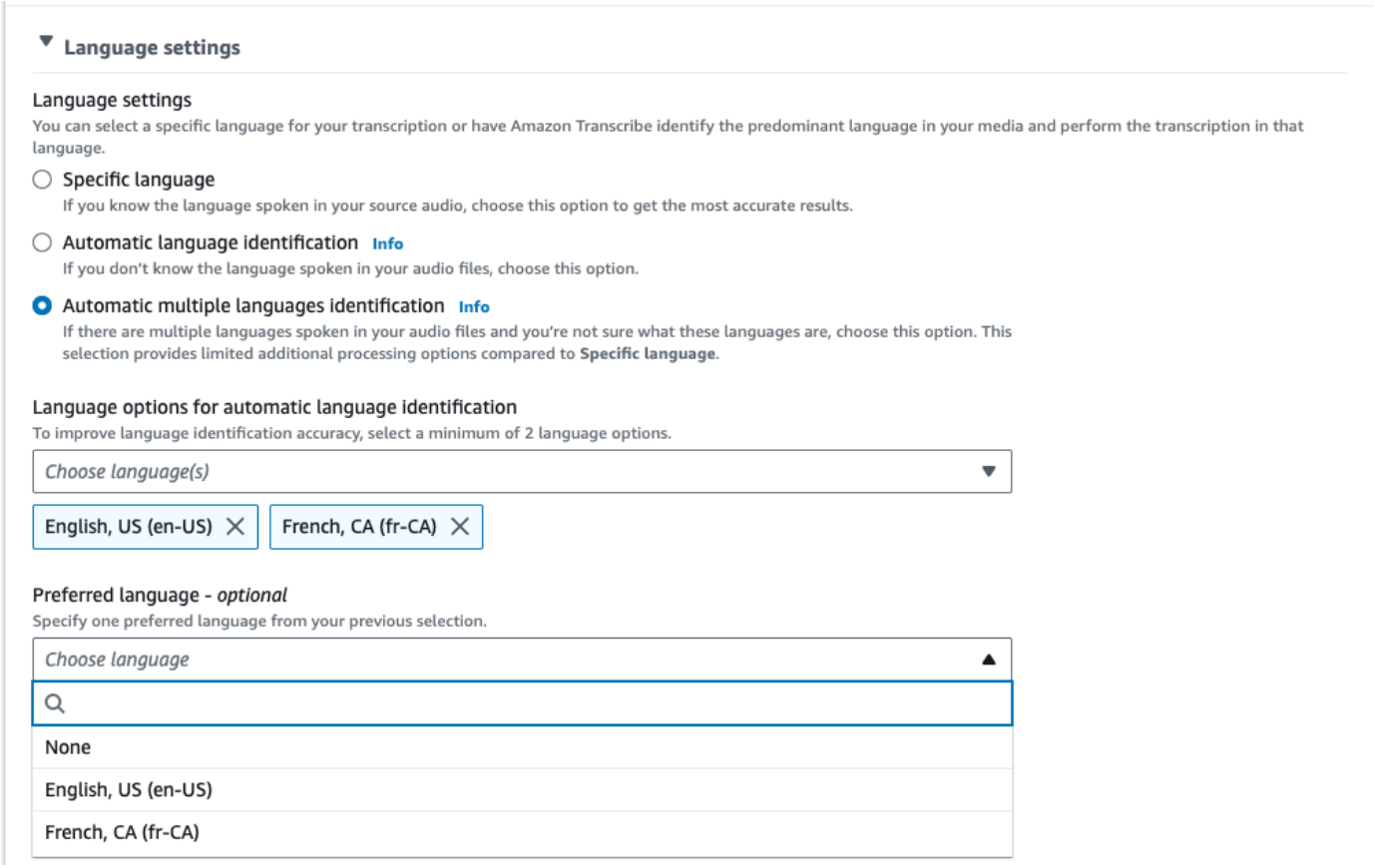
1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、[リアルタイム文字起こし] を選択します。言語設定にスクロールして、最小化されている場合はこのフィールドを展開します。



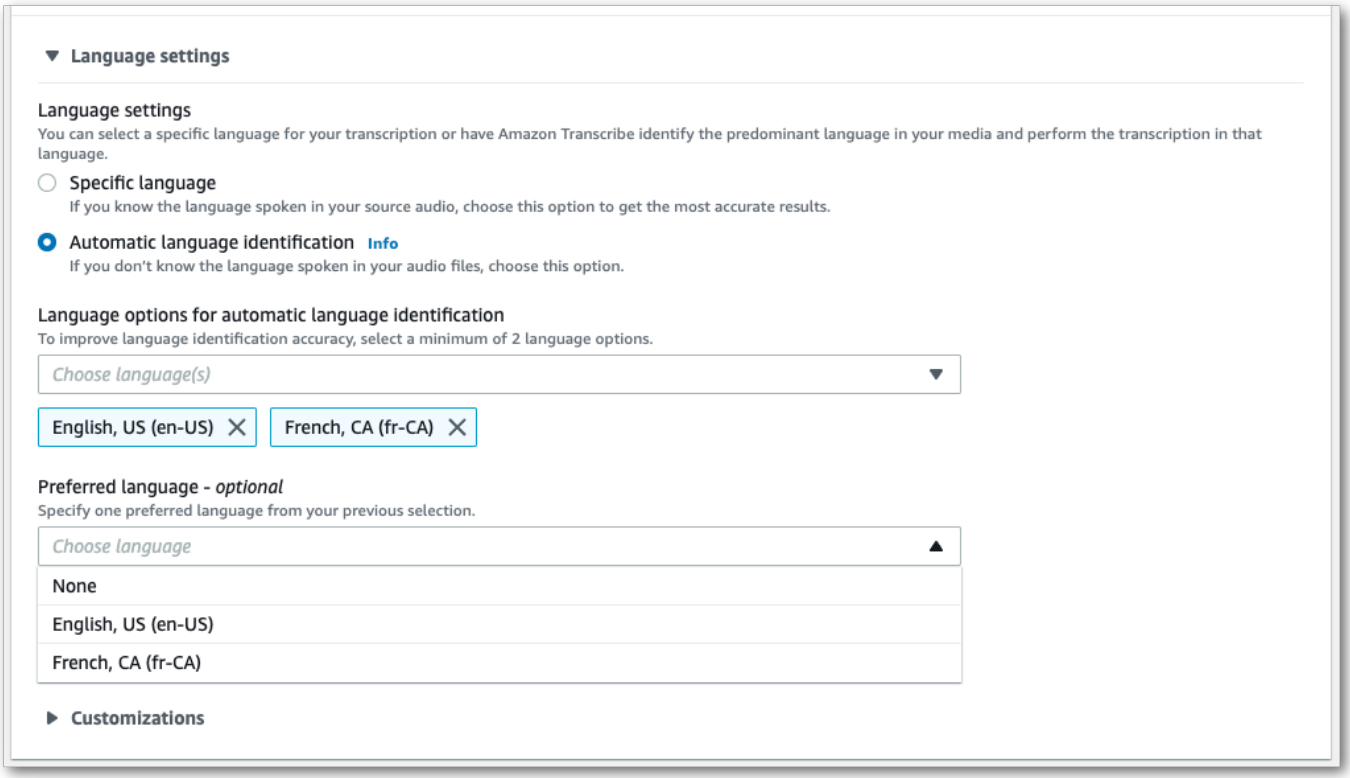
3. [自動言語識別] または [複数言語の自動識別] を選択します。



4. 文字起こしの言語コードを 2 つ以上指定してください。1 つの言語ごとに 1 つの方言しか提供できません。例えば、en-US と fr-CA を同じ文字起こしの言語オプションとして選択することはできません。



5. (オプション) 前の手順で選択した言語のサブセットから、トランスクリプトの優先言語を選択できます。



6. これで、ストリームを書き起こす準備ができました。[ストリーミングを開始する] を選択し、話し始めます。ディクテーションを終了するには、[ストリーミングを停止する] を選択します。

HTTP/2 ストリーム

この例では、言語識別を有効にした状態で HTTP/2 リクエストを作成します。Amazon Transcribe で HTTP/2 ストリーミングを使用する際の詳細については、「HTTP/2 ストリームの設定」を参照してください。「 Amazon Transcribe に固有のパラメータとヘッダーの詳細については、「StartStreamTranscription」を参照してください。

```
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
```

この例では、多言語識別を有効にして HTTP/2 リクエストを作成します。Amazon Transcribe で HTTP/2 ストリーミングを使用する際の詳細については、「HTTP/2 ストリームの設定」を参照してください。Amazon Transcribe に固有のパラメータとヘッダーの詳細については、「StartStreamTranscription」を参照してください。

```
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
```

リクエストで `identify-language` または `identify-multiple-languages` を使用する場合は、`language-options` も含める必要があります。同じリクエストで `language-code` と `identify-language` 両方を使用することはできません。

パラメータの定義は API リファレンスにあり、すべての AWS API オペレーションに共通するパラメータは「共通パラメータ」セクションに記載されています。

WebSocket ストリーム

この例では、WebSocket ストリーミングで言語識別を使用する署名付き URL を作成します。読みやすくするために、改行が追加されています。Amazon Transcribe での WebSocket ストリームの使用の詳細については、「WebSocket ストリームのセットアップ」を参照してください。パラメータの詳細については、「StartStreamTranscription」を参照してください。

```
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
```

この例では、WebSocket ストリームで多言語識別を使用する署名付き URL を作成します。読みやすくするために、改行が追加されています。Amazon Transcribe での WebSocket ストリームの使用の詳細については、「WebSocket ストリームのセットアップ」を参照してください。パラメータの詳細については、「StartStreamTranscription」を参照してください。

```
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
```

リクエストで identify-language または identify-multiple-languages を使用する場合は、language-options も含める必要があります。同じリクエストで language-code と identify-language 両方を使用することはできません。

パラメータの定義は API リファレンスにあり、すべての AWS API オペレーションに共通するパラメータは「共通パラメータ」セクションに記載されています。

代替文字起こし

Amazon Transcribeオーディオを文字起こしすると、同じトランスクリプトの異なるバージョンが作成され、各バージョンに信頼スコアが割り当てられます。一般的な文字起こしでは、信頼度スコアが最も高いバージョンのみが表示されます。

代替文字起こしを有効にすると、Amazon Transcribe信頼度の低い他のバージョンの文字起こしが返されます。代替文字起こは、最大 10 個選択できます。Amazon Transcribe指定されている数よりも多くの選択肢を指定した場合、実際の選択肢の数だけが返されます。

すべての代替案は同じトランスクリプション出力ファイル内にあり、セグメントレベルで表示されます。セグメントは、話し手の交代や音声の一時停止など、話し手の交代や音声の一時停止など、話し手の交代や音声の一時停止です。

代替文字起こは、バッチ文字起こでのみ使用できます。

トランスクリプションの出力は以下のように構成されています。楕円 (`...` コード例の) は、簡潔にするためにコンテンツが削除された場所を示しています。

1. 特定のセグメントの完全な最終文字起こし。

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

2. `transcript`前のセクションの各単語の信頼度スコア。

```
"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. 代替文字起こしは、segments文字起こし出力の一部にあります。各セグメントの選択肢は、信頼スコアが高い順に並べられています。

```
"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. 文字起こし出力の最後にあるステータス。

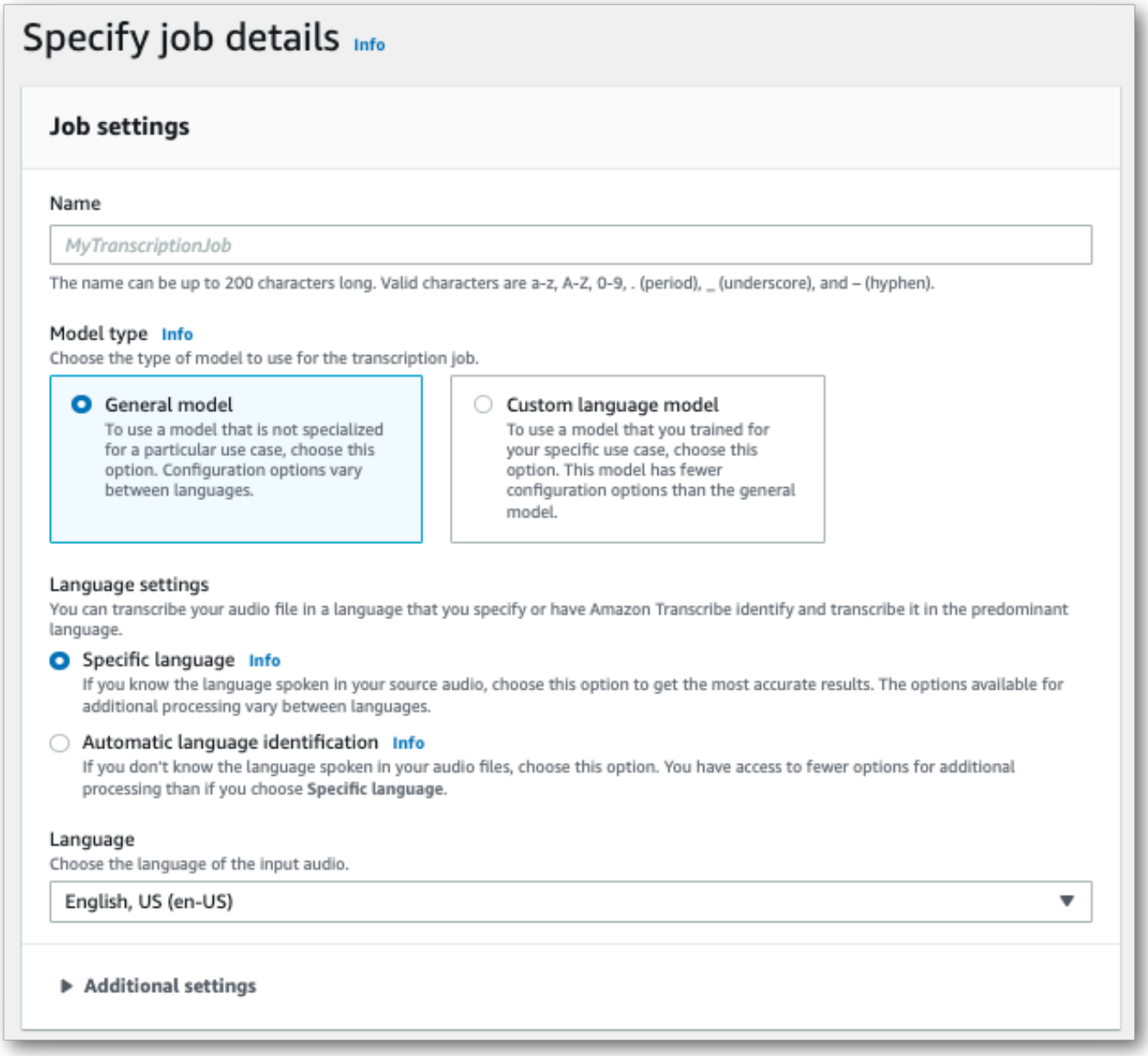
```
"status": "COMPLETED"
}
```

代替文字起こ成

、、または AWSSDK AWS Management Consoleを使用して代替文字変換をリクエストできます。例については、以下を参照してください。AWS CLI

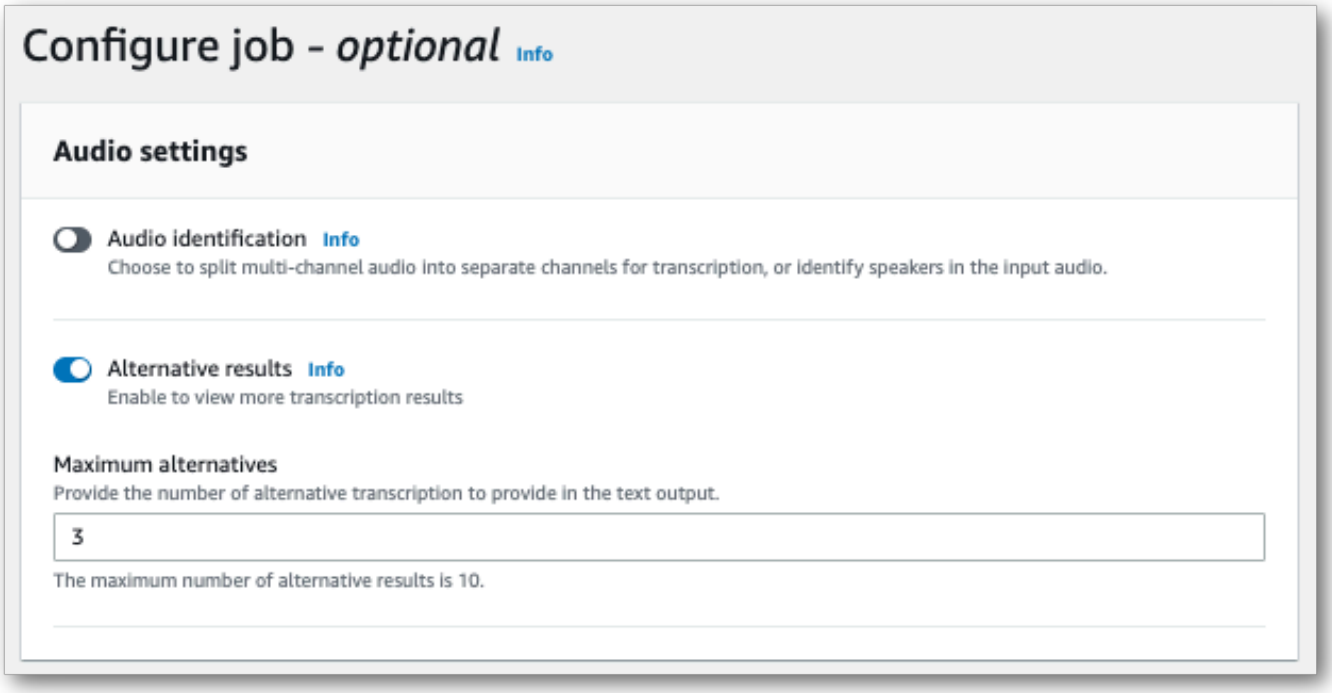
AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペイン で、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、これにより、ジョブの詳細を指定 ページが開きます。



3. [ジョブの詳細の指定] ページに含めるフィールドを入力し、[次へ] を選択します。これにより、ジョブの設定 : オプション ページ へ移動します。

   「代替結果」を選択し、トランスクリプトに含める代替トランスクリプション結果の最大数を指定します。



4. ジョブの作成を選択して、、、、、、、、、、、、、、、、、、

AWS CLI

この例では、start-transcription-jobShowAlternativesコマンドとパラメータを使用しています。詳細については、「StartTranscriptionJob」および「ShowAlternatives」を参照してください。

ShowAlternatives=trueリクエストに含める場合は、必ず含める必要があることに注意してください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
```

ここでは、start-transcription-jobコマンドを使用した別の例として、代替文字起こします。

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

```
--region us-west-2 \
--cli-input-json file://filepath/my-first-alt-transcription-job.json
```

ファイル my-first-alt-transcription-job .json には、次のリクエストボディが含まれます。

```
{
  "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)

次の例では、transstart_transcription_jobShowAlternatives メソッドの引数で、Tstart_transcription_job メソッドの引数で、代替文字起こです。AWS SDK for Python (Boto3)詳細については、「StartTranscriptionJob」および「ShowAlternatives」を参照してください。

機能固有、シナリオ、サービス間の例など、AWS SDK を使用するその他の例については、SDK を使用した Amazon Transcribe のコード例 AWS SDKsこの章を参照してください。

'ShowAlternatives':Trueリクエストに含める場合は、必ず含める必要があることに注意してください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)
```

カスタム語彙をカスタム言語モデルによる文字起こしの精度の向上

メディアにブランド名、略語、専門用語、専門用語、専門用語など、ドメイン固有または非標準の用語が含まれていると、Amazon Transcribeそれらの用語が文字起こし出力に正しく取り込まれない可能性があります。

文字起こしの誤りを修正し、特定のユースケースに合わせて出力をカスタマイズするには、カスタム語彙とを作成しますカスタム言語モデル。

- カスタム語彙あらゆる文脈で特定の単語の認識とフォーマットの両方を調整し、強化するように設計されています。これには、単語と、Amazon Transcribeオプションで発音と表示形式を入力することが含まれます。

  トランスクリプトで特定の用語が正しく表示されない場合はAmazon Transcribe、Amazon Transcribeこれらの用語の表示方法を示すカスタムボキャブラリファイルを作成できます。この単語固有のアプローチは、ブランド名や頭字語などの用語の修正に最も適しています。
- カスタム言語モデル用語に関連する文脈を捉えるように設計されています。これには、Amazon Transcribeドメイン固有のテキストデータを大量に提供することが含まれます。

  専門用語が正しく表示されていない場合や、トランスクリプトに間違った同音異義語を使用している場合はAmazon Transcribe、Amazon Transcribeドメイン固有の言語を教えるカスタム言語モデルを作成できます。たとえば、カスタム言語モデルでは、「フロー」(アイスフロー) と「フロー」(リニアフロー) のどちらを使用するかを学習できます。

  このコンテキスト認識アプローチは、ドメイン固有の音声を大量に文字起こしする場合に最も適しています。カスタム語彙を単独で達成するよりも精度が大幅に向上する可能性があります。バッチトランスクリプションを使用する場合、リクエストにカスタム言語モデルとカスタムボキャブラリーの両方を含めることができます。

> **Tip**
>
> 最高の文字起こしの精度を達成するには、カスタム語彙をカスタム言語モデルと組み合わせて使用します。

を使用してカスタムボキャブラリーを作成する方法のビデオデモについてはAWS Management Console、「カスタムボキャブラリーの使用」を参照してください。

カスタム言語モデルを作成して使用する方法に関するビデオデモについては、「カスタム言語モデル (CLM) を使用して文字起こしの精度を高める」を参照してください。

AWSMachine Learning ブログでさらに詳しく

カスタム語彙をカスタム語彙を設定

- を使用したF1レースのライブトランスクリプションAmazon Transcribe

カスタム言語モデル:

- speech-to-text パフォーマンスを向上させるためのカスタム言語モデルの構築Amazon Transcribe
- カスタム言語モデルにより、授業の講義の文字起こしの精度を高めましょうAmazon Transcribe

カスタム語彙

カスタム語彙を追加して、1 つまたは複数の単語の文字起こし精度を向上させます。これらは通常、ブランド名や頭字語、固有名詞、 Amazon Transcribe が正しく表示されない単語など、ドメイン固有の用語です。

カスタム語彙は、サポートされているすべての言語で使用できます。カスタム語彙で使用できるのは、その言語の文字セットにリストされている文字だけであることに注意してください。

Important

Amazon Transcribeを使用する場合、お客様はご自身のデータの完全性について責任を負うものとします。機密情報、個人情報 (PII)、または保護対象の医療情報 (PHI) をカスタム語彙に入力しないでください。

カスタム語彙を作成する際の考慮事項

- ごとに最大 100 個のカスタム語彙ファイルを含めることができます AWS アカウント

- カスタム語彙のサイズは 50 KB に制限されます。
- API を使用してカスタム語彙を作成する場合、語彙ファイルはテキスト (*.txt) 形式である必要があります。を使用する場合 AWS Management Console、語彙ファイルはテキスト (*.txt) 形式またはカンマ区切り値 (*.csv) 形式にすることができます。
- カスタム語彙内の各エントリは 256 文字を超えることはできません。
- カスタム語彙を使用するには、文字起こし AWS リージョン と同じ で作成されている必要があります。

> **Tip**
>
> を使用してカスタム語彙をテストできます AWS Management Console。カスタム語彙を使用する準備ができたら、 にログインし AWS Management Console、リアルタイム文字起こし を選択し、カスタマイズ にスクロールし、カスタム語彙 をオンにして、ドロップダウンリストからカスタム語彙を選択します。次に [ストリーミングを開始する] を選択します。カスタム語彙のいくつかの単語をマイクに向かって話し、正しくレンダリングされるかどうかを確認します。

カスタム語彙テーブルとリスト

> **Important**
>
> リスト形式のカスタム語彙は廃止される予定です。新しいカスタム語彙を作成する場合は、テーブル形式を使用してください。

テーブルを使用すると、カスタム語彙内の単語の入出力に対するオプションがより多くなり、より詳細に制御できます。テーブルでは、出力を微調整できるように、複数のカテゴリ (Phrase and DisplayAs) を指定する必要があります。

リストには追加のオプションがないため、入力できるのは文字起こしに表示したいエントリのみで、スペースはすべてハイフンに置き換えます。

AWS Management Console、 AWS CLI、 AWS SDKs はすべて同じ方法でカスタム語彙テーブルを使用します。リストはメソッドごとに異なるため、メソッド間で正常に使用するために追加のフォーマットが必要になる場合があります。

詳細については、「テーブルを使用してカスタム語彙を作成する」および「リストを使用してカスタム語彙を作成する」を参照してください。

もう少し深く掘り下げて、カスタム語彙で Amazon Augmented AI を使用する方法を学ぶには、「Amazon Transcribeによるヒューマンレビューの作成を始める」を参照してください。

> カスタム語彙に固有の API オペレーション
>
> CreateVocabulary, DeleteVocabulary, GetVocabulary, ListVocabularies, UpdateVocabulary

テーブルを使用してカスタム語彙を作成する

カスタム語彙を作成するには、テーブル形式を使用することをおすすめします。語彙テーブルは 4 つの (Phrase, SoundsLike, IPA, and DisplayAs) 列で構成されている必要があり、どの順序でも含めることができます。

フレーズ	SoundsLike	IPA	DisplayAs
必須。テーブルのすべての行には、この列のエントリが含まれている必要があります。 この列にはスペースを使用しないでください。 エントリに複数の単語が含まれている場合は、各単語をハイフン (-) で区切ります。例えば、**Andorra-la-Vella**、**Los-Angeles** などです。	SoundsLike はカスタム語彙ではサポートされなくなりました。列は空のままにしてください。この列の値はすべて無視されます。この列のサポートは今後削除されます。	IPA はカスタム語彙ではサポートされなくなりました。列は空のままにしてください。この列の値はすべて無視されます。この列のサポートは今後削除されます。	オプション。この列の行は空のままでかまいません。 この列にはスペースを使用できます。 文字起こし出力でのエントリの表示方法を定義します。たとえば、Phrase 列の **Andorra-la-Vella** は DisplayAs 列の **Andorra la Vella** にあります。 この列の行が空の場合、 はPhrase列の内

フレーズ	SoundsLike	IPA	DisplayAs
頭字語の場合は、発音する文字をすべてピリオドで区切る必要があります。末尾のピリオドも発音する必要があります。頭字語が複数形の場合は、頭字語と「s」の間にハイフンを使用する必要があります。たとえば、「CLI」は **C.L.I.** (**C.L.I** ではない) で、「ABCs」は **A.B.C.-s** (**A.B.C-s** ではない) です。 フレーズが単語と頭字語の両方で構成されている場合は、これら 2 つの要素をハイフンでつなぐ必要があります。たとえば、「DynamoDB」は **Dynamo-D.B.** です。 この列には数字を含めないでください。数字はスペルアウトする必要があります。たとえば、「VX 02Q」は **V.X.-zero-two-Q.** です。			容 Amazon Transcribe を使用して出力を決定します。 この列には数字 (0-9) を含めることができます。

テーブルを作成する際の注意事項

- テーブルには、4 つの列ヘッダー がすべて含まれている必要があります(Phrase, SoundsLike, IPA, and DisplayAs)。Phrase 列には、各行にエントリが含まれている必要があります。IPA および を介して発音入力を提供する機能はサポートされSoundsLikeなくなり、列を空のままにすることができます。これらの列の値は無視されます。
- 各列は TAB またはカンマ (,) で区切る必要があります。これはカスタム語彙ファイルのすべての行に適用されます。行に空の列がある場合でも、各列に区切り記号 (TAB またはカンマ) を含める必要があります。
- スペースは IPA 列と DisplayAs 列のみ使用できます。列を区切るのにスペースを使用しないでください。
- IPA および SoundsLikeは、カスタム語彙ではサポートされなくなりました。列は空のままにしてください。これらの列の値はすべて無視されます。この列のサポートは今後削除されます。
- DisplayAs 列は記号と特殊文字 (C++ など) をサポートします。他のすべての列は、使用している言語の文字セットページに記載されている文字をサポートします。
- Phrase 列に数字を含めたい場合は、数字をスペルアウトする必要があります。数字 (0-9) は DisplayAs 列でのみサポートされています。
- テーブルは LF 形式のプレーンテキスト (*.txt) ファイルとして保存する必要があります。CRLF など、他の形式を使用した場合、カスタム語彙は処理できません。
- 文字起こしリクエストに含めるCreateVocabulary前に、カスタム語彙ファイルを Amazon S3 バケットにアップロードし、 を使用して処理する必要があります。手順については、「カスタム語彙テーブルを作成する」を参照してください。

> **Note**
>
> 頭字語など、1 文字ずつ個別に発音する単語は、ピリオド (**A.B.C.**) で区切って 1 文字で入力します。「ABC」のように複数形の頭字語を入力するには、「s」と頭字語をハイフン (**A.B.C.-s**) で区切ります。頭字語の入力には、大文字と小文字のどちらでも使用できます。頭字語はすべての言語には対応していません。「サポートされている言語および言語固有の機能」を参照してください。

カスタム語彙テーブル (**[TAB]** はタブ文字を表す) の例を以下に示します。

```
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
```

見やすくするために、同じ表に列をそろえて示します。カスタム語彙テーブルの列間にスペースを入れないでください。前の例のようにテーブルの位置がずれて見えるはずです。

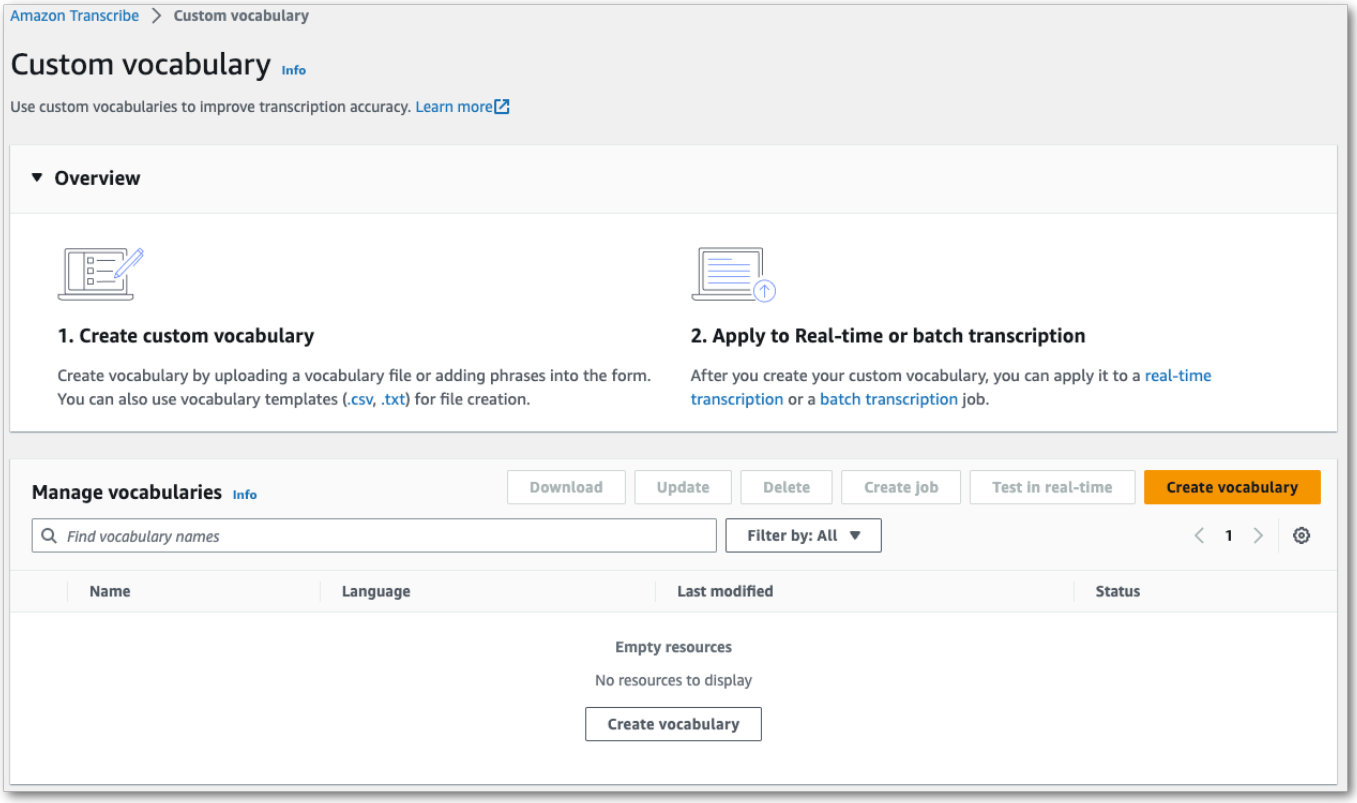
```
Phrase          [TAB]SoundsLike         [TAB]IPA                [TAB]DisplayAs
Los-Angeles     [TAB]                   [TAB]                   [TAB]Los Angeles
Eva-Maria       [TAB]                   [TAB]                   [TAB]
A.B.C.-s        [TAB]                   [TAB]                   [TAB]ABCs
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
```

カスタム語彙テーブルを作成する

で使用するカスタム語彙テーブルを処理するには Amazon Transcribe、次の例を参照してください。

AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、[カスタム語彙] を選択します。カスタム語彙のページが開き、既存の語彙の表示したり、新しい語彙を作成したりできます。
3. [語彙の作成] を選択します。

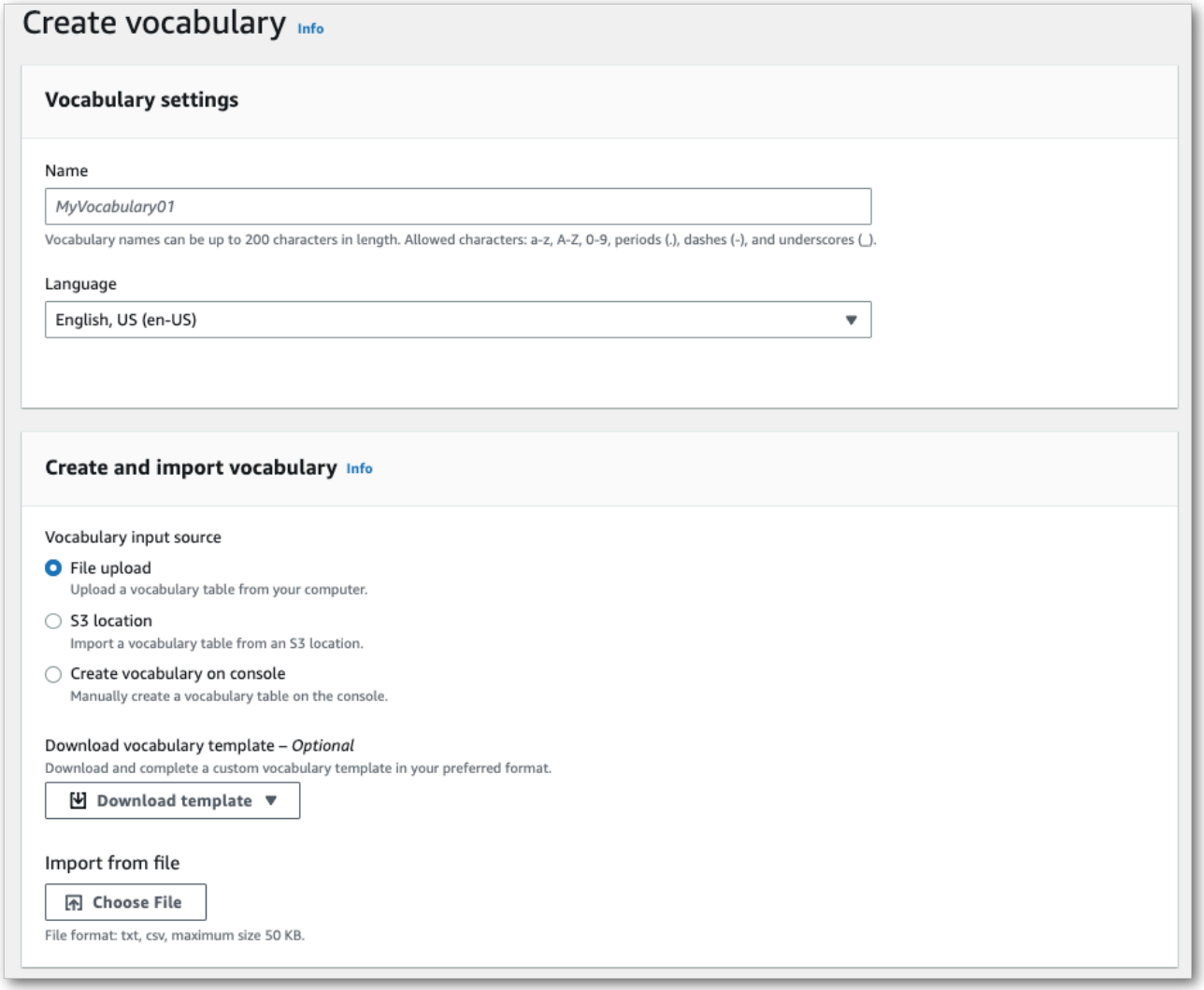


「語彙の作成」ページに移動します。新しいカスタム語彙の名前を入力します。

次の 3 つの選択肢があります。

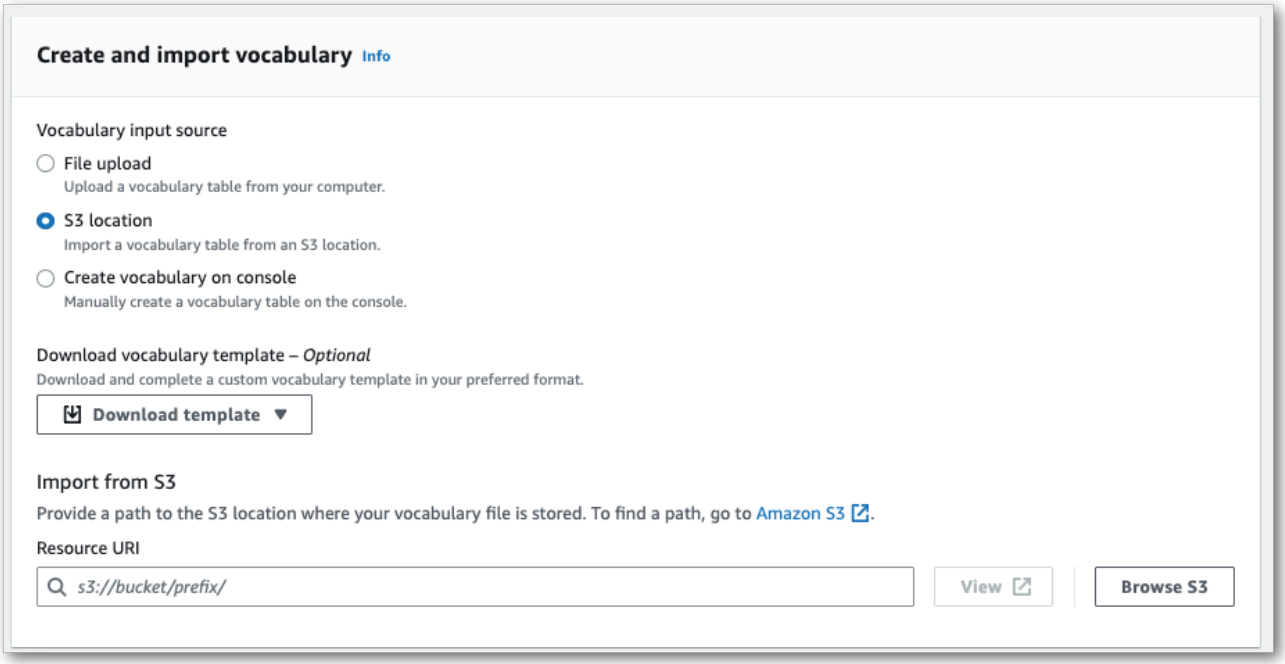
a. コンピュータから txt または csv ファイルをアップロードします。

   カスタム語彙を一から作成することも、テンプレートをダウンロードして始めることもできます。その後、語彙の表示と編集ペインに語彙が自動入力されます。



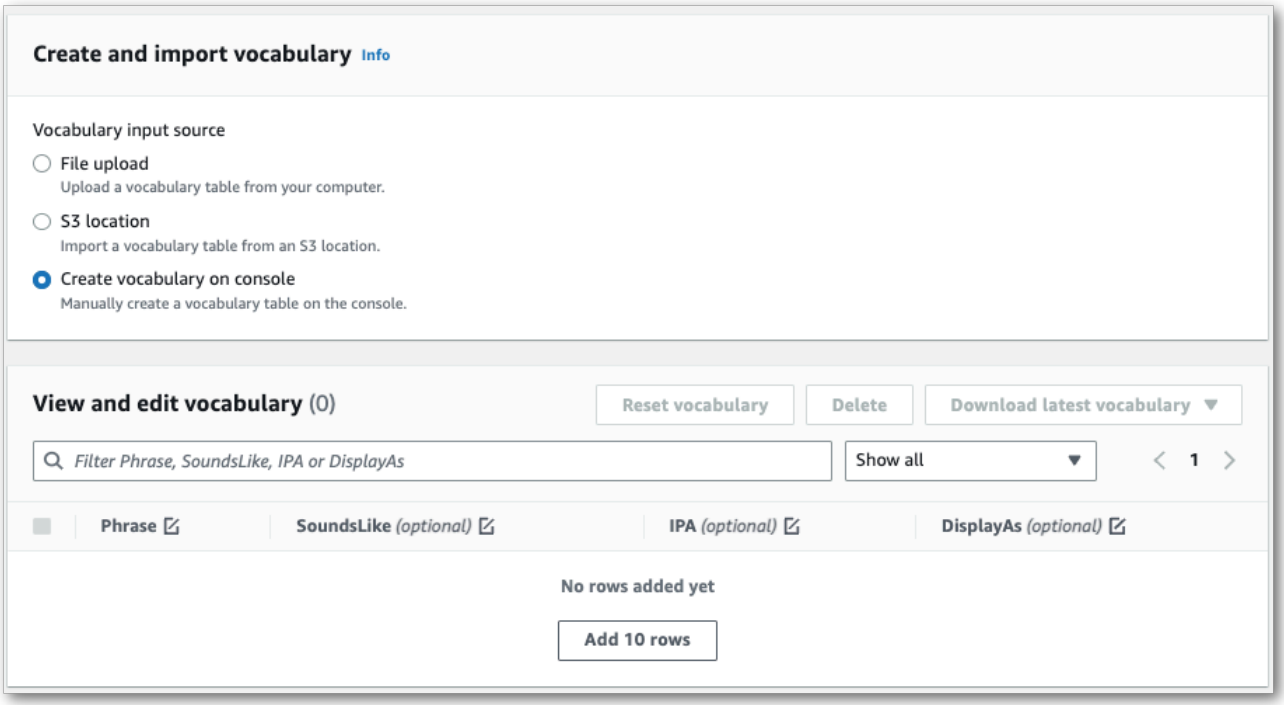
b. Amazon S3 場所から txt または csv ファイルをインポートします。

   カスタム語彙を一から作成することも、テンプレートをダウンロードして始めることもできます。完成した語彙ファイルを Amazon S3 バケットにアップロードし、リクエストにその URI を指定します。その後、語彙の表示と編集ペインに語彙が自動入力されます。

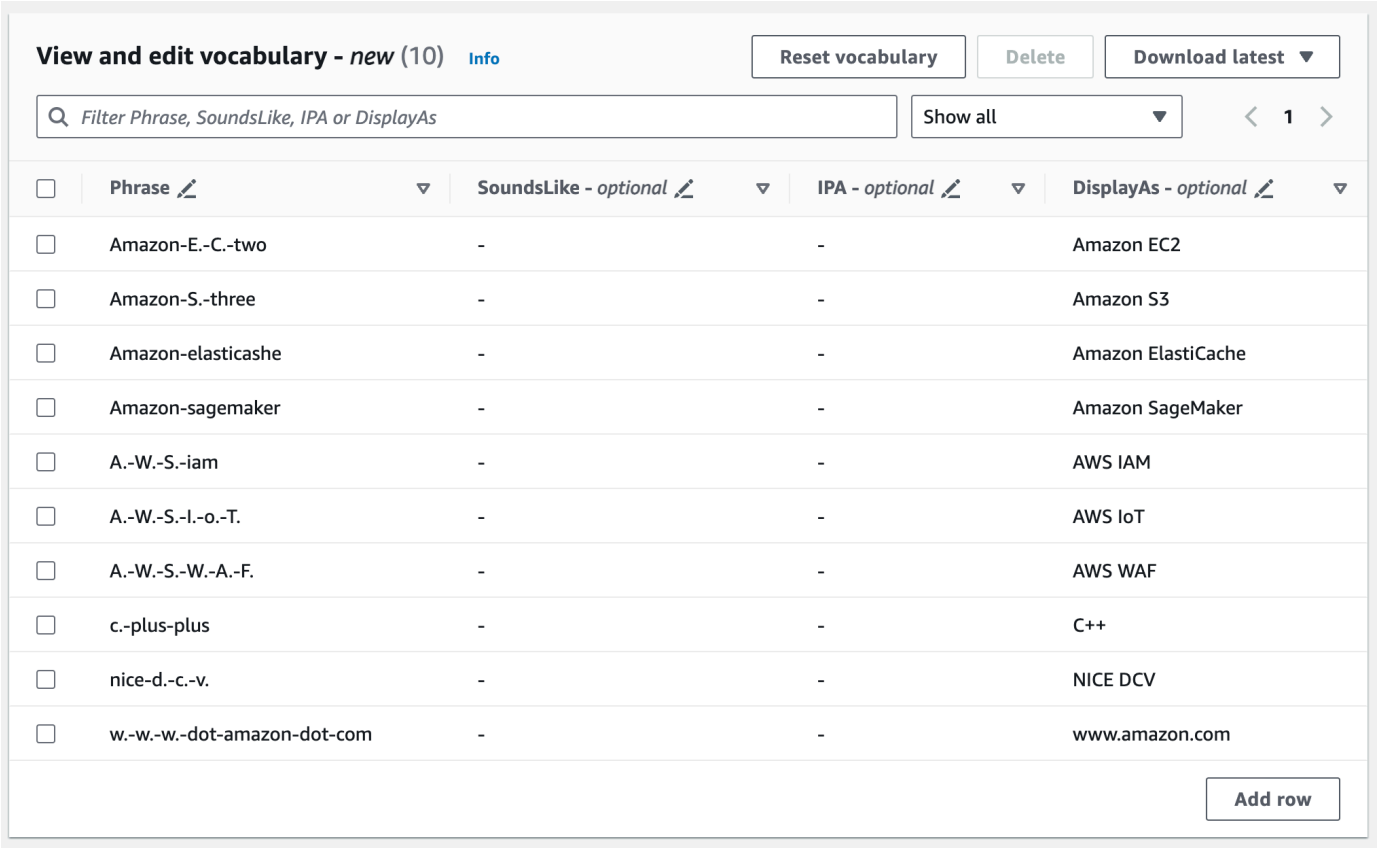


c. コンソールで語彙を手動で作成します。

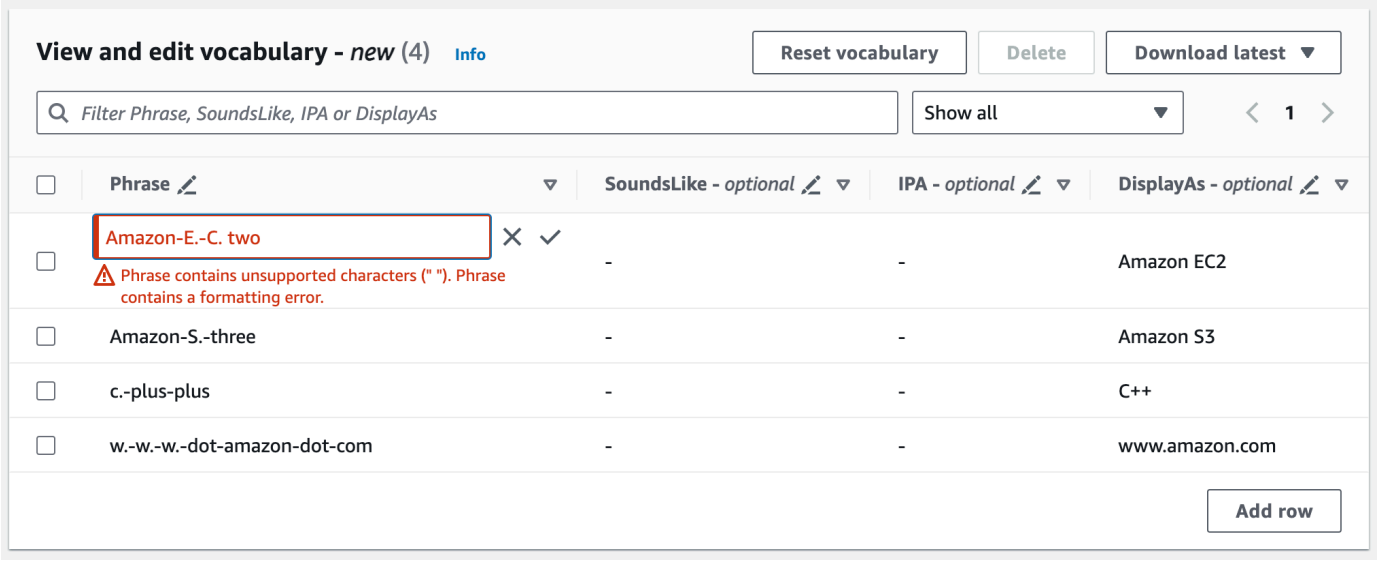
語彙の表示と編集ペインまでスクロールし、[10 行追加] を選択します。用語を手動で入力できるようになりました。



4. 語彙の表示と編集ペインで語彙を編集できます。変更するには、変更するエントリをクリックします。

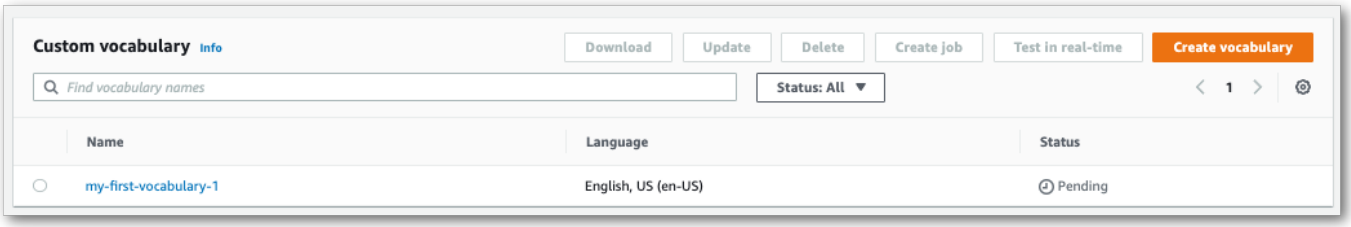


エラーがあると詳細なエラーメッセージが表示されるので、語彙を処理する前に問題を修正できます。[語彙の作成] を選択する前にすべてのエラーを修正しないと、語彙のリクエストは失敗するので注意してください。

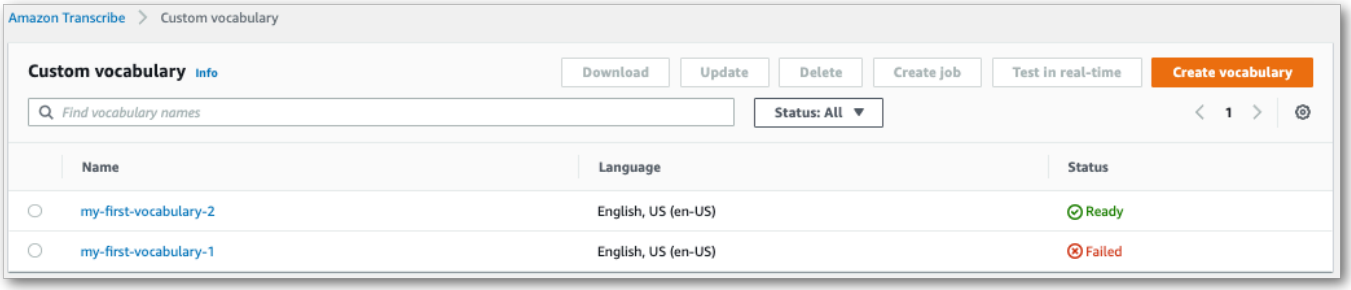


チェックマーク (✓) を選択して変更を保存するか、「X」を選択して変更を破棄します。

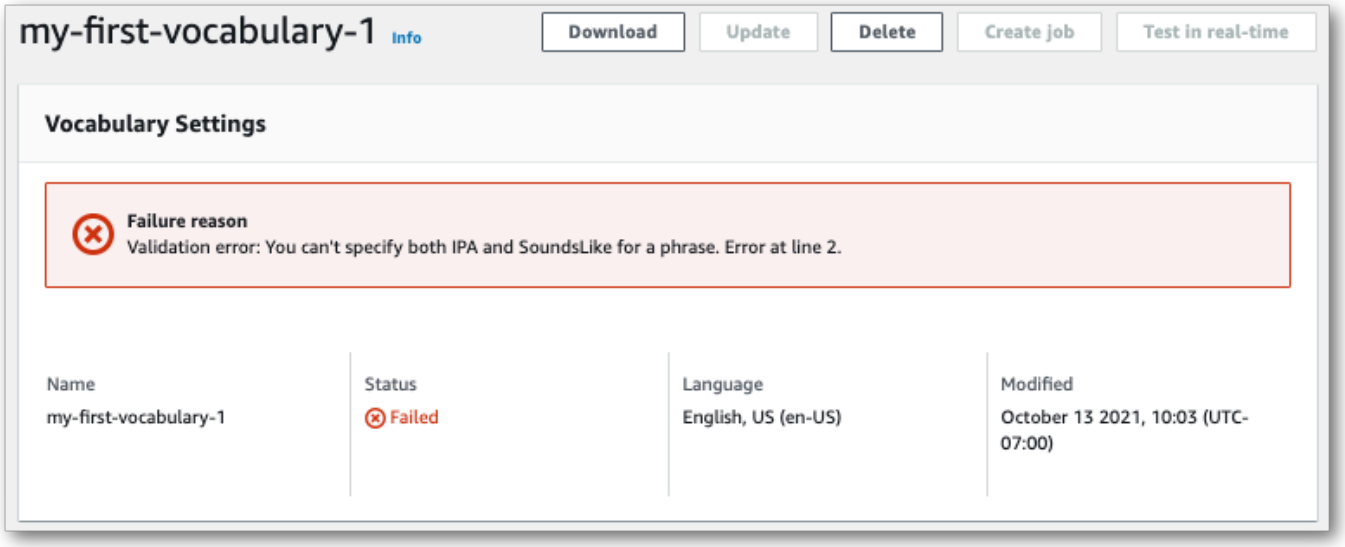
5. オプションで、カスタム語彙にタグを追加します。すべてのフィールドを入力し、語彙に問題がなければ、ページの一番下にある [語彙の作成] を選択します。カスタム語彙のページに戻ると、カスタム語彙のステータスを確認できます。ステータスが「保留中」から「準備完了」に変わったら、カスタム語彙を文字起こしに使用できます。



6. ステータスが「失敗」に変わったら、カスタム語彙の名前を選択して、その語彙の情報ページに移動します。



このページの上部には、カスタム語彙が失敗した理由に関する情報が記載された失敗の理由バナーがあります。テキストファイルのエラーを修正して、もう一度試してください。



AWS CLI

この例では、テーブル形式の語彙ファイルで語彙の作成コマンドを使用します。詳細については、「CreateVocabulary」を参照してください。

文字起こしジョブで既存のカスタム語彙を使用するには、StartTranscriptionJobオペレーションを呼び出すときに VocabularyNameSettingsフィールドに を設定するか、 からドロップダウンリストからカスタム語 AWS Management Console彙を選択します。

```
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
```

ここでは、語彙の作成コマンドと、カスタム語彙を作成するリクエストボディを使用した別の例を示します。

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

ファイル my-first-vocab-table.json には、次のリクエスト本文が含まれています。

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

VocabularyState をPENDING から READY に変更すると、カスタム語彙を文字起こしに使用できるようになります。カスタム語彙の現在のステータスを表示するには、以下を実行します。

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

AWS SDK for Python (Boto3)

この例では AWS SDK for Python (Boto3) 、 を使用して create_vocabulary メソッドを使用してテーブルからカスタム語彙を作成します。詳細については、「CreateVocabulary」を参照してください。

文字起こしジョブで既存のカスタム語彙を使用するには、StartTranscriptionJobオペレーションを呼び出すときに VocabularyNameSettingsフィールドに を設定するか、 からドロップダウンリストからカスタム語 AWS Management Console彙を選択します。

機能固有の例、シナリオ例、クロスサービス例など、 AWS SDKs「」のSDK を使用した Amazon Transcribe のコード例 AWS SDKs章を参照してください。

```
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**
>
> カスタム語彙ファイル用に新しい Amazon S3 バケットを作成する場合は、CreateVocabularyリクエストを行う IAM ロールにこのバケットへのアクセス許可があることを確認してください。ロールに正しいアクセス許可がない場合、リクエストは失敗します。オプションで、 DataAccessRoleArnパラメータを含めることで、リクエスト内で IAM ロールを指定できます。の IAM ロールとポリシーの詳細については、 Amazon Transcribe「」を参照してくださいAmazon Transcribe アイデンティティベースポリシーの例。

リストを使用してカスタム語彙を作成する

> **Important**
>
> リスト形式のカスタム語彙は廃止されつつあるため、新しいカスタム語彙を作成する場合は、表形式を使用することを強くお勧めします。

、、または AWS SDK を使用してリストからカスタムボキャブラリを作成できます。AWS Management Console AWS CLI

- AWS Management Console: カスタムボキャブラリを含むテキストファイルを作成してアップロードする必要があります。行で区切られたエントリまたはカンマで区切られたエントリを使用できます。リストはテキスト (*.txt) ファイルとしてフォーマットして保存する必要があることに注意してください。LFなど、他の形式を使用した場合CRLF、カスタム語彙は受け入れられませんAmazon Transcribe。
- AWS CLIおよび AWSSDK: フラグを使用して API 呼び出しにカスタム語彙をカンマで区切ったエントリとして含める必要があります。`Phrases`

エントリに複数の単語が含まれている場合は、各単語をハイフンでつなぐ必要があります。たとえば、「ロサンゼルス」を、「**Los-Angeles**アンドララベリャ」をとして含めます。**Andorra-la-Vella**

2 つの有効なリスト形式の例を次に示します。カスタム語彙リストの作成メソッド固有の例については、を参照してください。

- カンマで区切られたエントリ:

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

- 行で区切られたエントリ:

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

> **Important**
>
> 使用する言語でサポートされている文字のみを使用できます。詳細については、ご使用の言語の文字セットを参照してください。

CreateMedicalVocabularyこの操作では、カスタム語彙リストはサポートされていません。カスタム医療用語を作成する場合は、表形式を使用する必要があります。テーブルを使用してカスタム語彙を作成する手順については、を参照してください。

カスタム語彙リストの作成

カスタムボキャブラリーリストを処理して使用するにはAmazon Transcribe、次の例を参照してください。

AWS CLI

この例では、リスト形式のカスタム語彙ファイルで create-` コマンドを使用しています。詳細については、「CreateVocabulary」を参照してください。

```
aws transcribe create-vocabulary \
--vocabulary-name my-first-vocabulary \
--language-code en-US \
--phrases {CLI,Eva-Maria,ABCs}
```

ここでは、create-langualコマンドと、カスタム語彙を作成するリクエストボディを使用した別の例を示します。

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

ファイル my-first-vocab-list.json に次のリクエストボディが入ります。

```
{
  "VocabularyName": "my-first-vocabulary",
  "LanguageCode": "en-US",
  "Phrases": [
        "CLI","Eva-Maria","ABCs"
  ]
}
```

VocabularyStatePENDINGからに変更したらREADY、カスタムボキャブラリーを文字起こしで使用できるようになります。カスタム語彙の現在のステータスを表示するには、以下を実行します。

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

AWS SDK for Python (Boto3)

この例では、AWS SDK for Python (Boto3)を使用して create_` メソッドを使用してリストからカスタムボキャブラリを作成します。詳細については、「CreateVocabulary」を参照してください。

機能固有の例、シナリオ、クロスサービスの例など、AWS SDK を使用するその他の例については、この章を参照してください。SDK を使用した Amazon Transcribe のコード例 AWS SDKs

```
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**
>
> Amazon S3カスタムボキャブラリーファイル用に新しいバケットを作成する場合は、IAMCreateVocabularyリクエストを行うロールにこのバケットにアクセスする権限があることを確認してください。ロールに正しいアクセス許可がない場合、リクエストは失敗します。DataAccessRoleArnパラメータを含めることで、IAMリクエスト内のロールを任

意で指定できます。IAMのロールとポリシーの詳細についてはAmazon Transcribe、を参照してくださいAmazon Transcribe アイデンティティベースポリシーの例。

Using a custom vocabulary

カスタムボキャブラリーを作成したら、それを文字起こしリクエストに含めることができます。例については、次のセクションを参照してください。

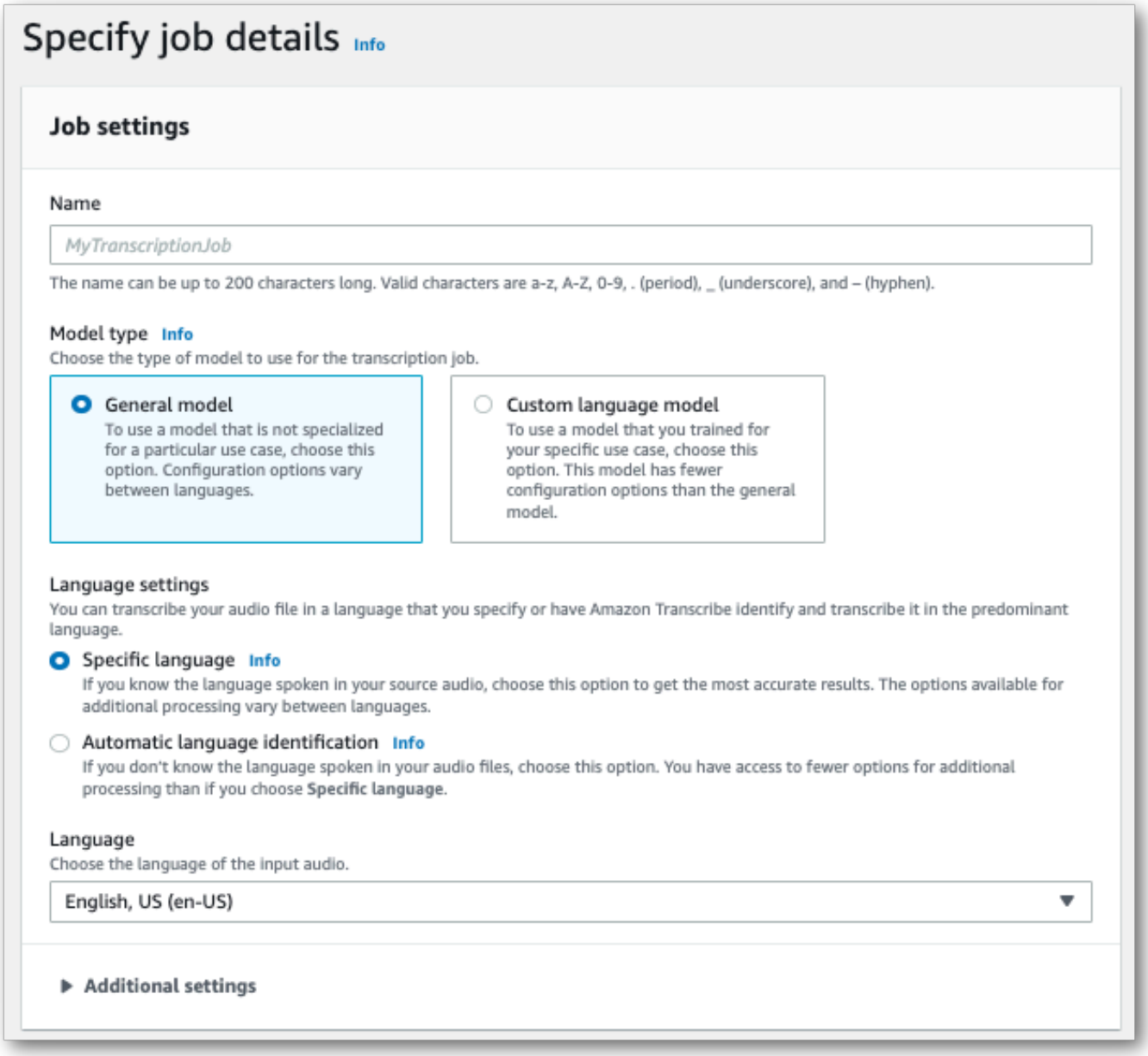
リクエストに含めるカスタムボキャブラリーの言語は、メディアに指定する言語コードと一致する必要があります。言語が一致しない場合、カスタムボキャブラリーは文字起こしに適用されず、警告やエラーもありません。

バッチトランスクリプションでのカスタムボキャブラリーの使用

バッチトランスクリプションでカスタムボキャブラリーを使用するには、次の例を参照してください。

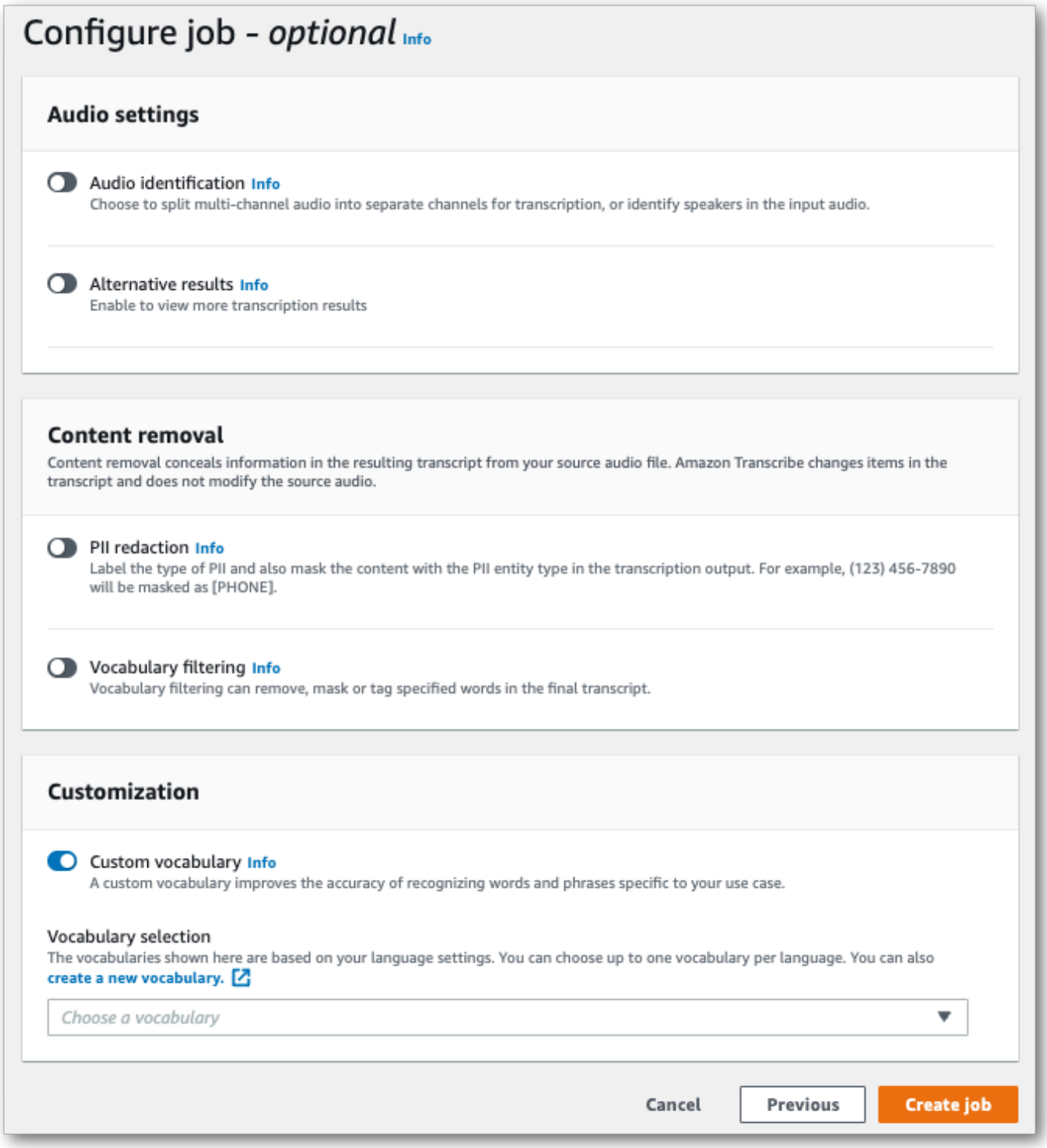
AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、変換ジョブ、ジョブの表示 これにより、ジョブの詳細を指定 ページが開きます。



ジョブに名前を付け、入力メディアを指定します。オプションで他のフィールドを追加して、[次へ] を選択します。

3. ジョブ設定ページの下部にあるカスタマイズパネルで、カスタムボキャブラリーをオンに切り替えます。



4. Select your custom vocabulary from the dropdown menu.

   ジョブの表示

AWS CLI

この例では、start-transcription-jobSettingsVocabularyNameコマンドとパラメーターをサブパラメーターと共に使用しています。詳細については、StartTranscriptionJobおよびSettingsを参照してください。

```
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
```

start-transcription-jobコマンドと、そのジョブのカスタム語

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

my-first-vocabulary-job.json ファイルには、次のリクエストボディボディボディが入ります。

```
{
  "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)

この例では、AWS SDK for Python (Boto3) transstart_transcription_job メソッドのSettings引数で、カスタム語 詳細については、StartTranscriptionJobおよびSettingsを参照してください。

機能固有、シナリオ、サービス間の例など、AWS SDK を使用するその他の例については、SDK を使用した Amazon Transcribe のコード例 AWS SDKsこの章を参照してください。

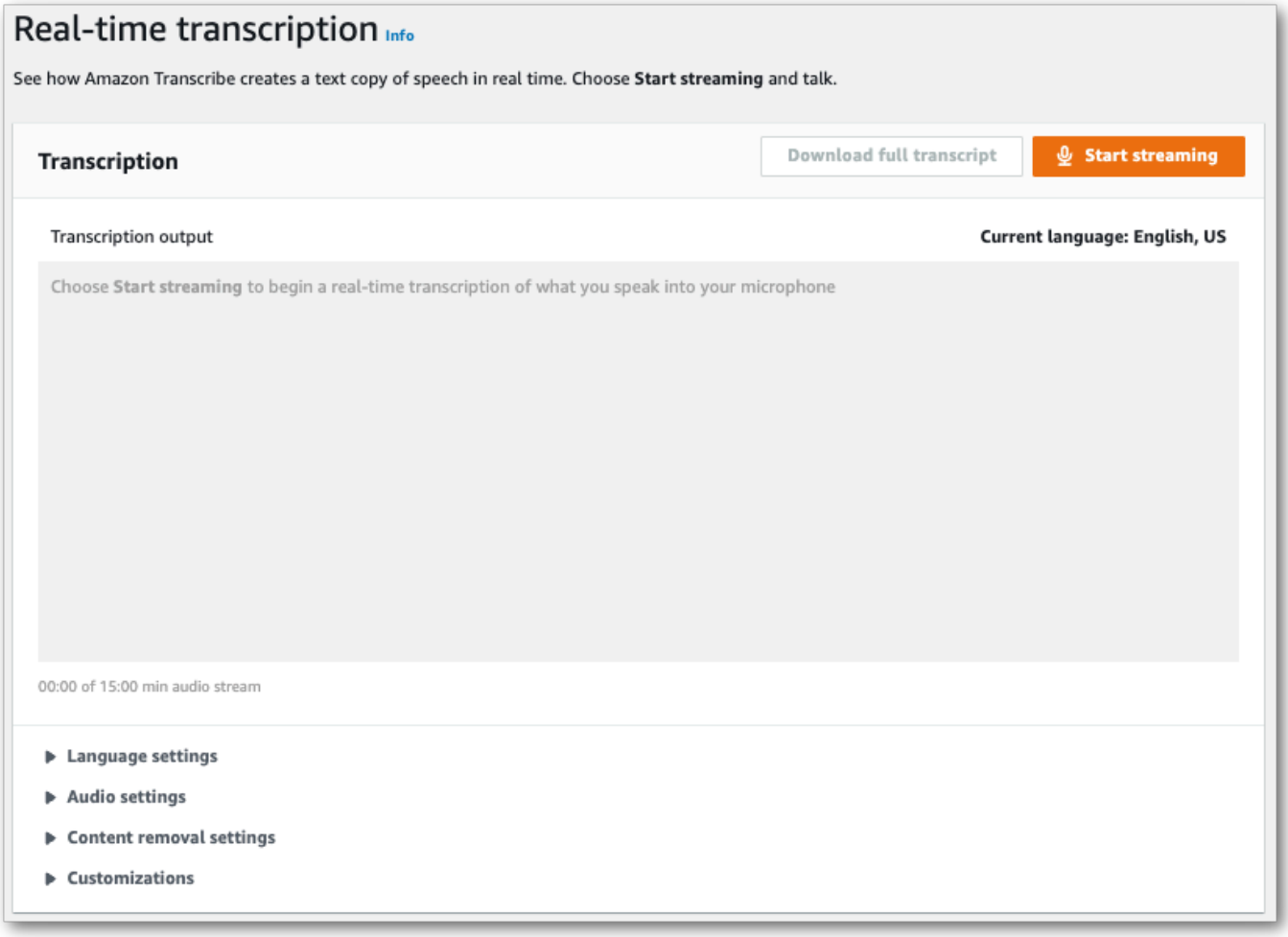
```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    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)
```

ストリーミング文字起こしでのカスタムボキャブラリーの使用

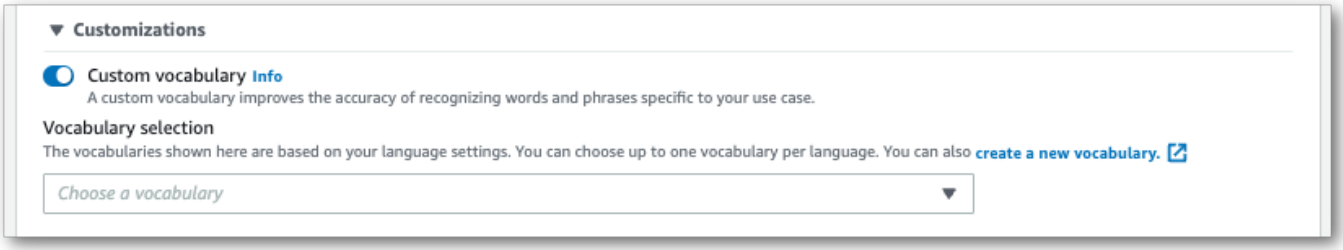
ストリーミング文字起こしでカスタムボキャブラリーを使用するには、以下の例を参照してください。

AWS Management Console

1. AWS Management Console にサインインします。
2. ナビゲーションペインで、[リアルタイム文字起こし] を選択します。カスタムフィールドフィールドフィールドフィールドフィールドフィールドが最小化されている場合はこのフィールドを展開



3. カスタムボキャブラリーをオンにして、ドロップダウンメニューからカスタムボキャブラリーを選択します。



ストリームに適用するその他の設定を行います。

4. これで、ストリームを書き起こす準備ができました。[ストリーミングを開始] を選択し、話し始めます。ディクテーションを終了するには、「ストリーミングを停止」を選択します。

HTTP/2 ストリーミング

この例では、カスタム語 HTTP/2 ストリーミングでHTTP/2 ストリーミングを使用する際の詳細についてはAmazon Transcribe、を参照してくださいHTTP/2 ストリームの設定。特定のパラメータとヘッダーヘッダーの詳細についてはAmazon Transcribe、を参照してください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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

WebSocket ストリーム

この例では、 WebSocketカスタムボキャブラリーをストリームに適用する署名付き URL を作成します。読みやすくするために、改行が追加されています。 WebSocketでのストリームの使用の詳細についてはAmazon Transcribe、を参照してください WebSocket ストリームのセットアップ。パラメータの詳細については、「StartStreamTranscription」を参照してください。

```
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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

カスタム言語モデル

カスタム言語モデルは、ドメイン固有の音声の文字起こしの精度を向上させるように設計されています。これには、通常の日常会話で聞く内容以外の内容も含まれます。たとえば、科学会議の議事録を転記する場合、標準的な文字起こしでは、発表者が使用する科学用語の多くを認識することはまずありません。このような場合は、自分の専門分野で使用されている専門用語を認識するようにカスタム言語モデルをトレーニングできます。

ヒント (発音など) を提供することで単語の認知度を高めるカスタム語彙とは異なり、カスタム言語モデルは特定の単語に関連する文脈を学習します。これには、単語がいつどのように使用されるか、および単語と他の単語との関係が含まれます。たとえば、気候科学の研究論文を使用してモデルをトレーニングすると、モデルが「氷流」よりも「氷流」という語句の組み合わせである可能性が高いことを学習する可能性があります。

カスタム言語モデルでサポートされている言語を確認するには、を参照してくださいサポートされている言語および言語固有の機能。リクエストにカスタム言語モデルを含めると、言語識別を有効にできないことに注意してください (言語コードを指定する必要があります)。

> **カスタム言語モデル固有の API オペレーション**
>
> CreateLanguageModel, DeleteLanguageModel, DescribeLanguageModel, ListLanguageModels

データソース

モデルのトレーニングには、どのような種類のテキストデータでも使用できます。ただし、テキストコンテンツがオーディオコンテンツに近いほど、モデルの精度は高くなります。そのため、音声と同じ文脈で同じ用語を使用するテキストデータを選択することが重要です。

モデルのトレーニングに最適なデータは、正確なトランスクリプトです。これはドメイン内データと見なされます。ドメイン内のテキストデータには、書き起こしたい音声とまったく同じ用語、用法、コンテキストが含まれます。

正確なトランスクリプトがない場合は、ジャーナル記事、テクニカルレポート、ホワイトペーパー、会議議事録、取扱説明書、ニュース記事、ウェブサイトのコンテンツ、およびオーディオと同様の文脈で使用される必要な用語を含むその他のテキストを使用してください。これはドメイン関連データと見なされます。

堅牢なカスタム言語モデルを作成するには、大量のテキストデータが必要になる場合があり、そのデータには音声で話されている用語が含まれている必要があります。Amazon Transcribeモデルのトレーニング用に最大 2 GB のテキストデータを指定できます。これはトレーニングデータと呼ばれます。オプションで、ドメイン内のトランスクリプトがない（または少ない）場合は、モデルを調整するために最大 200 MB Amazon Transcribe のテキストデータを提供できます。これをチューニングデータと呼びます。

トレーニングデータとチューニングデータ

トレーニングデータの目的は、新しい用語を認識し、それらの用語がどのような文脈で使われるかを学ぶことです。Amazon Transcribe堅牢なモデルを作成するには、Amazon Transcribe大量の関連テキストデータが必要になる場合があります。2 GB の制限まで、できるだけ多くのトレーニングデータを提供することを強くお勧めします。

データをチューニングする目的は、トレーニングデータから学習したコンテクスト・リレーションシップの改善と最適化を支援することです。カスタム言語モデルの作成には、チューニングデータは必要ありません。

トレーニングの方法を決めるのはユーザーです。そして必要に応じてチューニングデータを決めるのはユーザーです。それぞれのケースは異なり、所有しているデータの種類と量によって異なります。ドメイン内のトレーニングデータが不足している場合は、チューニングデータを使用することをおすすめします。

両方のデータ型を含める場合は、トレーニングデータとチューニングデータを重複させないでください。トレーニングとチューニングのデータは一意でなければなりません。データが重複すると、カスタム言語モデルに偏りや歪みが生じ、精度に影響する可能性があります。

一般的なガイダンスとして、可能な限り正確なドメイン内のテキストをトレーニングデータとして使用することをお勧めします。一般的なシナリオで、で指定した順に一覧表示されます。

- ドメイン内の正確なトランスクリプトテキストが10,000語以上ある場合は、それをトレーニングデータとして使用してください。この場合、チューニングデータを含める必要はありません。これは、カスタム言語モデルのトレーニングに最適なシナリオです。
- 10,000語未満の正確なドメイン内のトランスクリプトテキストがあっても、期待した結果が得られない場合は、テクニカルレポートなどのドメイン関連のテキストでトレーニングデータを補足することを検討してください。この場合、ドメイン内のトランスクリプトデータのごく一部（10 ~ 25%）をチューニングデータとして使用するために予約してください。
- ドメイン内のトランスクリプトテキストがない場合は、ドメイン関連のすべてのテキストをトレーニングデータとしてアップロードします。この場合、書かれたテキストよりもトランスクリプト形式のテキストの方が適しています。これは、カスタム言語モデルのトレーニングには最も効果の低いシナリオです。

モデルを作成する準備ができたら、を参照してくださいカスタム言語モデルの作成。

カスタム言語モデルの作成

カスタム言語モデルを作成したい場合、次のことを行う必要があります。

- データを準備する。データはプレーンテキスト形式で保存する必要があり、特殊文字を含めることはできません。
- Amazon S3データをバケットにアップロードします。トレーニングデータとチューニングデータ用に別々のフォルダーを作成することをお勧めします。
- Amazon S3バケットにアクセスできることを確認してくださいAmazon Transcribe。データを使用するには、IAMアクセス権限を持つロールを指定する必要があります。

データを準備する

すべてのデータを 1 つのファイルにまとめることも、複数のファイルとして保存することもできます。チューニングデータを含める場合は、トレーニングデータとは別のファイルに保存する必要があることに注意してください。

トレーニングやチューニングデータに使用するテキストファイルの数は関係ありません。100,000 ワードのファイルを 1 つアップロードすると、10,000 ワードのファイルを 10 個アップロードしたときと同じ結果になります。都合の良い方法でテキストデータを準備してください。

すべてのデータファイルが次の基準を満たすことを確認してください。

- これらはすべて、作成したいモデルと同じ言語であるか。例えば、米国英語 (en-US) の音声を書き起こすカスタム言語モデルを作成したい場合、テキストデータはすべて米国英語である必要があります。
- UTF-8 エンコーディングのプレーンテキスト形式です。
- HTML タグなどの特殊文字や書式設定は含まれません。
- これらを合わせたサイズは、トレーニングデータでは最大で 2 GB、チューニングデータについては 200 MB です。

これらの基準のいずれかが満たされない場合、モデルは不合格となります。

データをアップロードする

データをアップロードする前に、トレーニングデータ用の新しいフォルダーを作成します。チューニングデータを使用する場合は、別のフォルダーを作成してください。

バケットの URI は次のようになります。

- `s3://DOC-EXAMPLE-BUCKET/my-model-training-data/`
- `s3://DOC-EXAMPLE-BUCKET/my-model-tuning-data/`

トレーニングとチューニングのデータを適切なバケットにアップロードします。

後日、これらのバケットにさらにデータを追加できます。ただし、その場合は、新しいデータを使用してモデルを再作成する必要があります。既存のモデルを新しいデータで更新することはできません。

データへのアクセスを許可する

カスタム言語モデルを作成するには、IAMAmazon S3バケットにアクセスするアクセス許可を持つロールを指定する必要があります。Amazon S3トレーニングデータを置いたバケットにアクセスできるロールをまだ持っていない場合は、ロールを作成してください。ロールを作成した後、ポリシーをアタッチして、ロールにアクセス許可を付与できます。ポリシーをユーザーにアタッチしないでください。

エンドポイントポリシーの例については、Amazon Transcribe アイデンティティベースポリシーの例を参照してください。

新しい IAM ID を作成する方法については、「IAMID (ユーザー、ユーザーグループ、ロール)」を参照してください。

ポリシーの詳細については、次の情報を参照してください。

- のポリシーと権限 IAM
- IAMポリシーの作成
- AWS リソースのアクセス管理

カスタム言語モデルの作成

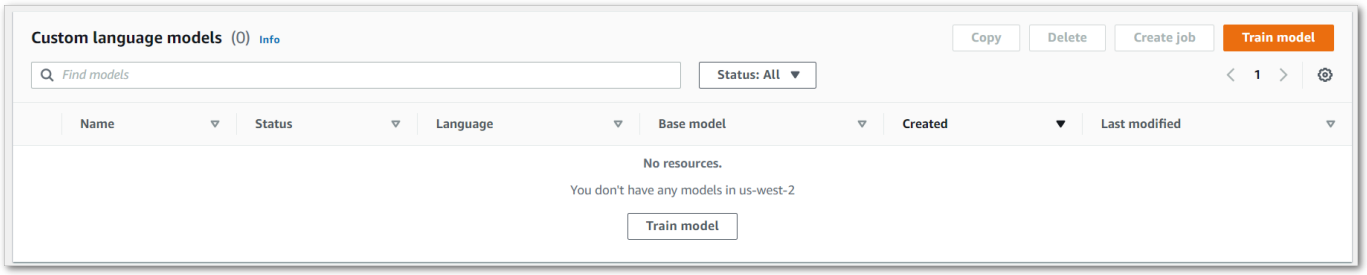
カスタム言語モデルを作成するときは、基本モデルを選択する必要があります。2 つのベースモデルのオプションがあります。

- `NarrowBand`: サンプルレートが 16,000 Hz 未満の音声にこのオプションを使用します。このモデルタイプは、通常 8,000 Hz で録音された電話での会話に使用されます。
- `WideBand`: サンプルレートが 16,000 Hz 以上の音声にこのオプションを使用します。

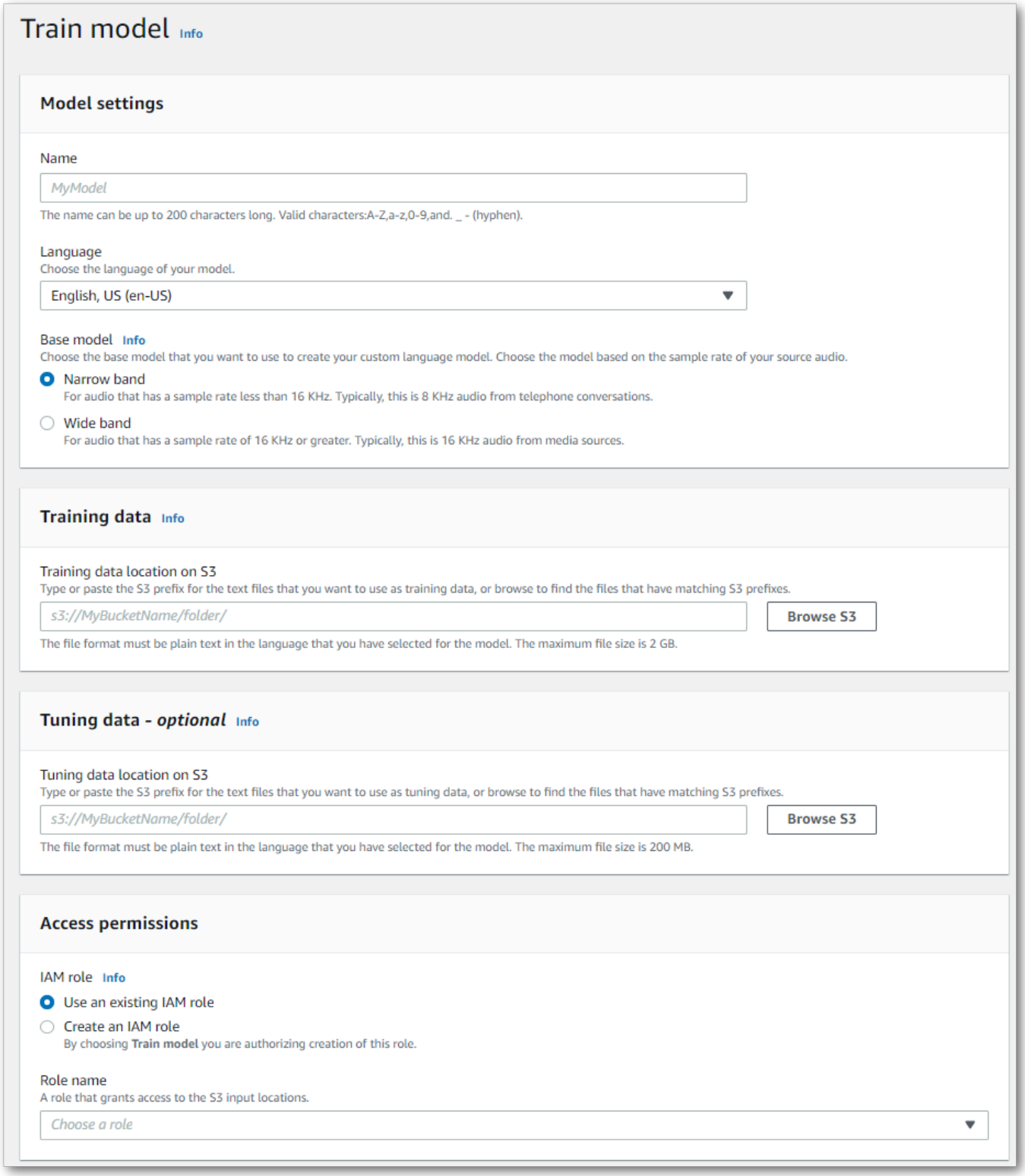
AWS Management Console、AWS CLIまたは AWS SDK を使用してカスタム言語モデルを作成できます。次の例を参照してください。

AWS Management Console

1. AWS Management Console にサインインします。
2. ナビゲーションペインで [カスタム言語モデル] を選択します。カスタム言語モデルのページが開き、既存のカスタム言語モデルを表示したり、新しいカスタム言語モデルをトレーニングしたりできます。
3. 新しいモデルをトレーニングするには、モデルのトレーニングを選択します。



列車のモデルページに移動します。名前を追加し、言語を指定して、モデルに必要な基本モデルを選択します。次に、トレーニングへのパスと、オプションでチューニングデータを追加します。IAMデータにアクセスする権限を持つロールを含める必要があります。



4. すべてのフィールドを完了したら、モデルのトレーニングを選択します。

AWS CLI

この例では、create-language-modelコマンドを使用しています。詳細については、CreateLanguageModelおよびLanguageModelを参照してください。

```
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
```

ここでは、create-language-modelコマンドと、カスタム言語モデルを作成するリクエストボディを使用した別の例を示します。

```
aws transcribe create-language-model \
--cli-input-json file://filepath/my-first-language-model.json
```

ファイル my-first-language-model.json には、次のリクエストボディが含まれます。

```
{
  "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)

この例では、AWS SDK for Python (Boto3) を使用して create_language_model メソッドで CLM を作成します。詳細については、CreateLanguageModelおよびLanguageModelを参照してください。

機能固有の例、シナリオ、クロスサービスの例など、AWS SDK を使用するその他の例については、この章を参照してください。SDK を使用した Amazon Transcribe のコード例 AWS SDKs

```
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)
```

カスタム言語モデルの更新

Amazon Transcribeカスタム言語モデルに使用できるベースモデルを継続的に更新します。これらのアップデートを活用するには、6 ~ 12 か月ごとに新しいカスタム言語モデルをトレーニングすることをお勧めします。

カスタム言語モデルが最新のベースモデルを使用しているかどうかを確認するには、AWS CLIまたは AWS SDK `DescribeLanguageModel`を使用してリクエストを実行し、`UpgradeAvailability`レスポンス内のフィールドを探します。

`UpgradeAvailability`そうである場合`true`、モデルで最新バージョンのベースモデルが実行されていないことになります。カスタム言語モデルで最新のベースモデルを使用するには、カスタム言語モデルを新規に作成する必要があります。カスタム言語モデルはアップグレードできません。

カスタム言語モデルの併用

カスタム言語モデルを作成したら、それを文字起こしリクエストに含めることができます。例については、次のセクションを参照してください。

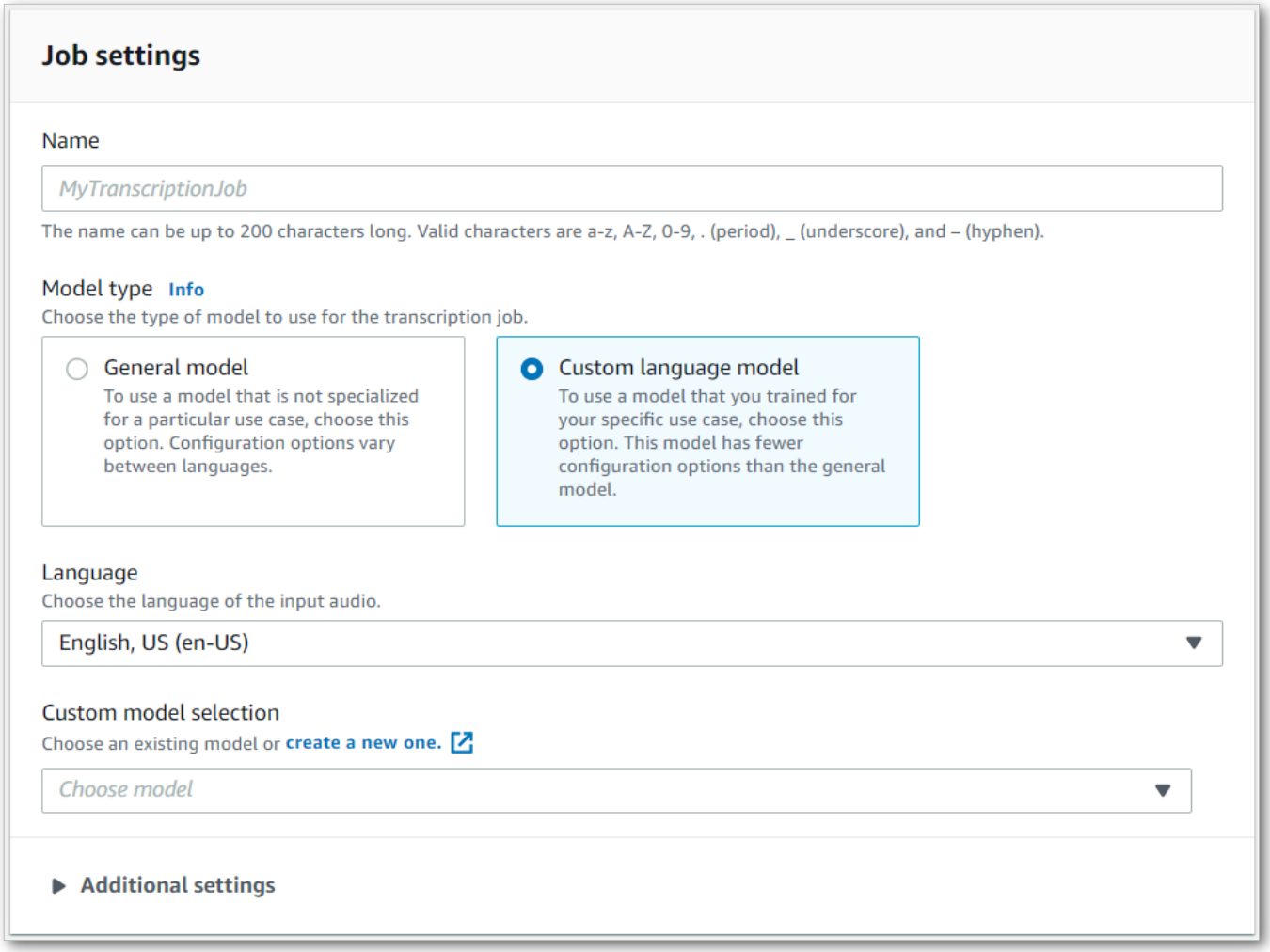
リクエストに含めるモデルの言語は、メディアに指定する言語コードと一致する必要があります。言語が一致しない場合、カスタム言語モデルは文字起こしに適用されず、警告やエラーも発生しません。

バッチ変換でカスタム言語モデルを使用する

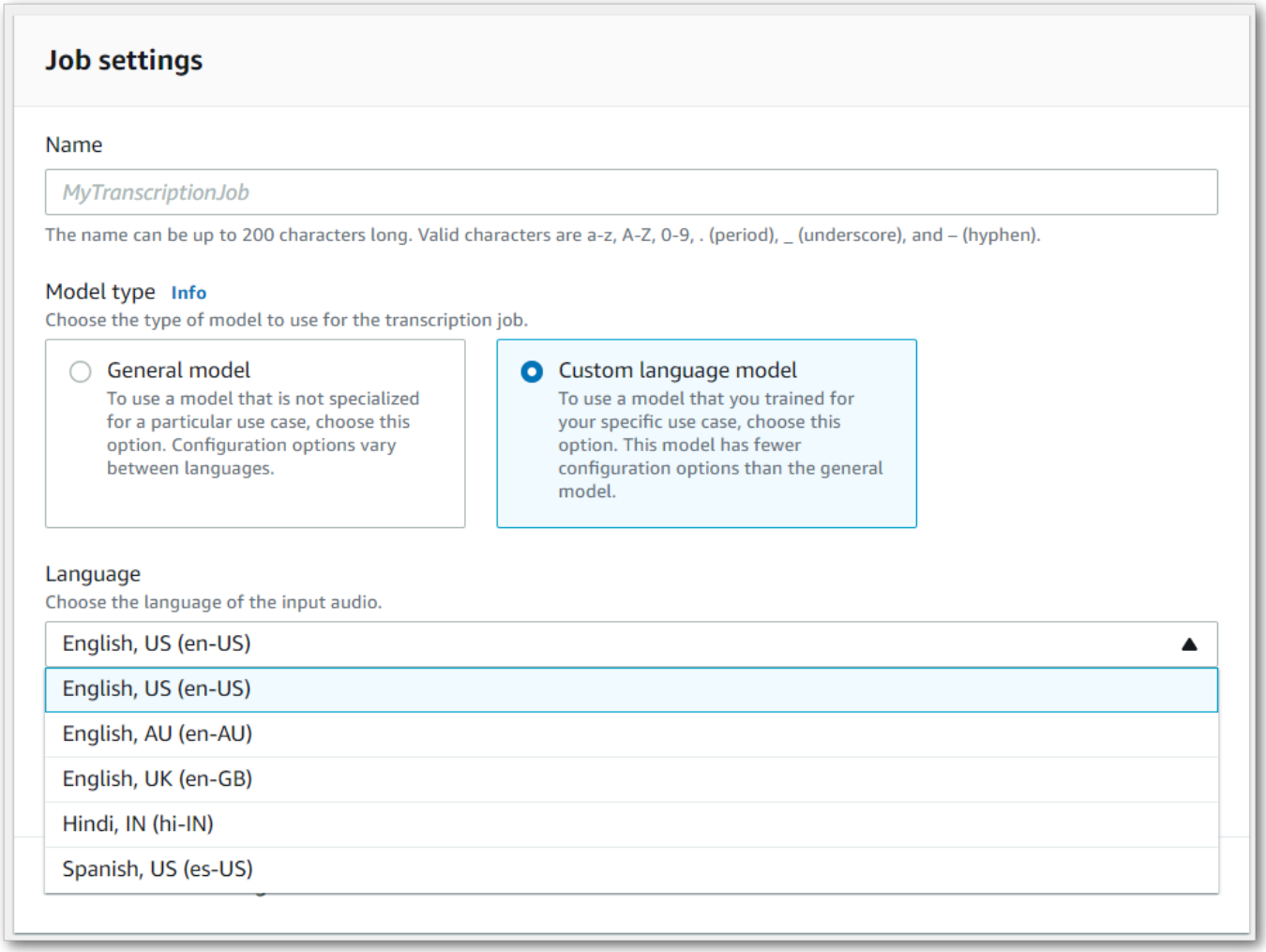
バッチ変換でカスタム言語モデルを使用するには、次の例を参照してください。

AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、変換ジョブを選択後、ジョブの作成を選択します (右上) を選択します。これにより、ジョブの詳細を指定 ページが開きます。
3. [ジョブ設定] パネルの [モデルタイプ] で、[カスタム言語モデル] ボックスを選択します。



また、ドロップダウンメニューから入力言語を選択する必要があります。



4. カスタムモデルの選択で、ドロップダウンメニューから既存のカスタム言語モデルを選択するか、新しいカスタム言語モデルを作成します。

   Amazon S3入力ファイルの場所を入力データパネルに追加します。

5. 次へを選択すると追加の設定オプションが表示されます。

   ジョブの作成を選択して、変換ジョブを実行します。

AWS CLI

この例では、start-transcription-jobModelSettingsVocabularyNameコマンドとパラメーターをサブパラメーターと共に使用しています。詳細については、StartTranscriptionJobおよびModelSettingsを参照してください。

```
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
```

別の例として、start-transcription-jobそのジョブでカスタム言語モデルを含むリクエストボディを使用します。

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

ファイル my-first-model-job.json に次のリクエストボディが入ります。

```
{
  "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)

この例では、AWS SDK for Python (Boto3) transstart_transcription_job メソッドのModelSettings引数で、カスタム言語モデルをインクルードします。詳細については、StartTranscriptionJobおよびModelSettingsを参照してください。

機能固有、シナリオ、サービス間の例など、AWS SDK を使用するその他の例については、SDK を使用した Amazon Transcribe のコード例 AWS SDKsこの章を参照してください。

```
from __future__ import print_function
import time
```

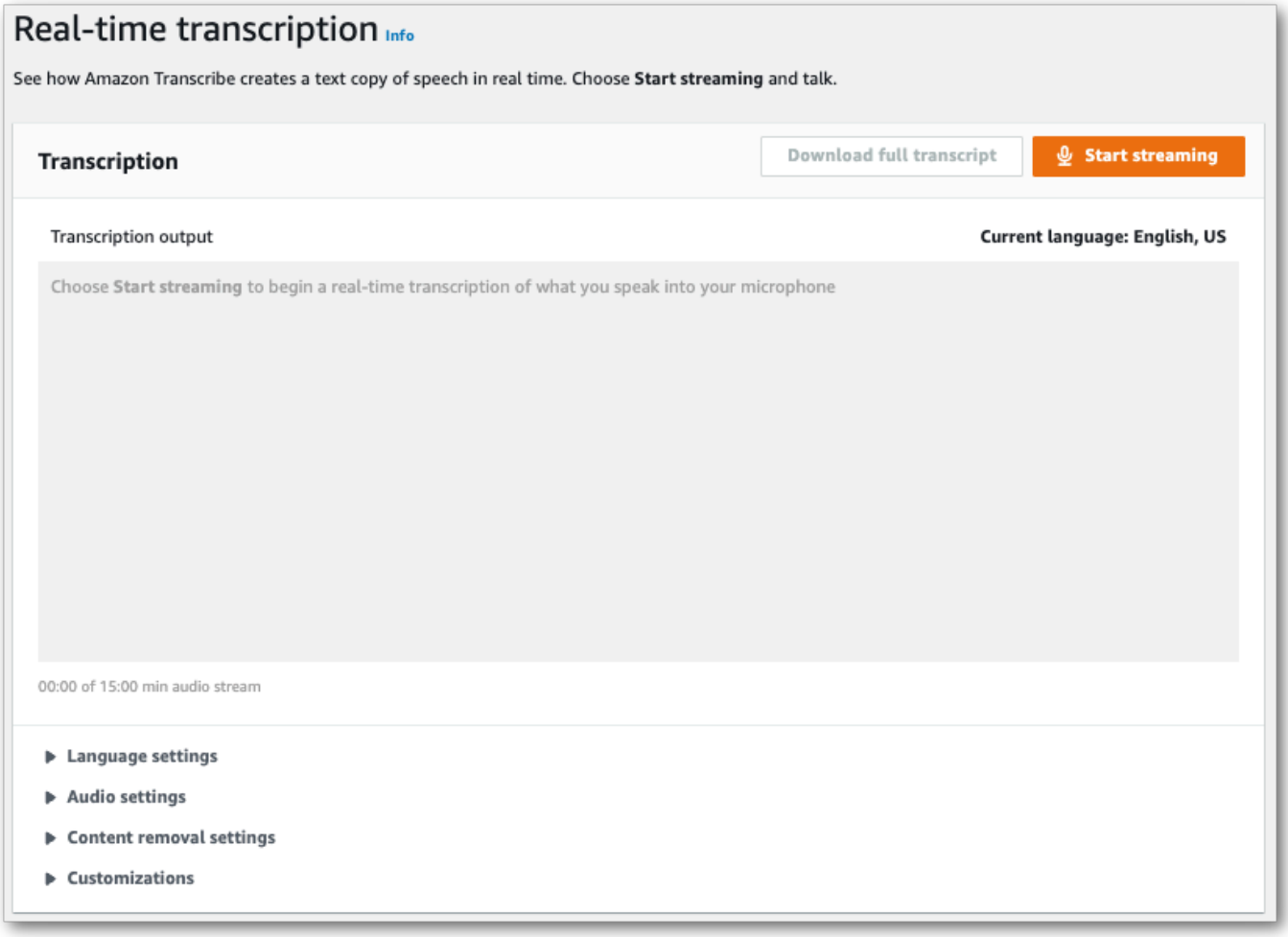
```
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    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)
```

ストリーミング変換でカスタム言語モデルを使用する

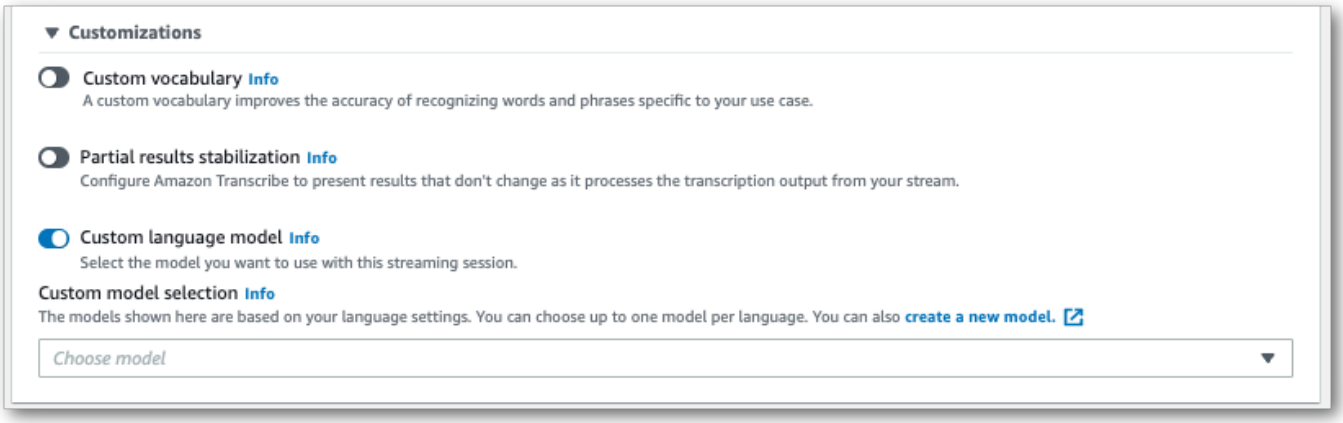
ストリーミングトランスクリプションでカスタム言語モデルを使用するには、次の例を参照してください。

AWS Management Console

1. AWS Management Console にサインインします。
2. ナビゲーションペインで、[リアルタイム文字起こし] を選択します。カスタマイズにスクロールして、最小化されている場合はこのフィールドを展開します。



3. カスタム言語モデルをオンに切り替え、ドロップダウンメニューからモデルを選択します。



ストリーミングに適用するその他の設定を追加します。

4. これで、ストリームを書き起こす準備ができました。[ストリーミングを開始] を選択し、話し始めます。ディクテーションを終了するには、「ストリーミングを停止」を選択します。

HTTP/2 ストリーミング

この例では、カスタム言語モデルを含む HTTP/2 リクエストを作成します。でHTTP/2 ストリーミングを使用する際の詳細についてはAmazon Transcribe、を参照してくださいHTTP/2 ストリームの設定。に固有のパラメータとヘッダーの詳細についてはAmazon Transcribe、を参照してください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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

WebSocket ストリーム

この例では、 WebSocket カスタム言語モデルをストリームに適用する署名付き URL を作成します。読みやすくするために、改行が追加されています。 WebSocket でのストリームの使用の詳細についてはAmazon Transcribe、を参照してください WebSocket ストリームのセットアップ。パラメータの詳細については、「StartStreamTranscription」を参照してください。

```
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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

カスタムボキャブラリーフィルターを使用して単語を削除、マスク、またはフラグを付ける

カスタムボキャブラリーフィルターは、文字起こし出力で変更したい個々の単語のカスタムリストを含むテキストファイルです。

一般的な使用例としては、攻撃的または冒涜的な用語を削除することがありますが、カスタムボキャブラリーフィルターは完全にカスタム化されているため、好きな単語を選択できます。たとえば、発売予定の新製品がある場合、会議の議事録に製品名を隠すことができます。この場合、 up-to-date 製品名は発売まで秘密にしておきながら、関係者を保護することになります。

語彙フィルタリングにはmask、、remove、の 3 つの表示方法がありますtag。次の例を参照して、それぞれの仕組みを確認してください。

- マスク:指定された単語を 3 つのアスタリスク (***) に置き換えます

```
"transcript": "You can specify a list of *** or *** words, and *** *** removes them
 from transcripts automatically."
```

- 削除:指定した単語を削除し、その場所には何も残しません。

```
"transcript": "You can specify a list of or words, and removes them from transcripts
 automatically."
```

- タグ:指定された各単語にタグ ("vocabularyFilterMatch": true) を追加しますが、単語自体は変更しません。タグ付けにより、トランスクリプトの置換や編集を迅速に行うことができます。

```
"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
```

文字起こしリクエストを送信するときに、カスタムボキャブラリーフィルターと適用するフィルター方法を指定できます。 Amazon Transcribe次に、指定したフィルター方法に従って、文字起こしに完全に一致する単語がトランスクリプトに表示されたときに修正されます。

カスタムボキャブラリーフィルターは、バッチおよびストリーミングの文字起こしリクエストに適用できます。カスタム語彙フィルタを作成する方法については、「」を参照してください語彙フィルターを作成する。カスタムボキャブラリフィルタを適用する方法については、を参照してくださいカスタムボキャブラリーフィルターを使用する。

> **Note**
>
> Amazon Transcribe人種差別に敏感な用語は自動的にマスクされますが、AWSテクニカル Support に連絡してこのデフォルトフィルターをオプトアウトできます。

ボキャブラリーフィルターのビデオチュートリアルについては、「ボキャブラリーフィルターの使用」を参照してください。

> **ボキャブラリーフィルタリングに固有の API 操作**
>
> `CreateVocabularyFilter`, `DeleteVocabularyFilter`, `GetVocabularyFilter`, `ListVocabularyFilters`, `UpdateVocabularyFilter`

語彙フィルターを作成する

カスタムボキャブラリーフィルターを作成するには、次の 2 つのオプションがあります。

1. 行で区切られた単語のリストを UTF-8 エンコーディングのプレーンテキストファイルとして保存します。
   - この方法はAWS Management Console、AWS CLI、またはAWS SDK で使用できます。
   - を使用する場合AWS Management Console、カスタムボキャブラリファイルのローカルパスまたはAmazon S3 URI を指定できます。
   - AWS CLIまたはAWS SDK を使用する場合は、Amazon S3カスタムボキャブラリファイルをバケットにアップロードし、リクエストにAmazon S3 URI を含める必要があります。
2. API リクエストに、カンマ区切りの単語のリストを直接追加します。
   - この方法は、AWS CLIまたはAWS SDK `Words`でパラメータを使用して使用できます。

各方法の例については、を参照してください。カスタム語彙フィルターを作成する

カスタムボキャブラリーフィルターを作成する際の注意点:

- 単語は、大文字と小文字が区別されません。たとえば、「呪い」と「呪い」は同じように扱われます。
- 完全に一致する単語のみがフィルタリングされます。例えば、フィルターに「swear」が含まれていても、メディアに「swears」が含まれていても「swears」が含まれていても、フィルタリングされません。「swear」のインスタンスのみがフィルタリングされます。そのため、フィルターしたい単語のバリエーションをすべて含める必要があります。
- フィルターは、他の単語に含まれる単語には適用されません。例えば、カスタム語彙フィルターに「marine」が含まれていても「submarine」が含まれていない場合、トランスクリプトの「submarine」をフィルタリングしません。
- 各エントリに含めることができる単語は 1 つだけです (スペースは不可)。
- カスタムボキャブラリフィルターをテキストファイルとして保存する場合は、UTF-8 エンコーディングのプレーンテキスト形式にする必要があります。
- 1AWS アカウント つあたり最大 100 個のカスタムボキャブラリフィルターを使用でき、それぞれのサイズは 50 KB までです。
- 使用している言語でサポートされている文字のみを使用できます。詳細については、使用する言語の文字セットを参照してください。

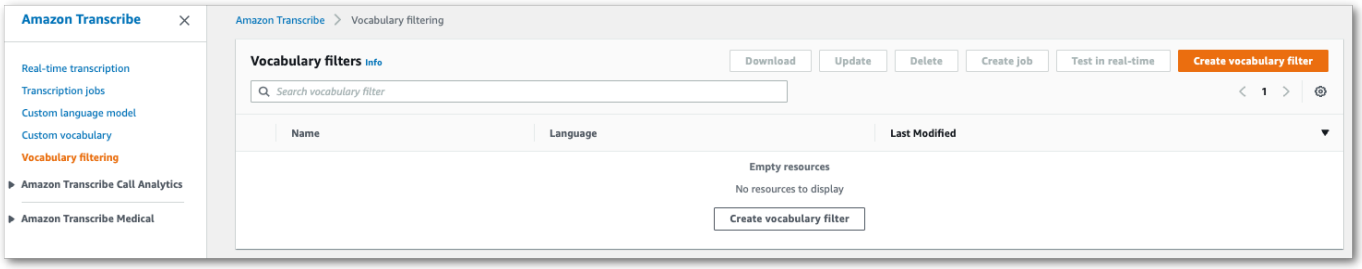
カスタム語彙フィルターを作成する

で使用するカスタムボキャブラリフィルターを処理するにはAmazon Transcribe、以下の例を参照してください。

AWS Management Console

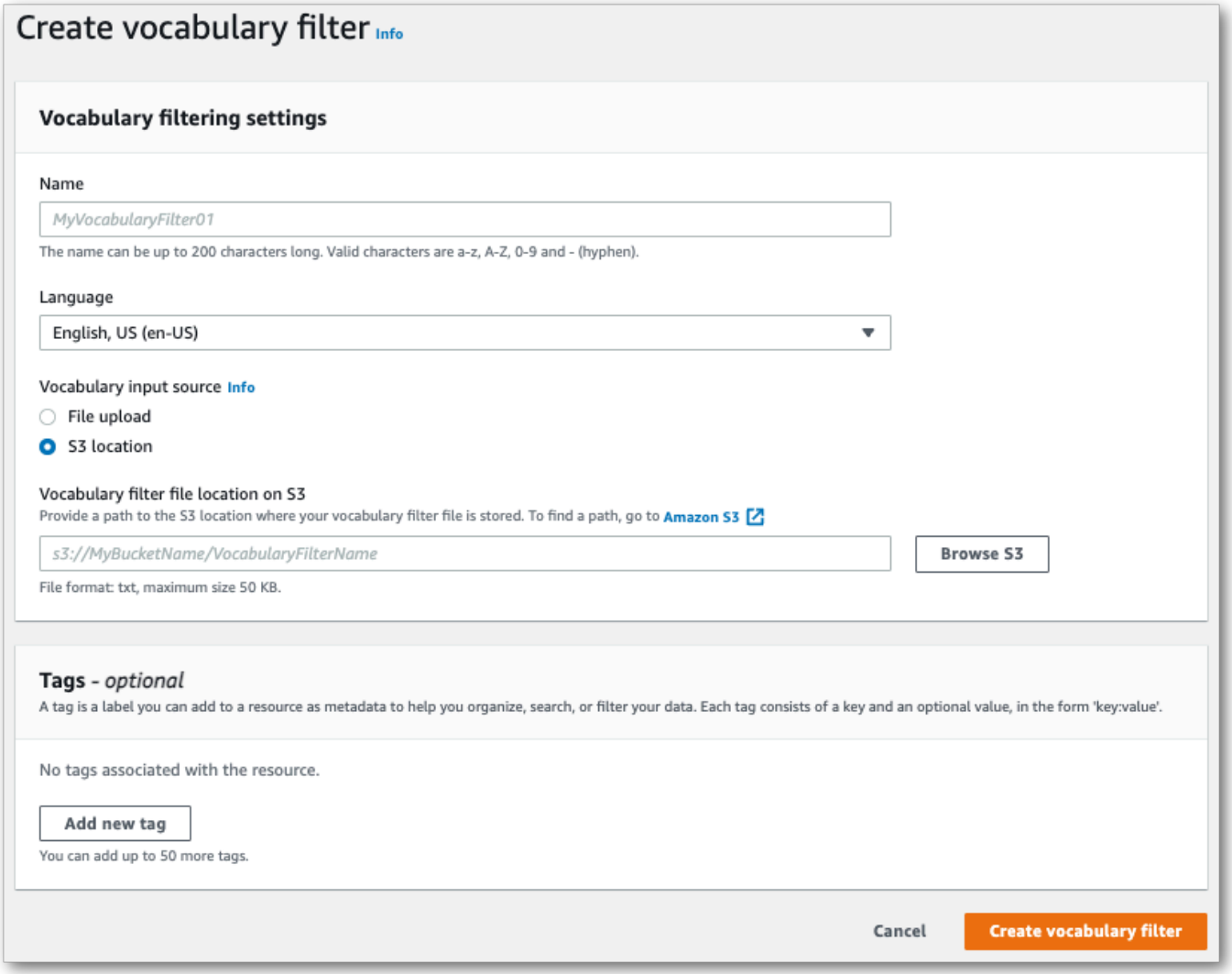
続行する前に、カスタムボキャブラリーフィルターをテキスト (*.txt) ファイルとして保存してください。Amazon S3オプションでファイルをバケットにアップロードできます。

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、[語彙フィルタリング] を選択します。これにより、「語彙フィルター」ページが開き、既存のカスタム語彙フィルターを表示したり、新しい語彙フィルターを作成したりできます。
3. [語彙フィルターを作成] を選択します。



これにより、「語彙フィルタの作成」ページが表示されます。新しいカスタム語彙フィルターの名前を入力します。

[語彙入力ソース] で [ファイルのアップロード] または [S3 の場所] オプションを選択します。次に、カスタム語彙ファイルの場所を指定します。



4. オプションで、カスタムボキャブラリーフィルターにタグを追加します。すべてのフィールドを完了したら、ページ下部の「語彙フィルターを作成する」を選択します。ファイルの処理中にエラーがなければ、「語彙フィルター」ページに戻ります。

   カスタム語彙フィルターを使用する準備ができました。

AWS CLI

この例では、create-vocabulary-filterコマンドを使用して単語リストを使用可能なカスタムボキャブラリフィルターに処理します。詳細については、「CreateVocabularyFilter」を参照してください。

オプション 1:words パラメータを使用して、リクエストに単語のリストを含めることができます。

```
aws transcribe create-vocabulary-filter \
--vocabulary-filter-name my-first-vocabulary-filter \
--language-code en-US \
--words profane,offensive,Amazon,Transcribe
```

オプション 2:Amazon S3 単語のリストをテキストファイルとして保存してバケットにアップロードし、vocabulary-filter-file-uriパラメータを使用してリクエストにファイルの URI を含めることができます。

```
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
```

次に、create-vocabulary-filterコマンドと、カスタム語彙フィルターを作成するリクエストボディを使用した別の例を示します。

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

ファイル my-first-vocab-filter.json に次のリクエストボディが入ります。

オプション 1:Words パラメータを使用して、リクエストに単語のリストを含めることができます。

```
{
```

```
  "VocabularyFilterName": "my-first-vocabulary-filter",
  "LanguageCode": "en-US",
  "Words": [
        "profane","offensive","Amazon","Transcribe"
  ]
}
```

オプション 2:Amazon S3 単語のリストをテキストファイルとして保存してバケットにアップロードし、VocabularyFilterFileUriパラメータを使用してリクエストにファイルの URI を含めることができます。

```
{
  "VocabularyFilterName": "my-first-vocabulary-filter",
  "LanguageCode": "en-US",
  "VocabularyFilterFileUri": "s3://DOC-EXAMPLE-BUCKET/my-vocabulary-filters/my-
vocabulary-filter.txt"
}
```

> **Note**
>
> VocabularyFilterFileUriリクエストに含めると、使用できませんWords。どちらか一方を選択する必要があります。

AWS SDK for Python (Boto3)

この例では、AWS SDK for Python (Boto3)を使用して create_vocabulary ary_filter メソッドを使用してカスタムボキャブラリフィルターを作成しています。詳細については、「CreateVocabularyFilter」を参照してください。

機能固有、シナリオ、サービス間の例など、AWS SDK を使用するその他の例については、SDK を使用した Amazon Transcribe のコード例 AWS SDKsこの章を参照してください。

オプション 1:Words パラメータを使用して、リクエストに単語のリストを含めることができます。

```
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'
    ]
)
```

オプション 2:Amazon S3 単語のリストをテキストファイルとして保存してバケットにアップロードし、VocabularyFilterFileUriパラメータを使用してリクエストにファイルの URI を含めることができます。

```
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**
>
> VocabularyFilterFileUriリクエストに含めると、使用できませんWords。どちらか一方を選択する必要があります。

> **Note**
>
> Amazon S3カスタムボキャブラリーフィルターファイル用に新しいバケットを作成する場合は、IAMCreateVocabularyFilterリクエストを行うロールにこのバケットにアクセスする権限があることを確認してください。ロールに正しいアクセス許可が含まれていない場合、リクエストは失敗します。DataAccessRoleArnパラメータを含めることで、IAMリクエスト内のロールを任意で指定できます。IAMのロールとポリシーの詳細についてはAmazon Transcribe、を参照してくださいAmazon Transcribe アイデンティティベースポリシーの例。

カスタムボキャブラリーフィルターを使用する

カスタムボキャブラリーフィルターを作成したら、それを文字起こしリクエストに含めることができます。例については、次のセクションを参照してください。

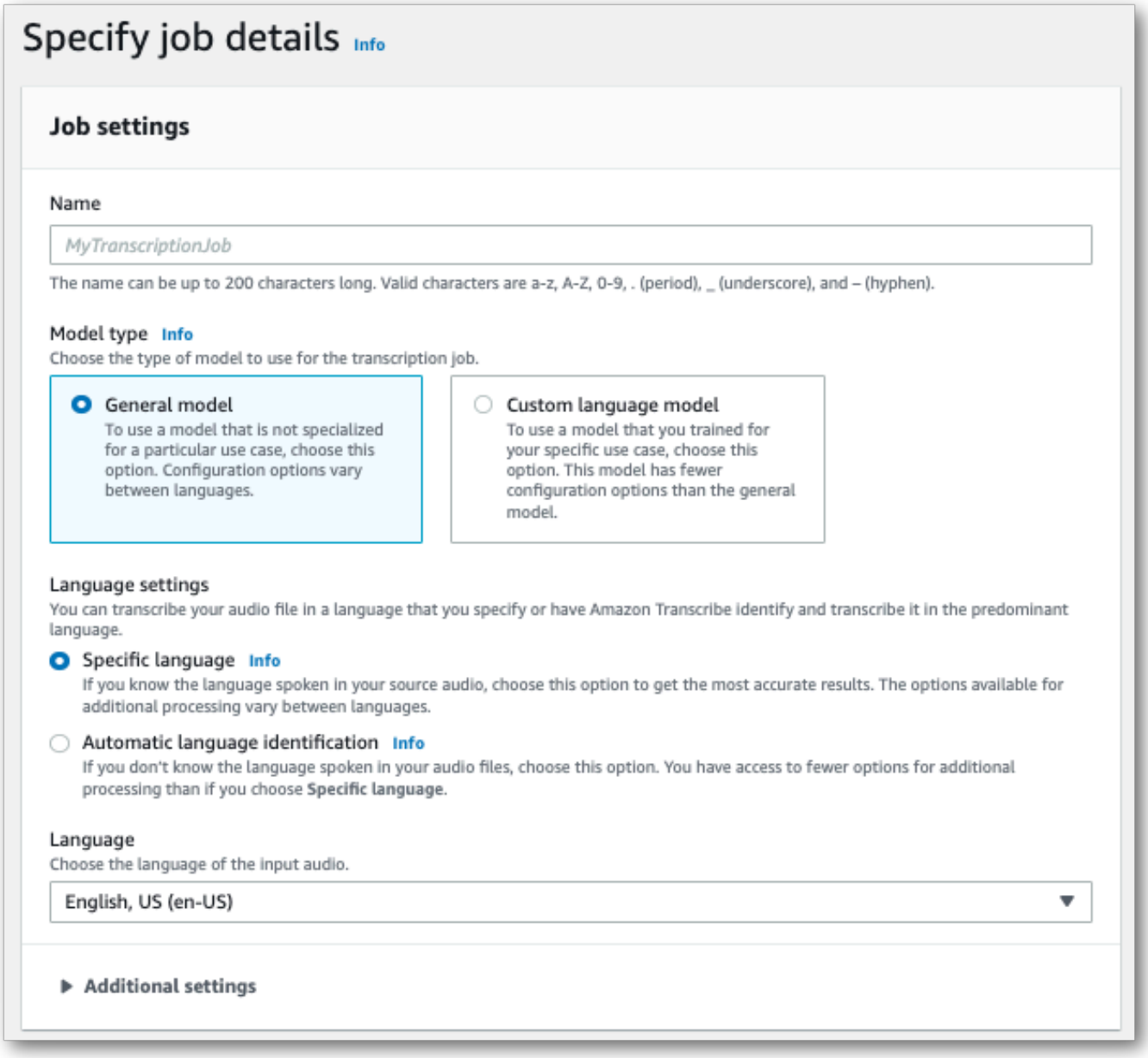
リクエストに含めるカスタムボキャブラリーフィルターの言語は、メディアに指定する言語コードと一致する必要があります。言語識別を使用して複数の言語オプションを指定する場合、指定した言語ごとに 1 つのカスタム語彙フィルターを含めることができます。カスタムボキャブラリーフィルターの言語がオーディオで指定された言語と一致しない場合、フィルターはトランスクリプションに適用されず、警告やエラーもありません。

バッチトランスクリプションでのカスタムボキャブラリーフィルターの使用

バッチトランスクリプションでカスタムボキャブラリーフィルターを使用するには、次の例を参照してください。

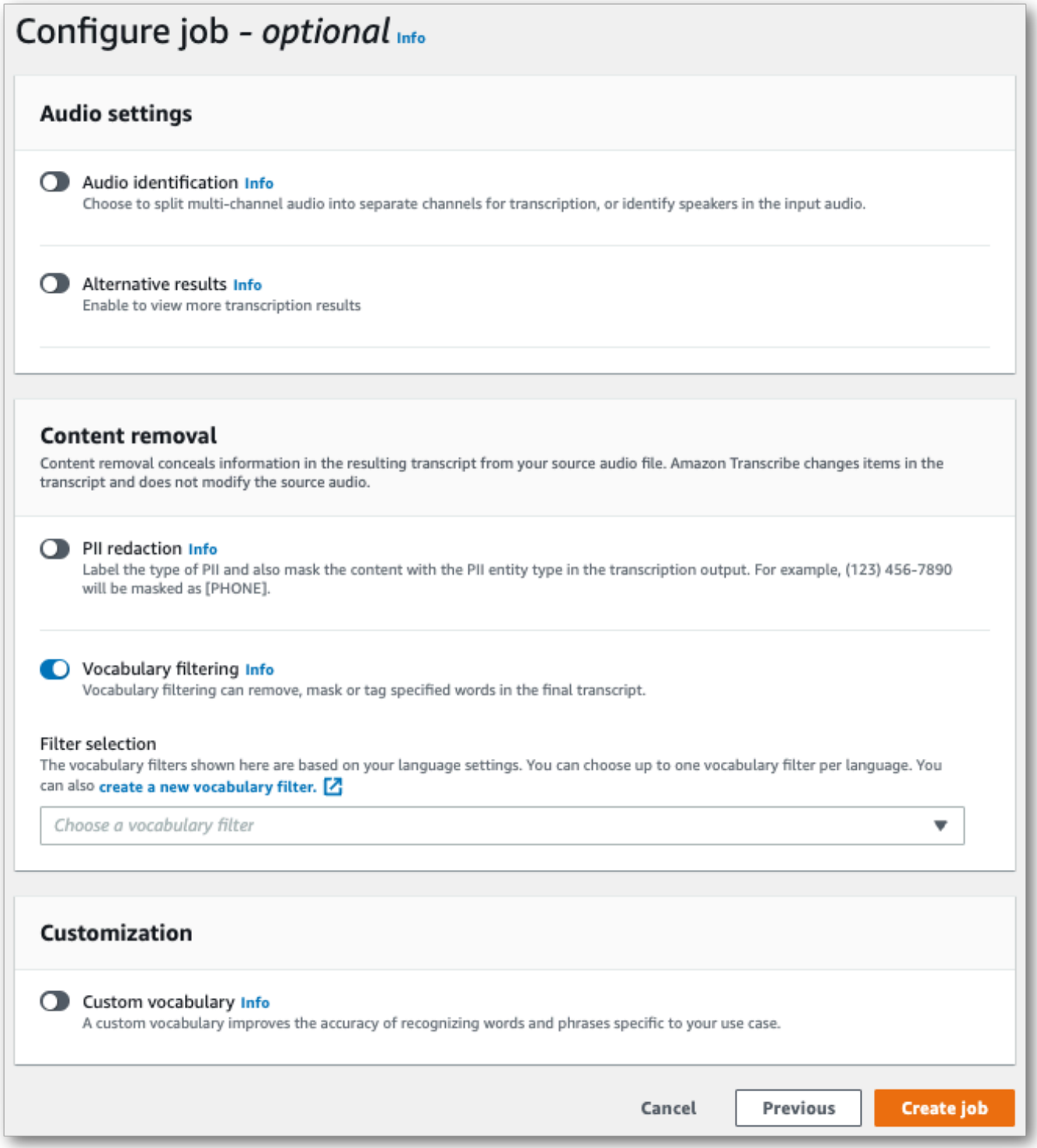
AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、次にジョブの作成を選択します。ジョブの作成 (右上) を選択します。これにより、ジョブの詳細を指定 ページが開きます。

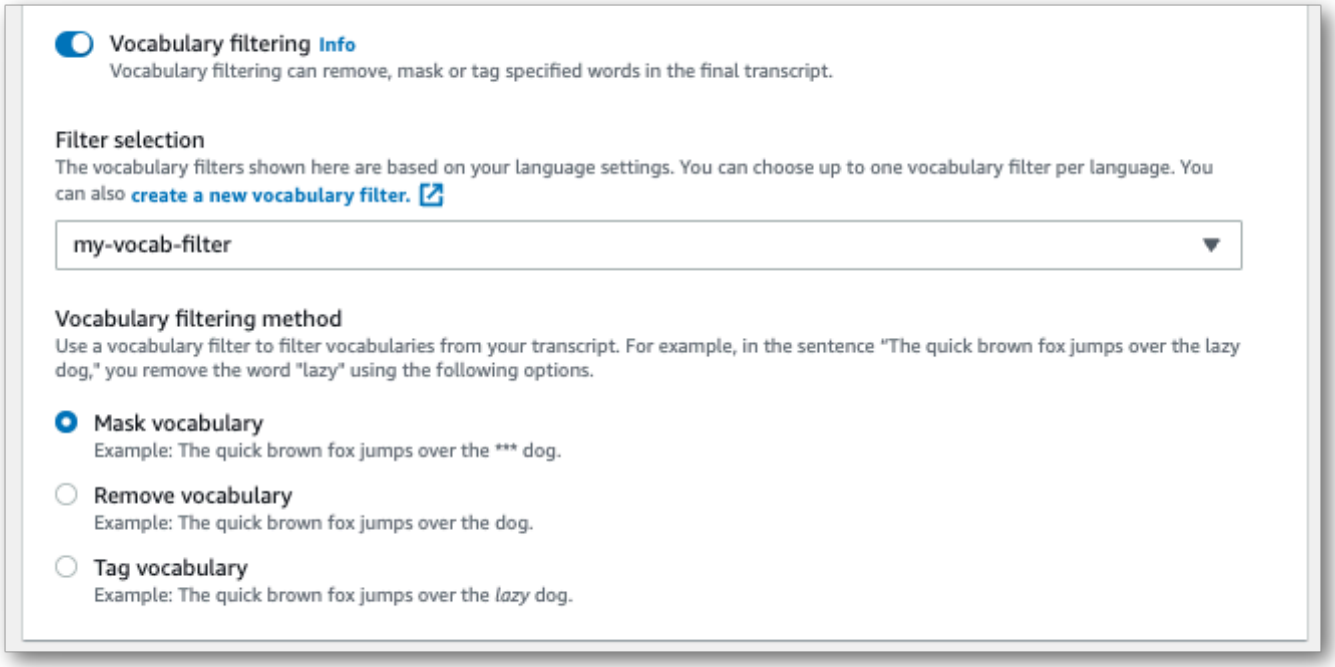


ジョブに名前を付け、入力メディアを指定します。オプションで他のフィールドを追加して、[次へ] を選択します。

3. [ジョブの設定] ページのコンテンツ削除パネルで、[語彙フィルター] をオンに切り替えます。



4. ドロップダウンメニューからカスタムボキャブラリーフィルターを選択し、フィルター方法を指定します。



5. ジョブの作成を選択し、書き起こします。

AWS CLI

この例では、start-transcription-jobSettingsVocabularyFilterNameVocabularyFilterMethodコマンドとパラメーターをおよびサブパラメーターと共に使用しています。詳細については、StartTranscriptionJobおよびSettingsを参照してください。

```
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
```

別の例 start-transcription-job

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

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

ファイル my-first-vocabulary-filter-job.json には次のリクエストボディが含まれます。

```
{
  "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)

この例では、AWS SDK for Python (Boto3) transstart_transcription_job メソッドのSettings引数で、使用します。詳細については、StartTranscriptionJobおよびSettingsを参照してください。

機能固有、シナリオ、サービス間の例など、AWS SDK を使用するその他の例については、SDK を使用した Amazon Transcribe のコード例 AWS SDKsこの章を参照してください。

```
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 = {
```

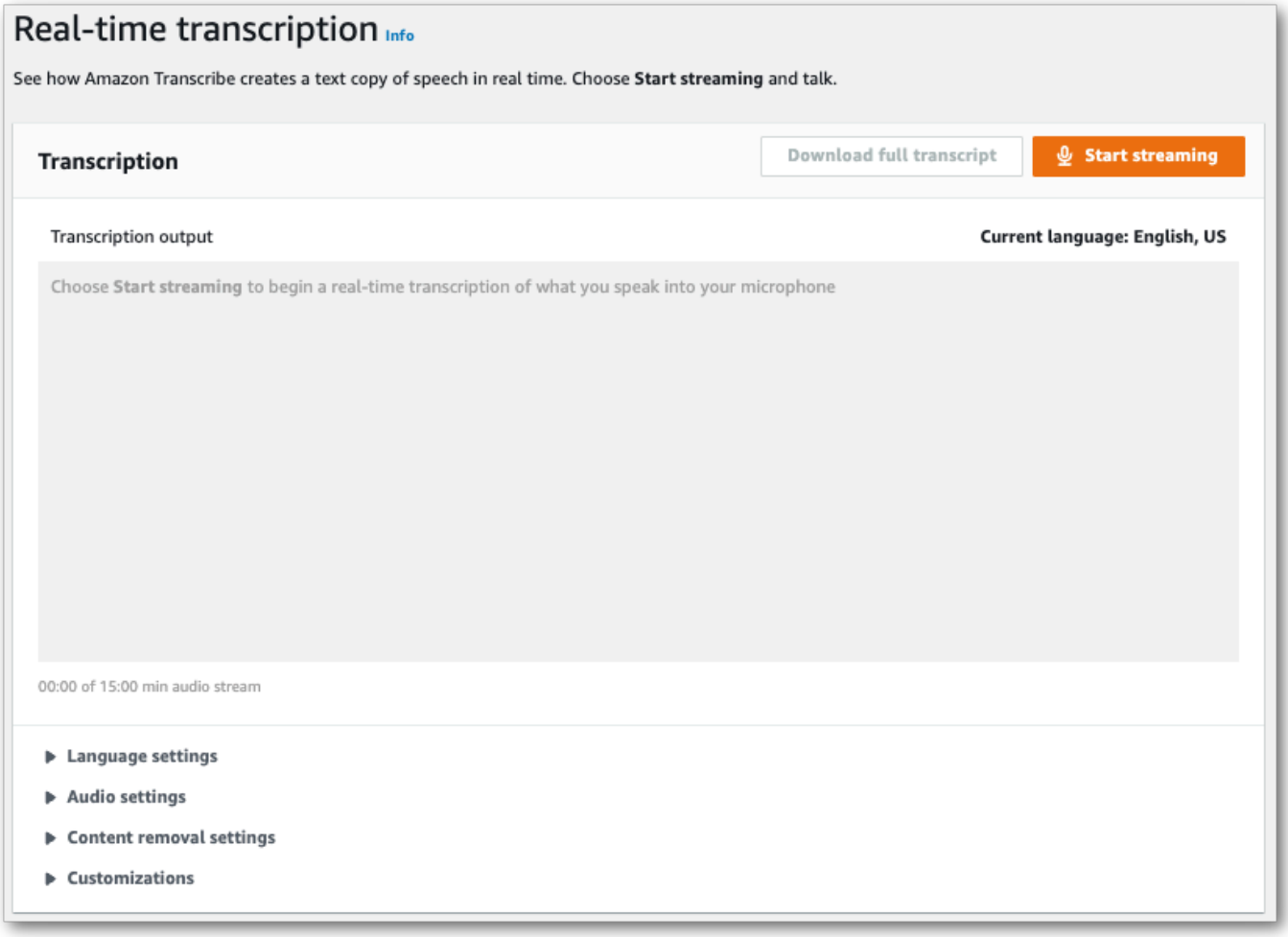
```
        'VocabularyFilterName': 'my-first-vocabulary-filter',
        'VocabularyFilterMethod': 'mask'
    }
)

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)
```

ストリーミングトランスクリプションでのカスタムボキャブラリーフィルターの使用

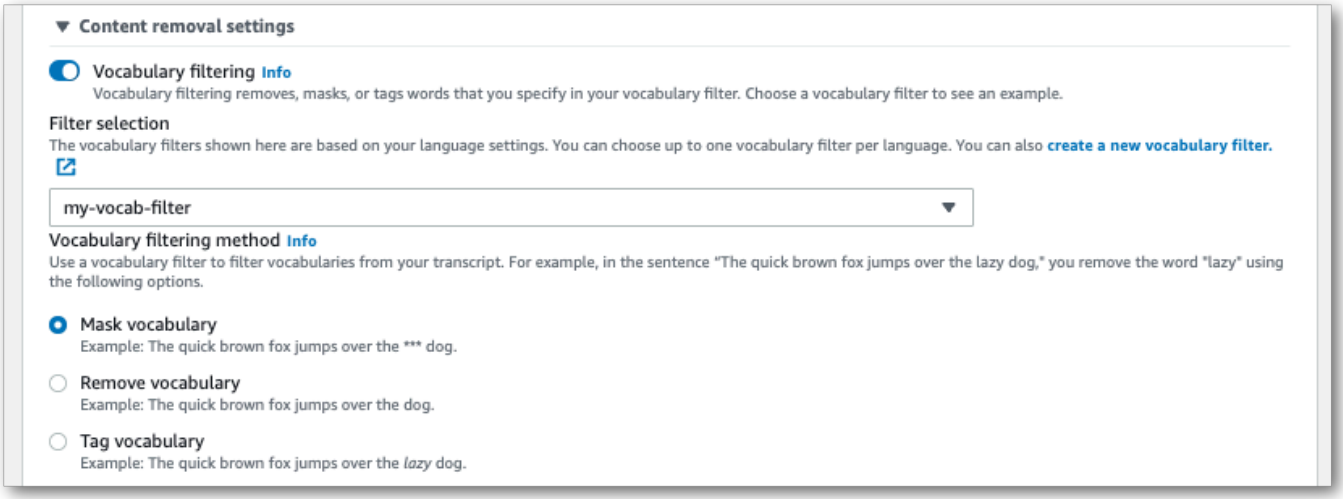
ストリーミング文字起こしでカスタムボキャブラリーフィルターを使用するには、次の例を参照してください。

AWS Management Console

1. AWS Management Console にサインインします。
2. ナビゲーションペインで、[リアルタイム文字起こし] を選択します。コンテンツ削除設定にスクロールして、最小化されている場合はこのフィールドを展開します。



3. 語彙フィルタリングをオンに切り替えます。ドロップダウンメニューからカスタムボキャブラリフィルターを選択し、フィルター方法を指定します。



ストリームに適用するその他の設定を含めます。

4. これで、ストリームを書き起こす準備ができました。[ストリーミングを開始] を選択し、話し始めます。ディクテーションを終了するには、「ストリーミングを停止」を選択します。

HTTP/2 ストリーミング

この例では、カスタム語彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙彙 HTTP/2 ストリーミングを使用する際の詳細についてはAmazon Transcribe、を参照してくださいHTTP/2 ストリームの設定。固有のパラメータとヘッダーの詳細についてはAmazon Transcribe、を参照してください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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

WebSocket ストリーム

この例では、 WebSocket カスタムボキャブラリフィルターをストリームに適用する署名付き URL を作成します。読みやすくするために、改行が追加されています。 WebSocket でのストリームの使用の詳細についてはAmazon Transcribe、を参照してください WebSocket ストリームのセットアップ。パラメータの詳細については、「StartStreamTranscription」を参照してください。

```
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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

有害な発話の検知

有毒な音声検出は、関与するソーシャルメディアプラットフォームをモデレートするのに役立つように設計されていますpeer-to-peerオンラインゲームやソーシャルチャットプラットフォームなどの対話。毒性のある言葉の使用は、個人、仲間グループ、コミュニティに深刻な悪影響を及ぼす可能性があります。有害な言葉にフラグを付けることで、組織は会話を礼儀正しく保ち、ユーザーが自由に作成、共有、参加できる安全で包括的なオンライン環境を維持できます。

Amazon TranscribeToxicity Detectionは、音声とテキストの両方の手がかりを活用して、性的嫌がらせ、ヘイトスピーチ、脅迫、虐待、冒涜、侮辱、グラフィックを含む7つのカテゴリにわたって、音声ベースの有害コンテンツを識別して分類します。テキストに加えて、Amazon Transcribe毒性検出では、トーンやピッチなどの音声合図を使用して、発話に含まれる有害な意図に焦点を合わせます。これは、意図を考慮せずに特定の用語のみに焦点を当てるように設計された標準的なコンテンツ管理システムからの改善点です。

Amazon Transcribe有害な発言にフラグを付けて分類することで、手動で処理しなければならないデータ量を最小限に抑えます。これにより、コンテンツモデレーターはプラットフォーム上の談話を迅速かつ効率的に管理できます。

有害な発話のカテゴリには以下が含まれます。

- 冒とく的な言葉: 失礼な、下品な、または攻撃的な言葉、フレーズ、または略語を含むスピーチ。
- ヘイトスピーチ: アイデンティティ（人種、民族、性別、宗教、性的指向、能力、出身国など）に基づいて個人やグループを批判、侮辱、非人間化する言葉。
- 性的: 身体の部位、身体的特徴、性別への直接的または間接的な言及を使用して、性的興味、活動、または覚醒を示す発言。
- 侮辱: 品位を落とす、屈辱的な、嘲笑する、侮辱する、または軽蔑的な言葉を含む言葉。この種の言葉はいじめとも呼ばれます。
- 暴力または脅迫: 個人や団体に苦痛、傷害、敵意を与えるような脅迫を含む言葉。
- グラフィック: 視覚的にわかりやすく、不快なほど鮮明な画像を使用するスピーチ。この種の言葉は、受信者の不快感を増幅させるために意図的に冗長化されていることがよくあります。
- 嫌がらせまたは虐待: 侮辱的または客観的な表現を含む、受取人の心理的健康に影響を及ぼすことを意図した発言 この種の言葉は嫌がらせとも呼ばれます。

毒性検出は、音声セグメント（自然な休止間の発話）を分析し、これらのセグメントに信頼度スコアを割り当てます。信頼度スコアは 0 から 1 の間の値です。信頼度スコアが高いほど、その内容が関

連力テゴリの有害な表現である可能性が高くなります。これらの信頼度スコアを使用して、ユースケースに適した毒性検出の閾値を設定できます。

> **Note**
>
> 毒性検出は、米国英語でのバッチ転写でのみ利用可能です(en-US)。

[ビュー]出力例JSON フォーマットで。

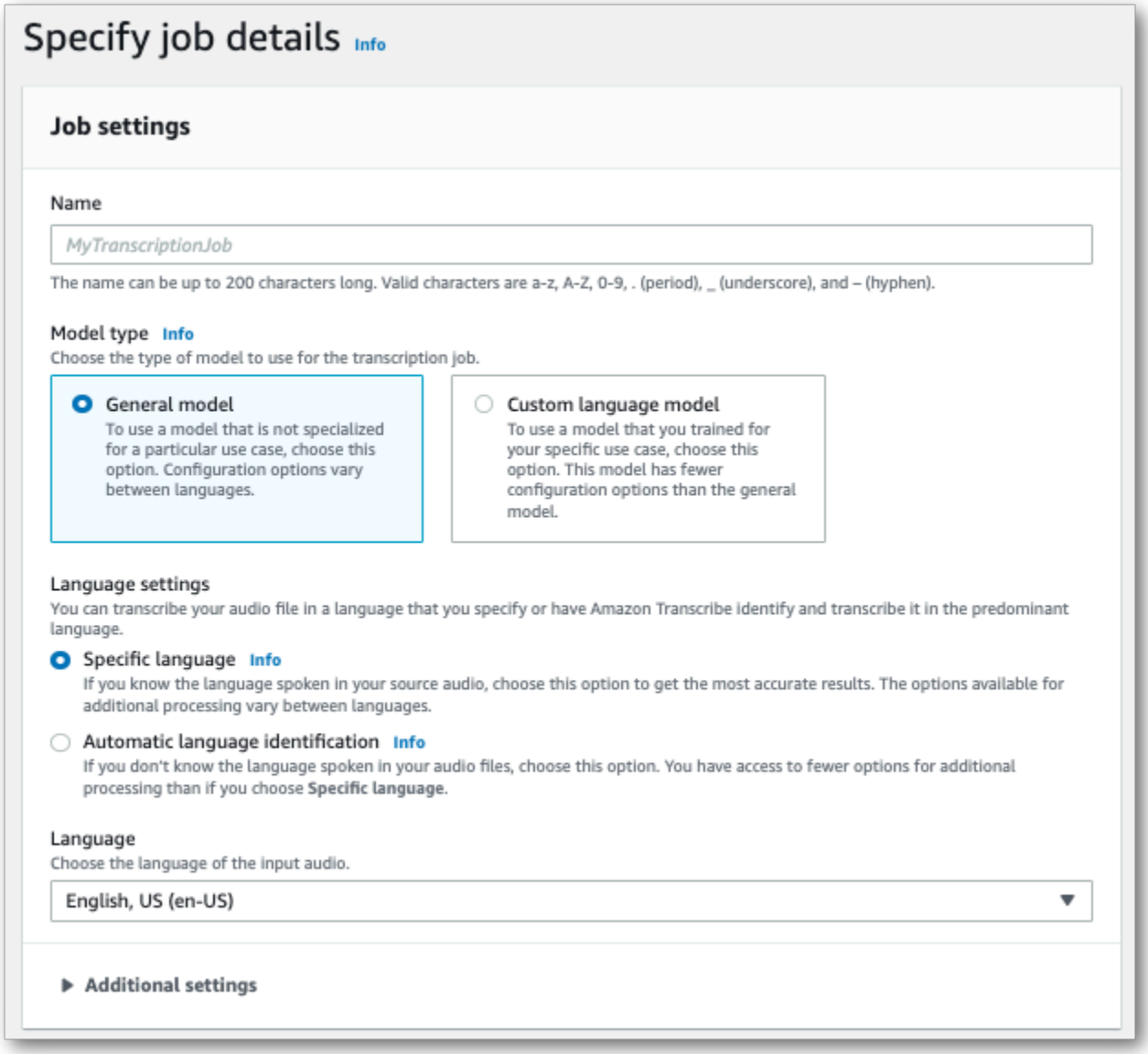
有害音声検出機能の使用

一括書き起こしでの有害音声検出機能の使用

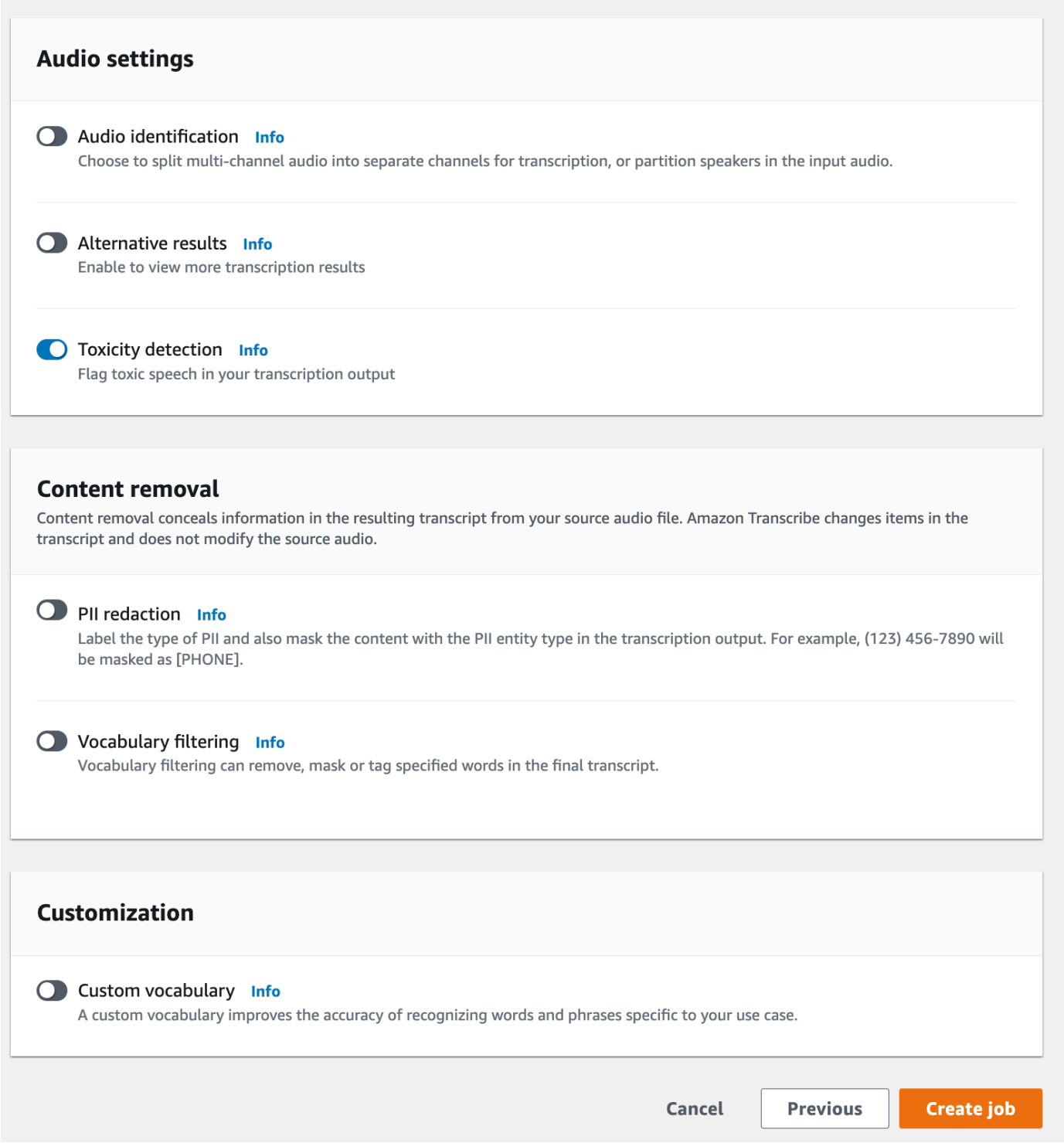
一括書き起こしで有害音声検出機能を使用するには、以下の例を参照してください。

AWS Management Console

1. AWS Management Console にサインインします。
2. ナビゲーションペインで、トランスクリプションの求人をクリックし、ジョブの作成(右上)。これにより、ジョブの詳細を指定 ページが開きます。



3. に仕事の詳細を指定ページ、必要に応じてPIIリダクションを有効にすることもできます。記載されている他のオプションは毒性検出ではサポートされていないことに注意してください。[Next] (次へ) を選択します。これにより、ジョブの設定-オプションページ。にオーディオ設定パネル、選択毒性検出。



4. [選択]ジョブの作成トランスクリプションジョブを実行します。
5. 文字起こしジョブが完了したら、から文字起こしをダウンロードできます[ダウンロード]トランスクリプションジョブの詳細ページのドロップダウンメニュー。

AWS CLI

この例では、start-transcription-jobコマンドとToxicityDetectionパラメーター。詳細については、「StartTranscriptionJob」と「ToxicityDetection」を参照してください。

```
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
```

次は、を使用した別の例ですstart-transcription-jobコマンド、および毒性検出を含むリクエストボディ。

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

ファイルmy-first-toxicity-job.jsonには、次のリクエストボディが含まれます。

```
{
  "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)

この例では、AWS SDK for Python (Boto3)有効にするToxicityDetectionのためのトランスクリプションジョブの開始方法。詳細については、「StartTranscriptionJob」と「ToxicityDetection」を参照してください。

その他の使用例については、AWS機能別、シナリオ、クロスサービスの例を含む SDK については、SDK を使用した Amazon Transcribe のコード例 AWS SDKs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)
```

出力例

有害な音声はタグ付けされ、トランスクリプション出力で分類されます。有害な発話はそれぞれ分類され、信頼度スコア (0 ~ 1 の値) が割り当てられます。信頼値が大きいほど、その内容が指定されたカテゴリ内の有害な表現である可能性が高くなります。

出力例 (JSON)

以下は JSON 形式の出力例で、分類された不適切な表現とそれに関連する信頼度スコアを示しています。


```
{
    "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"
}
```

個人を特定できる情報の編集または特定

リダクションは、トランスクリプトから個人を特定できる情報 (PII) という形で、機密性の高いコンテンツをマスキングまたは消去するために使用されます。Amazon Transcribeリダクションできる PII の種類は、バッチ文字起こしとストリーミング文字起こしによって異なります。各トランスクリプション方法の PII リストを確認するには、「」バッチジョブで PII を編集する と「」を参照してくださいリアルタイムストリームの PII の編集または識別。ストリーミングトランスクリプションを使用すれば、PII を編集せずにフラグを立てるオプションもあります。PII 識別の出力例出力例については、を参照してください:。

リダクションが有効になっている場合、編集済みのトランスクリプトのみを生成するか、または編集済みのトランスクリプトと未編集のトランスクリプトの両方を生成するオプションがあります。編集したトランスクリプトのみを生成することを選択した場合は、会話全体の保存先がメディアだけであることに注意してください。元のメディアを削除した場合は、未編集の PII の記録は残りません。このため、編集されたトランスクリプトに加えて、未編集のトランスクリプトを生成することが賢明な場合があります。

バッチトランスクリプションによる PII リダクションの詳細については、以下を参照してください:バッチジョブで PII を編集する。

ストリーミングトランスクリプションによる PII リダクションまたは識別の詳細については、以下を参照してください:リアルタイムストリームの PII の編集または識別。

> **Important**
>
> 秘匿化機能は、機密データを識別して削除するように設計されています。ただし、Amazon Transcribe機械学習が持つ予測的な性質の関係上、トランスクリプト内の機密データの存在を、すべて特定して削除することはできません。出力が要求通りに秘匿化されていることを、自分自身で再確認することを強くお勧めします。
> 秘匿化機能は、1996 年に米国で制定された、医療保険の相互運用性と説明責任に関する法律 (HIPAA) 等の医療プライバシー法に基づく匿名性の要件を満たすものではありません。

Amazon Transcribeの編集機能のビデオチュートリアルについては、「コンテンツ編集による個人識別情報の特定と編集」 を参照してください。

バッチジョブで PII を編集する

バッチ文字起こしジョブ中にトランスクリプトから個人を特定できる情報 (PII) を編集すると、 は、識別された PII の各インスタンスをトランスクリプトの本文[PII]の Amazon Transcribe に置き換えます。文字起こし出力の word-for-word 部分で編集された PII のタイプを表示することもできます。出力サンプルについては、「編集後の出力例 (バッチ)」を参照してください。

バッチ文字起こしによる編集は、米国英語 (en-US) と米国スペイン語 () で利用できますes-US。リダクションは言語識別と互換性がありません。

編集済みトランスクリプトと未編集トランスクリプトの両方が同じ出力 Amazon S3 バケットに保存されます。 Amazon Transcribe は、指定したバケット、またはサービスによって管理されるデフォルト Amazon S3 バケットにトランスクリプトを保存します。

バッチ文字起こしで認識 Amazon Transcribe できる PII のタイプ

PII タイプ	説明
ADDRESS	実際の住所、米国、エニータウン市。メインストリート 100 番地や、ビル 123 番、スイート 12 番など。住所には、通り、ビル、場所、市区町村、州、国、郡、郵便番号、管区、近隣などを含めることができます。
ALL	この表に記載されているすべての PII のタイプを編集または特定します。
BANK_ACCOUNT_NUMBER	米国の銀行口座番号 この番号は通常 10～12 桁の長さですが、 Amazon Transcribe は下 4 桁のみの銀行口座番号も認識します。
BANK_ROUTING	米国の銀行口座の支店コード この番号は通常 9 桁の長さですが、 Amazon Transcribe は下 4 桁のみの支店コードも認識します。
CREDIT_DEBIT_CVV	VISA に存在する 3 桁のカード検証コード (CVV) MasterCard、Discover クレジットカードとデビットカード。American Express のク

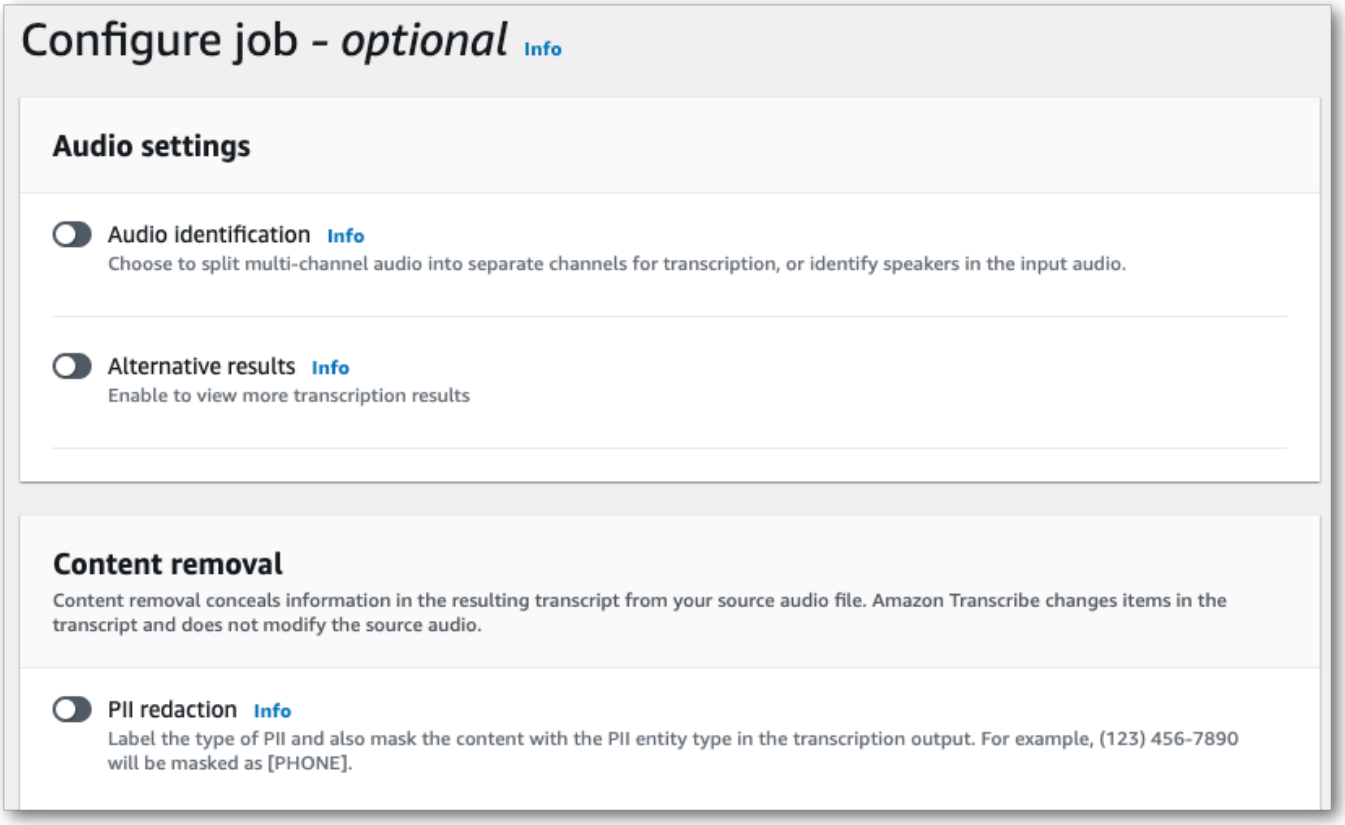
PII タイプ	説明
	レジットカードまたはデビットカードでは、4 桁の数字コードです。
CREDIT_DEBIT_EXPIRY	クレジットカードまたはデビットカードの有効期限日 この番号は通常 4 桁で、「月/年」または「MM/YY」という形式になっています。例えば、 は 01/21、01/2021、Jan 2021 などの有効期限を認識 Amazon Transcribe できます。
CREDIT_DEBIT_NUMBER	クレジットカードまたはデビットカードの番号。これらの番号の長さは 13 桁から 16 桁までさまざまですが、最後の 4 桁のみが存在する場合はクレジットカード番号またはデビットカード番号 Amazon Transcribe も認識されます。
EMAIL	efua.owusu@email.com などのメールアドレス。
NAME	個人の名前。このエンティティタイプには、Mr.、Mrs.、Miss、Dr. Amazon Transcribe does などのタイトルは含まれません。このエンティティタイプは、組織または住所の一部である名前には適用されません。例えば、は John Doe Organization を組織として認識し、Jane Doe Street を住所として Amazon Transcribe 認識します。
PHONE	電話番号 このエンティティタイプには、ファックス番号とポケットベル番号も含まれます。
PIN	銀行口座情報へのアクセスを可能にする 4 桁の個人識別番号 (PIN)。

PII タイプ	説明
SSN	社会保障番号 (SSN) は、米国市民、永住者、および一時的な労働居住者に発行される 9 桁の番号です。 は、最後の 4 桁のみが存在する場合に社会保障番号 Amazon Transcribe も認識します。

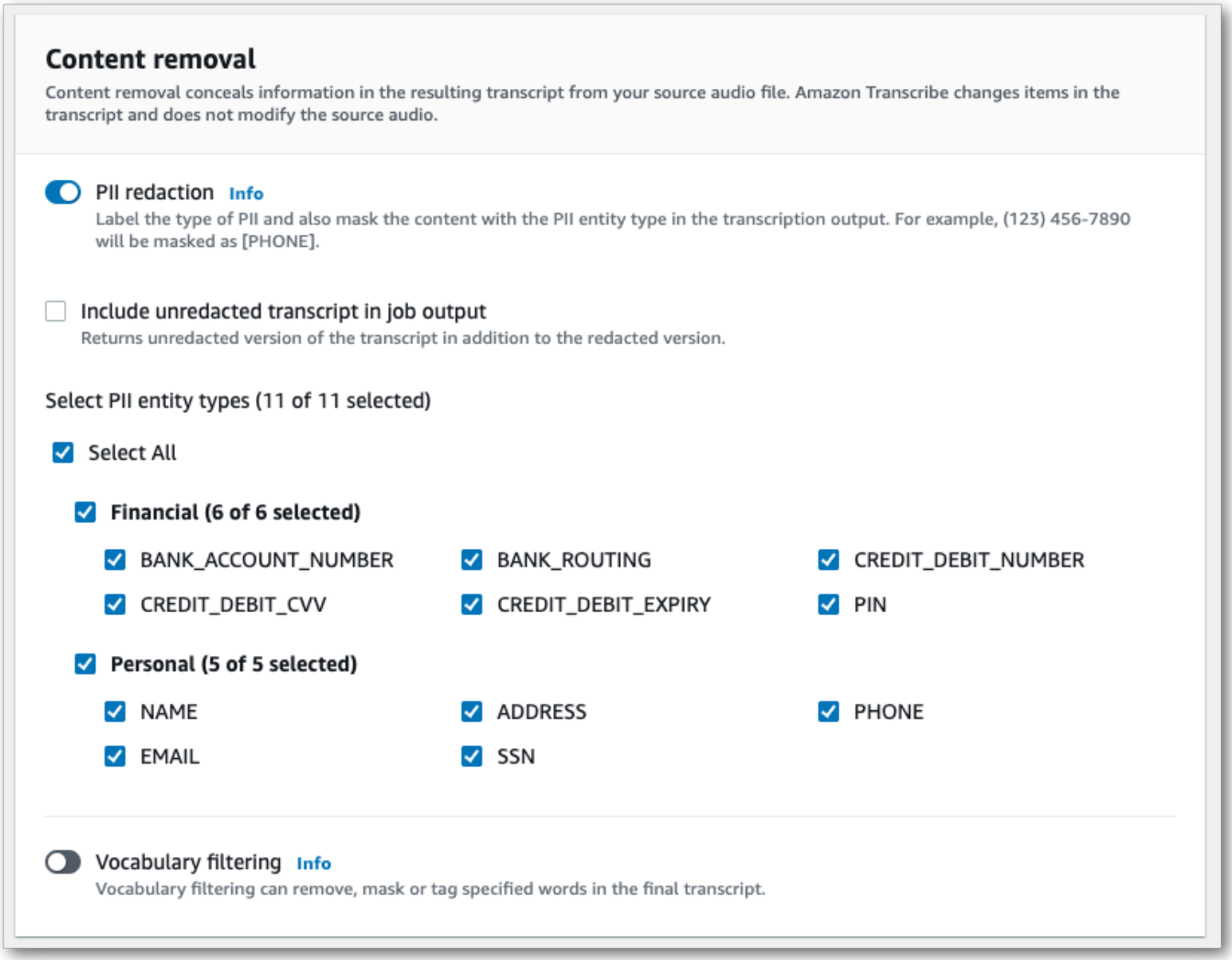
バッチ文字起こしジョブは AWS Management Console、、 AWS CLI、または AWS SDK を使用して開始できます。

AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、[文字起こしジョブ] を選択後、[ジョブの作成] (右上) を選択します。これにより、ジョブの詳細を指定 ページが開きます。
3. ジョブの詳細を指定する ページで必要な項目を入力したら、[次へ] を選択して、ジョブの設定 - オプション ページに進みます。ここには PII リダクション切り替えが付いたコンテンツ削除パネルがあります。



4. [PII リダクション] を選択すると、編集したいすべての PII タイプを選択するオプションがあります。「未編集のトランスクリプトをジョブ出力に含める」ボックスを選択した場合は、未編集のトランスクリプトを選択することもできます。



5. [ジョブの作成] を選択して、文字起こしジョブを実行します。

AWS CLI

この例では、 start-transcription-job コマンドと content-redactionパラメータを使用します。詳細については、「StartTranscriptionJob」および「ContentRedaction」を参照してください。

```
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
```

start-transcription-job メソッドを使用した別の例を次に示します。リクエストボディはそのジョブの PII を編集します。

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

ファイル my-first-redaction-job.json には、次のリクエスト本文が含まれています。

```
{
  "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)

この例では、 を使用して AWS SDK for Python (Boto3) 、start_transcription_job メソッドの ContentRedaction引数を使用してコンテンツを編集します。詳細については、「StartTranscriptionJob」および「ContentRedaction」を参照してください。

機能固有の例、シナリオ例、クロスサービス例など、 AWS SDKs「」のSDK を使用した Amazon Transcribe のコード例 AWS SDKs章を参照してください。

```
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**
>
> バッチジョブの PII リダクションは、次の でのみサポートされています AWS リージョン。アジアパシフィック (香港) 、アジアパシフィック (ムンバイ) 、アジアパシフィック (ソウル) 、アジアパシフィック (シンガポール) 、アジアパシフィック (シドニー) 、アジアパシフィック (東京) GovCloud 、 (米国西部) 、カナダ (中部) 、欧州 (フランクフルト) 、欧州 (アイルランド) 、欧州 (ロンドン) 、欧州 (パリ) 、中東 (バーレーン) 、南米 (サンパウロ) 、米国東部 (バージニア北部) 、米国東部 (オハイオ) 、米国西部 (オレゴン) 、米国西部 (北カリフォルニア) 。

リアルタイムストリームの PII の編集または識別

ストリーミング文字起こしから個人を特定できる情報 (PII) を編集する場合、 Amazon Transcribe は、お客様のトランスクリプトに特定された PII の各インスタンスを [PII] に置き換えます。

ストリーミング文字起こしに使用できる追加オプションとして、PII 識別があります。PII 識別をアクティブ化すると、 は文字起こ Amazon Transcribe し結果の PII に Entities オブジェクトのラベルを付けます。出力サンプルについては、「編集済みストリーミング出力の例」と「PII 識別の出力例」を参照してください。

ストリーミング文字起こしによる PII の編集と識別は、オーストラリア (en-AU) 、英国 ()、米国 (en-GB) 、スペイン語の米国 (en-US) の英語の方言で使用できますes-US。

ストリーミングジョブの PII 識別とリダクションは、音声セグメントの完全な文字起こし時にのみ実行されます。

ストリーミング文字起こしで認識 Amazon Transcribe できる PII のタイプ

PII タイプ	説明
ADDRESS	実際の住所、米国、エニータウン市。メインストリート 100 番地や、ビル 123 番、スイート 12 番など。住所には、通り、ビル、場所、市区町村、州、国、郡、郵便番号、管区、近隣などを含めることができます。
ALL	この表に記載されているすべての PII のタイプを編集または特定します。
BANK_ACCOUNT_NUMBER	米国の銀行口座番号 この番号は通常 10～12 桁の長さですが、 Amazon Transcribe は下 4 桁のみの銀行口座番号も認識します。
BANK_ROUTING	米国の銀行口座の支店コード この番号は通常 9 桁の長さですが、 Amazon Transcribe は下 4 桁のみの支店コードも認識します。
CREDIT_DEBIT_CVV	VISA に存在する 3 桁のカード検証コード (CVV) MasterCard、Discover クレジットカードとデビットカード。American Express のク

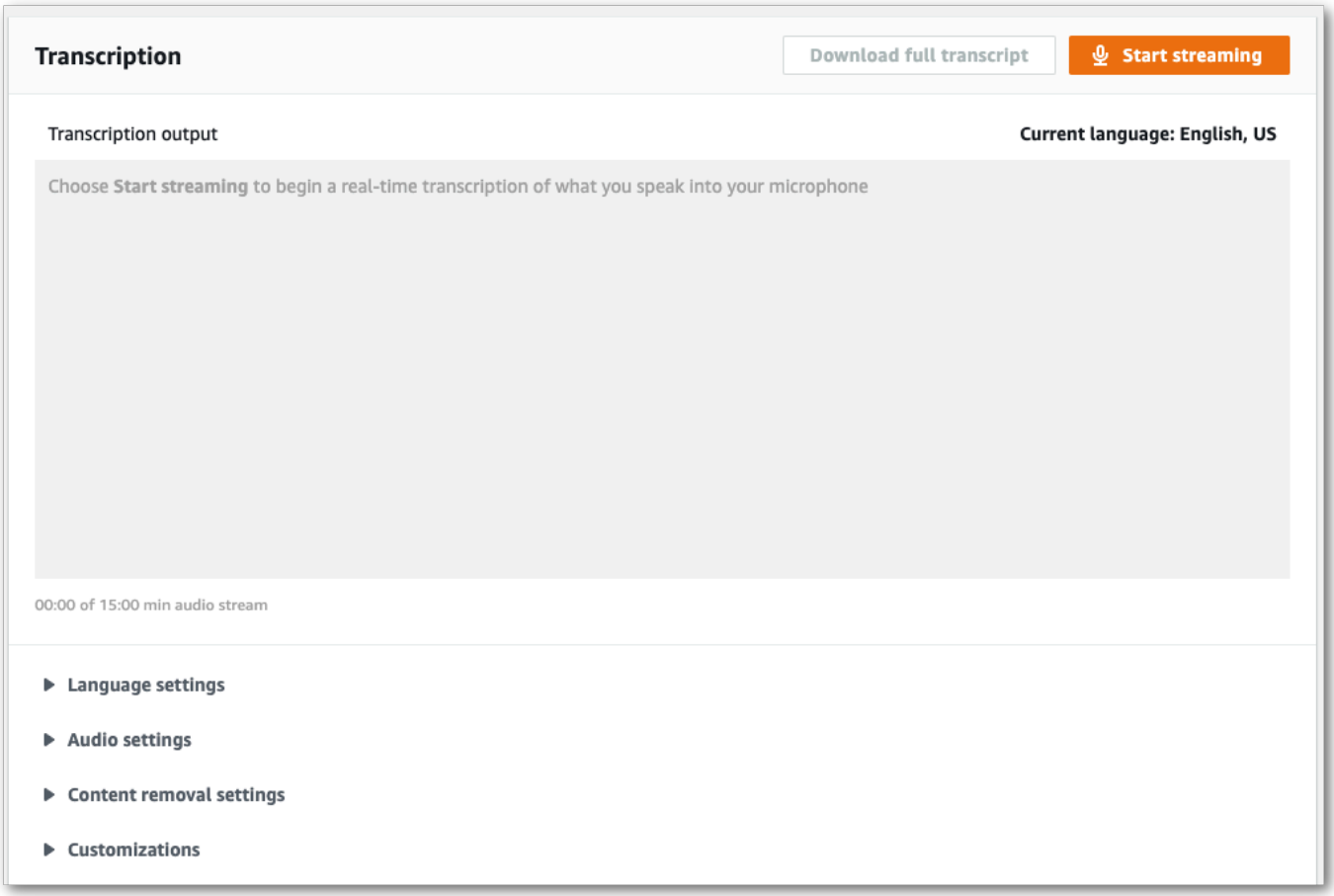
PII タイプ	説明
	レジットカードまたはデビットカードでは、4 桁の数字コードです。
CREDIT_DEBIT_EXPIRY	クレジットカードまたはデビットカードの有効期限日 この番号は通常 4 桁で、「月/年」または「MM/YY」という形式になっています。例えば、 は 01/21、01/2021、Jan 2021 などの有効期限を認識 Amazon Transcribe できます。
CREDIT_DEBIT_NUMBER	クレジットカードまたはデビットカードの番号。これらの番号の長さは 13 桁から 16 桁までさまざまですが、最後の 4 桁のみが存在する場合はクレジットカード番号またはデビットカード番号 Amazon Transcribe も認識されます。
EMAIL	efua.owusu@email.com などのメールアドレス。
NAME	個人の名前。このエンティティタイプには、Mr.、Mrs.、Miss、Dr. Amazon Transcribe does などのタイトルは含まれません。このエンティティタイプは、組織または住所の一部である名前には適用されません。例えば、は John Doe Organization を組織として認識し、Jane Doe Street を住所として Amazon Transcribe 認識します。
PHONE	電話番号 このエンティティタイプには、ファックス番号とポケットベル番号も含まれます。
PIN	銀行口座情報へのアクセスを可能にする 4 桁の個人識別番号 (PIN)。

PII タイプ	説明
SSN	社会保障番号 (SSN) は、米国市民、永住者、および一時的な労働居住者に発行される 9 桁の番号です。 は、最後の 4 桁のみが存在する場合に社会保障番号 Amazon Transcribe も認識します。

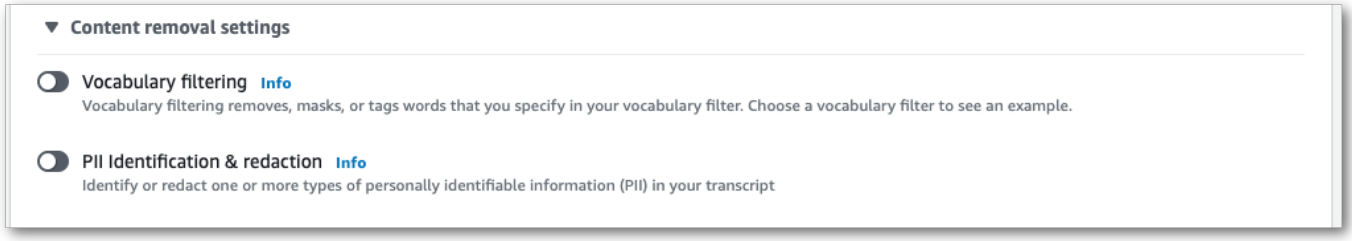
ストリーミング文字起こしは AWS Management Console、、 WebSocket、または HTTP/2 を使用して開始できます。

AWS Management Console

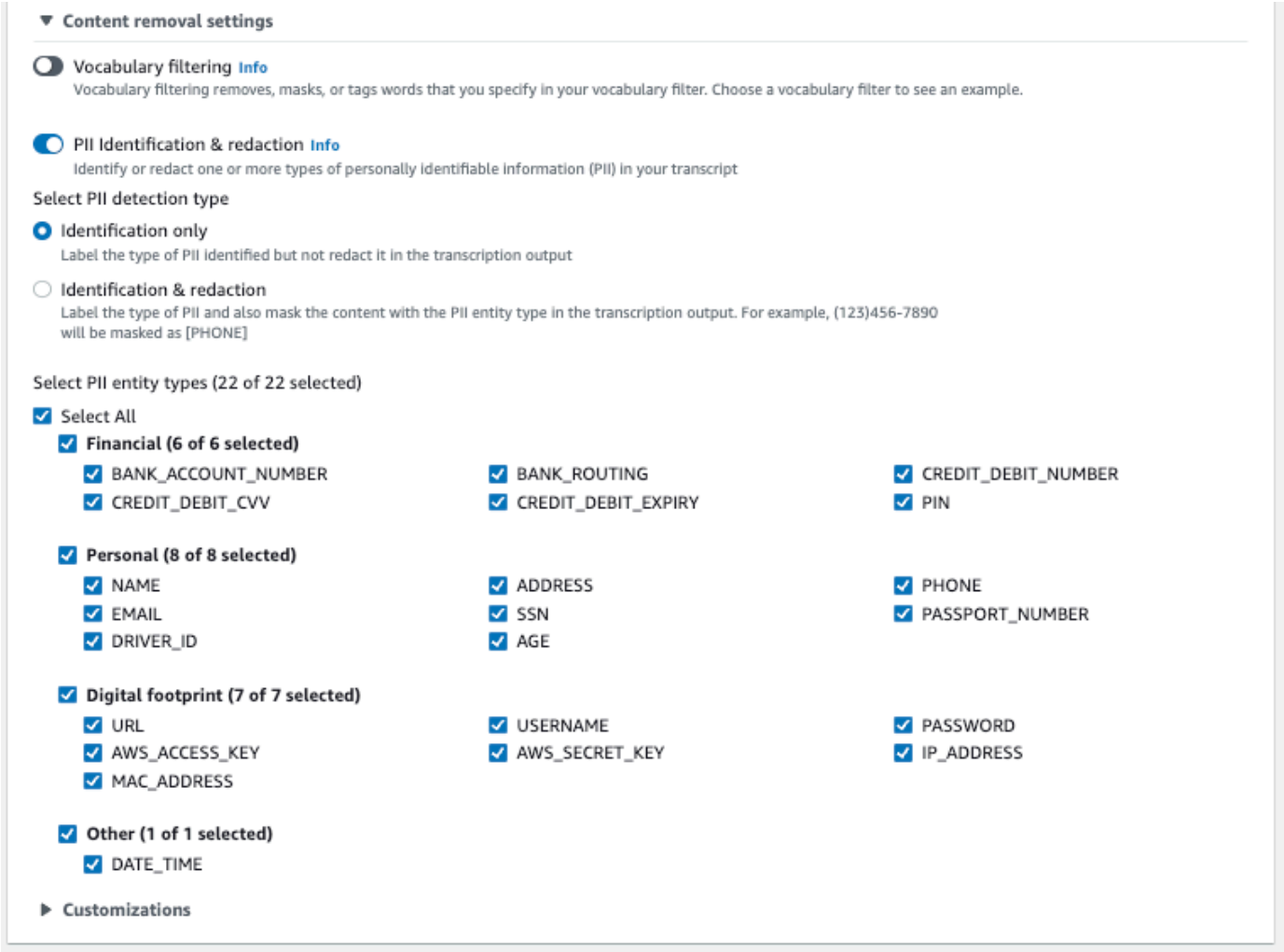
1. AWS Management Console にサインインします。
2. ナビゲーションペインで、[リアルタイム文字起こし] を選択します。コンテンツ削除の設定 にスクロールして、最小化されている場合はこのフィールドを展開します。



3. 「PII の識別とリダクション」をオンに切り替えます。



4. 「識別のみ」または「識別とリダクション」を選択し、トランスクリプトで識別または編集したい PII エンティティタイプを選択します。



5. これで、ストリームを書き起こす準備ができました。[ストリーミングを開始する] を選択し、話し始めます。ディクテーションを終了するには、[ストリーミングを停止する] を選択します。

WebSocket ストリーム

この例では、 WebSocket ストリームで PII リダクション (または PII 識別) を使用する署名付き URL を作成します。読みやすくするために、改行が追加されています。で WebSocket ストリームを使用する方法の詳細については、 Amazon Transcribe「」を参照してください WebSocket ストリームのセットアップ。パラメータの詳細については、「StartStreamTranscription」を参照してください。

```
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)
```

同じリクエストで content-identification-type と content-redaction-type 両方を使用することはできません。

パラメータ定義は API リファレンス にあります。すべての API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。 AWS

HTTP/2 ストリーミング

この例では、PII 識別または PII リダクションを有効にした状態で HTTP/2 リクエストを作成します。での HTTP/2 ストリーミングの使用の詳細については Amazon Transcribe、「」を参照してくださいHTTP/2 ストリームの設定。に固有のパラメータとヘッダーの詳細については、 Amazon Transcribe「」を参照してください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
```

同じリクエストで content-identification-type と content-redaction-type 両方を使用することはできません。

パラメータ定義は API リファレンス にあります。すべての API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。 AWS

> **Note**
>
> ストリーミングの PII リダクションは、 AWS リージョンアジアパシフィック (ソウル) 、アジアパシフィック (シドニー) 、アジアパシフィック (東京) 、カナダ (中部) 、欧州 (フランクフルト) 、欧州 (アイルランド) 、欧州 (ロンドン) 、米国東部 (バージニア北部) 、米国東部 (オハイオ) 、米国西部 (オレゴン) の でのみサポートされています。

PII リダクションと識別の出力例

次の例は、バッチジョブおよびストリーミングジョブからの編集された出力、およびストリーミングジョブからの PII 識別を示しています。

コンテンツのマスキングを使用する文字起こしジョブでは、2 種類の confidence 値を生成します。自動音声認識 (ASR) 信頼度は、type が pronunciation または punctuation である項目が特定の発話であることを示します。次の文字起こしの出力では、単語 Good の confidence が 1.0 です。この信頼度の値は、この文字起こしで発話された単語が「Good」であることを 100% 確信していることを示します。Amazon Transcribe[PII] タグの confidence 値は、マスキングのフラグを付けた発話が真に PII であるという信頼度を示します。次の文字起こしの出力で

は、confidenceconfidence の 0.99%Amazon Transcribe は、文字起こしのマスキングしたエンティティが PII であることを 99.99%0.9999 確信していることを示します。

編集後の出力例 (バッチ)

```
{
    "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"
}
```

比較のための未編集のトランスクリプトは次のとおりです。

```
{
    "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"
}
```

編集済みストリーミング出力の例

```
{
    "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 識別の出力例

PII 識別は、ストリーミングトランスクリプションジョブで使用できる追加機能です。特定されたPIIは、Entities各セグメントのセクションに記載されています。

```
{
    "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
                }
            ]
        }
    }
}
```

ビデオ字幕の作成

Amazon TranscribeWebVTT (*.vtt) および SubRip (*.srt) 出力をサポートし、ビデオ字幕として使用できます。バッチビデオ文字起こしジョブを設定するときに、1 つまたは両方のファイルタイプを選択できます。字幕機能を使用すると、選択した文字起こしファイルと通常の文字起こしファイル (追加情報を含む) が生成されます。字幕と文字起こしファイルは、同じ出力先に出力されます。

字幕は、テキストが話されると同時に表示され、自然な一時停止があるか、スピーカーが話しかけるまで表示され続けます。トランスクリプションリクエストで字幕を有効にしていて、音声に音声が含まれていない場合、字幕ファイルは作成されないことに注意してください。

> **Important**
>
> Amazon Transcribe0字幕出力にはデフォルトの開始インデックスを使用します。これは、1広く使用されているの値とは異なります。の開始インデックスが必要な場合は1、`OutputStartIndex`パラメータを使用して API リクエストの AWS Management Console OR でこれを指定できます。

間違った開始インデックスを使用すると、他のサービスとの互換性エラーが発生する可能性があるため、字幕を作成する前に、必要な開始インデックスを確認してください。使用する値がわからない場合は、1選択することをおすすめします。`Subtitles`詳細については、を参照してください。

字幕でサポートされる機能 :

- コンテンツの墨消し — 編集されたコンテンツは、字幕と通常の文字起こし出力で反映されます。PII音声は変更されません。
- ボキャブラリーフィルター : 字幕ファイルは文字起こしファイルから生成されるため、標準の文字起こし出力でフィルタリングした単語も字幕でフィルタリングされます。フィルタされたコンテンツは文字起こしと字幕ファイルに空白 または *** として表示されます。音声は変更されません。
- スピーカー文字起こしセグメントに複数のスピーカーがある場合、ダッシュを使用して各スピーカーを区別します。ダッシュを使用して各スピーカーを区別します。これは WebVTT SubRip と形式の両方に適用されます。次に例を示します。
  - — 人によって話されるテキスト 1
  - — 人によって話されるテキスト 2

字幕ファイルは、Amazon S3文字起こし出力と同じ場所に保存されます。

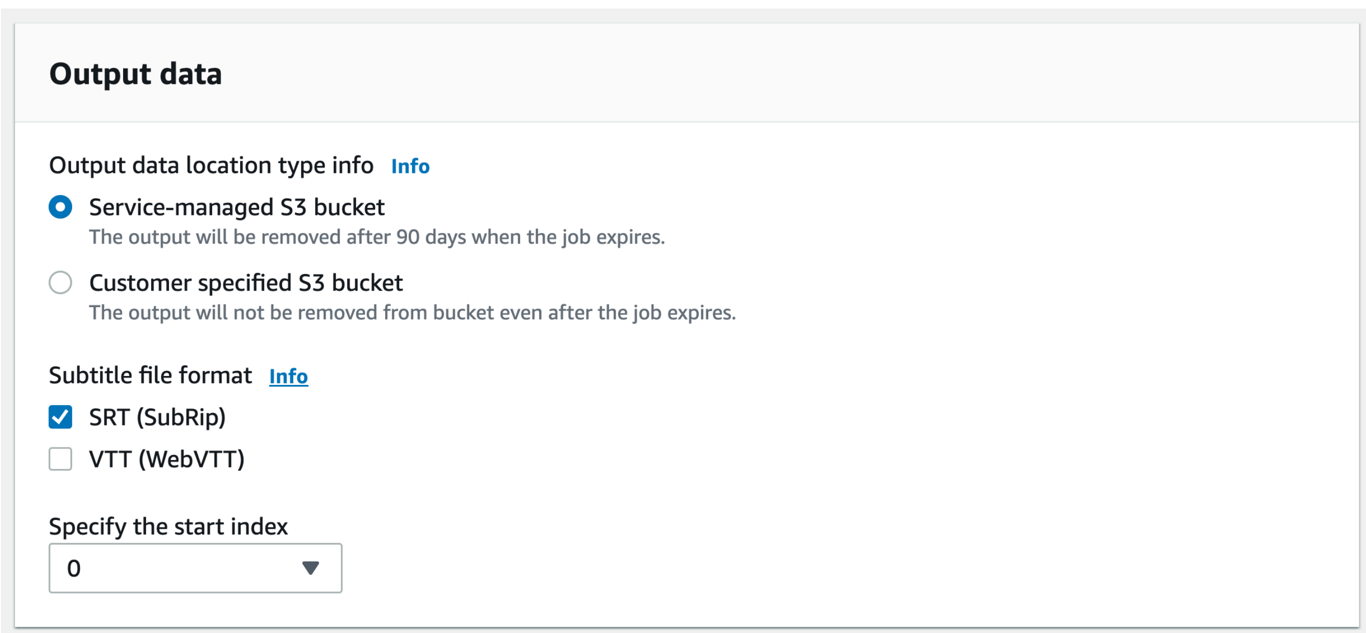
字幕作成のビデオチュートリアルについては、「Amazon Transcribe Video Snacks: コードを書かずに動画字幕を作成」を参照してください。

字幕ファイルの生成

、、または AWSSDK を使用して字幕ファイルを作成できます。次の例を参照してください。AWS Management ConsoleAWS CLI

AWS Management Console

1. AWS Management Console にサインインします。
2. ナビゲーションペインで変換の文字起こしジョブの作成ボタンをクリックして変換を実行します変換 これにより、ジョブの詳細を指定 ページが開きます。字幕オプションは、出力データ パネルにあります。
3. 字幕ファイルに必要な形式を選択し、スタートインデックスの値を選択します。Amazon Transcribe1デフォルトはですが0、より広く使用されていることに注意してください。どの値を使うべきかわからない場合は、他のサービスとの互換性が向上する可能性があるため1、選択することをおすすめします。



4. 他に追加したい他のフィールドに入力後、変換を実行します。これにより、ジョブの設定 : オプション ページ へ移動します。
5. ジョブの作成ボタンをクリックして変換を実行します変換を実行します変換を実行します変換を実行します変換を実行します変換を実行します変換を実行します変換を実行します

AWS CLI

この例では、start-transcription-jobSubtitlesコマンドとパラメーターを使用しています。詳細については、StartTranscriptionJobおよびSubtitlesを参照してください。

```
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
```

別の例として、start-transcription-job変換に字幕を追加するリクエストボディを使用します。

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

ファイル my-first-subtitle-job.json に次のリクエストボディを実行します。

```
{
  "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)

この例では、transstart_transcription_job メソッドの AWS SDK for Python (Boto3) 引数で、Subtitles を使用して字幕を追加します。詳細については、StartTranscriptionJobおよびSubtitlesを参照してください。

機能固有の例、シナリオ、クロスサービスの例など、AWS SDK を使用するその他の例については、この章を参照してください。SDK を使用した Amazon Transcribe のコード例 AWS SDKs

```
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)
```

コール分析によるコールセンターの音声の分析

Amazon Transcribe コール分析を使用して、カスタマーとエージェントのやり取りに関するインサイトを取得します。コール分析はコールセンターの音声専用に設計されており、各通話や各参加者に関する貴重なデータを自動的に提供します。また、通話中の特定の時点のデータに的を絞ることもできます。たとえば、通話の最初の数秒と通話の最後の 4 分の 1 の時間におけるお客様の感情を比較して、エージェントがポジティブなエクスペリエンスを提供したかどうかを確認できます。その他の使用例を次のセクションに示します。

コール分析は、通話後の文字起こしとリアルタイム文字起こしに使用できます。 Amazon S3 バケットにあるファイルを文字起こしする場合は、通話後文字起こしを実行します。音声ストリームを文字起こしする場合は、リアルタイム文字起こしを実行していることになります。この 2 つの文字起こしの方法では、コール分析のインサイトや機能が異なります。各メソッドの詳細については、「通話後分析」と「リアルタイムコール分析」を参照してください。

リアルタイムコール分析の文字起こしを使うと、リクエストに通話後分析を含めることもできます。通話後分析のトランスクリプトは、リクエストで指定した Amazon S3 バケットに保存されます。詳細については、「通話後分析とリアルタイム文字起こし」を参照してください。

> コール分析特有の API オペレーション
>
> 通話後:
> CreateCallAnalyticsCategory、DeleteCallAnalyticsCategory、DeleteCallAnalytics
> リアルタイム: StartCallAnalyticsStreamTranscription、
> StartCallAnalyticsStreamTranscriptionWebSocket

一般的なユースケース

通話後文字起こし:

- 問題の頻度を経時的にモニタリング: 通話の分類を使用して、トランスクリプト内で繰り返し使われるキーワードを特定します。
- カスタマーサービスエクスペリエンスに関するインサイトの取得: コールの特性 (非通話時間、通話時間、中断、声の大きさ、通話速度) と感情分析を使用して、通話中にお客様の問題が適切に解決されているかどうかを判断します。

- 規制遵守または企業ポリシーの遵守を確保: 会社固有の挨拶や免責事項にキーワードやフレーズを設定して、エージェントが規制要件を満たしていることを確認します。
- お客様の個人情報の取り扱いの改善: お客様のプライバシーを保護するために、文字起こしの出力や音声ファイルには PII リダクションを使用します。
- スタッフトレーニングの改善: 基準 (感情、非通話時間、中断、通話速度) を使用して、お客様とのポジティブまたはネガティブなやり取りの例として使用できるトランスクリプトにフラグを付けます。
- ポジティブなカスタマーエクスペリエンスを生み出すスタッフの効果を測定: 感情分析を使用して、通話が進むにつれてエージェントがネガティブなカスタマー感情をポジティブな感情に変えることができるかどうかを測定します。
- データ整理の改善: カスタムカテゴリ (キーワードやフレーズ、感情、通話時間、中断など) に基づいて通話にラベルを付けて並べ替えます。
- 生成 AI を使用して通話の重要な側面を要約: 通話の生成要約を使用して、トランスクリプトの簡潔な要約を取得します。これには、通話で話し合った問題、アクション項目、結果などの主な要素が含まれます。

リアルタイム文字起こし

- エスカレーションをリアルタイムで軽減: お客様が「マネージャーと話したい」と言った場合など、重要なフレーズにリアルタイムアラートを設定して、通話がエスカレートし始めたときにフラグを付けます。リアルタイムのカテゴリーマッチを使用して、リアルタイムアラートを作成できます。
- お客様の個人情報の取り扱いの改善: お客様のプライバシーを保護するために、文字起こしの出力に PII 識別または PII リダクションを使用します。
- カスタムキーワードとフレーズの識別: カスタムカテゴリを使用して、通話中の特定のキーワードにフラグを付けます。
- 問題を自動的に特定: 自動問題検出を使用すると、通話中に特定されたすべての問題を簡潔にまとめることができます。
- ポジティブなカスタマーエクスペリエンスを生み出すスタッフの効果を測定: 感情分析を使用して、通話が進むにつれてエージェントがネガティブなカスタマー感情をポジティブな感情に変えることができるかどうかを測定します。
- エージェントアシストの設定: 選択したインサイトを使用して、お客様からの通話を解決するための積極的な支援をエージェントに提供します。詳細については、「Amazon 言語 AI サービスによるコンタクトセンターのライブコール分析とエージェントアシスト」を参照してください。

コール分析で使用可能な機能と Amazon Transcribe および Amazon Transcribe Medical の機能を比較するには、機能テーブル を参照してください。

開始するには、「通話後分析の文字起こしを開始する」と「リアルタイムコール分析の文字起こしを開始する」を参照してください。コール分析出力は、標準の文字起こしジョブの出力に似ていますが、追加の分析データが含まれています。サンプル出力を表示するには、「通話後分析出力」と「リアルタイムコール分析出力」とを参照してください。

考慮事項と追加情報

コール分析を使用する前に、次の点に注意してください。

- コール分析は、2 チャネルの音声のみをサポートします。エージェントは 1 つのチャネルに、お客様は 2 つ目のチャネルに存在します。
- Job キューイング は通話後分析ジョブに対して常に有効になっているため、同時に実行できるコール分析ジョブは 100 に制限されます。クォータの引き上げをリクエストするには、AWS Service Quotas を使用してください。
- 通話後分析ジョブの入力ファイルは、500 MB を超えてはならず、4 時間未満である必要があります。圧縮された非 WAV オーディオファイル形式によっては、ファイルサイズ制限が小さくなる場合があることに注意してください。
- カテゴリを使用する場合は、コール分析の文字起こしを開始する前に、必要なカテゴリをすべて作成する必要があります。新しいカテゴリは、既存の文字起こしには適用できません。新しいカテゴリの作成方法については、「通話後の文字起こしカテゴリの作成」と「リアルタイム文字起こしのカテゴリの作成」を参照してください。
- 一部のコール分析クォータは、 Amazon Transcribe および Amazon Transcribe Medical とは異なります。詳細については、AWS 「 全般のリファレンス」を参照してください。

AWS Machine Learningブログで詳しく知る

コール分析オプションの詳細については、次の情報を参照してください。

- Amazon 言語 AI サービスによるコンタクトセンターの通話後分析
- Amazon 言語 AI サービスによるコンタクトセンターのライブコール分析とエージェントアシスト

コール分析の出力と機能のサンプルを表示するには、GitHubデモ「」を参照してください。また、文字起こしを 形式に変換するための JSON から Word へのドキュメントアプリケーションも提供しています。 easy-to-read

利用可能なリージョンとクォータ

コール分析は、次の でサポートされています AWS リージョン。

リージョン	文字起こしタイプ
ap-northeast-1 (東京)	post-call, real-time
ap-northeast-2 (ソウル)	post-call, real-time
ap-south-1 (ムンバイ)	post-call
ap-southeast-1 (シンガポール)	post-call
ap-southeast-2 (シドニー)	post-call, real-time
ca-central-1 (カナダ、中部)	post-call, real-time
eu-central-1 (フランクフルト)	post-call, real-time
eu-west-2 (ロンドン)	post-call, real-time
us-east-1 (バージニア北部)	post-call, real-time
us-west-2 (オレゴン)	post-call, real-time

Amazon Transcribe、Amazon Transcribe Medical、およびコール分析ではリージョンのサポートが異なることに注意してください。

サポートされている各リージョンのエンドポイントを取得するには、「AWS 全般のリファレンス」の「サービスエンドポイント」を参照してください。

文字起こしに関連するクォータのリストについては、「AWS 全般のリファレンス」の「Service Quotas」を参照してください。一部のクォータは、リクエストに応じて変更することができます。調整可能列に「はい」と表示されている場合は、増加をリクエストできます。そのためには、表示されたリンクを選択します。

通話後分析

コール分析では、カスタマーサービスの傾向をモニタリングするのに役立つ通話後分析を提供します。

通話後文字起こしでは、次のような情報を得ることができます。

- 通話時間、非通話時間、話者の声の大きさ、中断、通話速度、問題、結果、アクション項目などの通話の特性
- 通話全体の簡潔な要約を作成する通話の生成要約
- 特定のキーワードや条件に絞り込むルールを使用したカスタムの分類
- テキストトランスクリプトと音声ファイルの PII リダクション
- 通話中のさまざまな場面における各発信者のスピーカーの感情

通話後のインサイト

このセクションでは、通話後分析の文字起こしで得られるインサイトについて詳しく説明します。

コールの特性

コールの特性機能は、次の基準で、エージェントとカスタマーの対話の品質を測定します。

- 中断: 一方の参加者がもう一方の参加者の発言を途中で中断したかどうかと、そのタイミングを測定します。頻繁な中断は無礼または怒りと関連している可能性があり、一方または両方の参加者の否定的な感情と関連していることもあります。
- ラウドネス: 各参加者が話している音量を測定します。このメトリクスを使用して、発信者またはエージェントが大声で話している、または怒鳴っているかどうかを確認します。多くの場合、怒っていることを示します。このメトリクスは、0 から 100 の範囲で正規化された値 (特定のセグメントにおける音声の1秒あたりの音声レベル) として表され、値が大きいほど音声が大きいことを示します。
- 非通話時間: 音声が含まれていない時間を測定します。このメトリクスを使用して、エージェントが顧客を過度に長い時間保留しているなど、長い無音時間があるかどうかを確認します。
- 通話速度: 両方の参加者が話している速度を測定します。1 人の参加者が話すのが速すぎると理解度に影響が出ることがあります。このメトリクスは 1 分あたりの単語数で測定されます。
- 通話時間: 通話中に各参加者が通話した時間 (ミリ秒単位) を測定します。このメトリクスを使って、1 人の参加者が通話を独占しているかどうか、または会話のバランスがとれているかどうかを識別します。

- 問題、結果、アクション項目: 通話トランスクリプトから問題、結果、アクション項目を特定します。

以下は出力例です。

通話の生成要約

生成通話の要約により、通話全体の簡潔な概要が作成され、通話の理由、問題を解決するために実行されたステップ、次のステップなどの主要コンポーネントがキャプチャされます。

通話の生成要約を使用して、以下のことができます。

- 通話中や通話後に手動でメモを取る必要を減らします。
- エージェントは、通話後の作業よりも、待機中の発信者との会話により多くの時間を費やすことができるため、エージェントの効率が向上します。
- 通話の要約はトランスクリプト全体よりもはるかに短時間でレビューできるため、スーパーバイザーによるレビューが効率化されます。

通話の生成要約を通話後の分析ジョブで使用するには、「通話の生成要約の有効化」を参照してください。出力例については、「通話の生成要約の出力例」を参照してください。通話の生成要約には別途料金がかかります (料金ページを参照してください)。

> **Note**
>
> 通話の生成要約は現在、us-east-1 と us-west-2 で利用可能です。この機能は、オーストラリア (en-AU)、英国 (en-GB)、インド (en-IN)、アイルランド (en-IE)、スコットランド (en-AB)、米国 (en-US)、ウェールズ (en-WL) の英語方言でサポートされています。

カスタムの分類

通話の分類を使用して、通話内のキーワード、フレーズ、感情、またはアクションにフラグを付けます。分類のオプションは、中断の多いネガティブな感情通話などのエスカレーションをトリアージしたり、通話を企業の部署などの特別なカテゴリに整理するのに役立ちます。

カテゴリに追加できる条件には以下が含まれます。

- 非通話時間: カスタマーとエージェントのどちらも通話していない時間。

- 中断: カスタマーまたはエージェントによって相手が中断された場合。
- カスタマーまたはエージェントの感情: 指定された期間におけるカスタマーまたはエージェントがどのように感じているか。指定した期間に会話の少なくとも 50% が (2 人のスピーカー back-and-forth 間の) 変わった場合、指定した感情と一致すると、 は感情の一致 Amazon Transcribe を考慮します。
- キーワードまたはフレーズ: 正確なフレーズに基づいて文字起こしの一部と一致させます。例えば、「マネージャーと話したい」というフレーズにフィルターを設定すると、 Amazon Transcribe は正確なフレーズをフィルターします。

また、前の基準とは逆の条件にフラグを立てることもできます(通話時間、中断なし、感情がない、特定のフレーズがない)。

以下は出力例です。

カテゴリの詳細、または新しいカテゴリの作成方法については、「通話後の文字起こしカテゴリの作成」を参照してください。

機密データのリダクション

機密データのリダクションでは、テキストトランスクリプトおよび音声ファイル内の個人を特定できる情報 (PII) が置き換えられます。編集されたトランスクリプトは、元のテキストを `[PII]` に置き換え、音声ファイルを編集すると、話された個人情報が無音に置き換えられます。このパラメータは、お客様の情報を保護するのに役立ちます。

> **Note**
>
> 通話後 PII リダクションは、米国英語 (en-US) および米国スペイン語 () でサポートされていますes-US。

この機能を使用して編集された PII のリストの表示、または Amazon Transcribeでのリダクションの詳細については、「個人を特定できる情報の編集または特定」を参照してください。

以下は出力例です。

感情分析

感情分析では、コール全体を通してカスタマーとエージェントがどのように感じているかを推定します。このメトリクスは、定量値 (5 から -5 の範囲) と定性値 (`positive`、`neutral`、`mixed` または

negative) の両方で表されます。定量値は四半期およびコールごとに提供され、定性値はターンごとに提供されます

このメトリクスは、コールが終了するまでに、エージェントが怒っているカスタマーを喜ばすことができるかどうかを識別するのに役立ちます。

感情分析は機能 out-of-the-box するため、モデルトレーニングやカスタムカテゴリなどのカスタマイズはサポートされていません。

以下は出力例です。

通話後の文字起こしカテゴリの作成

通話後分析ではカスタムカテゴリの作成がサポートされているため、特定のビジネスニーズに合わせてトランスクリプト分析を調整できます。

さまざまなシナリオをカバーするカテゴリをいくつでも作成できます。作成するカテゴリごとに、1 から 20 のルールを作成する必要があります。各ルールは、中断、キーワード、非通話時間、感情という 4 つの基準のいずれかに基づいています。`CreateCallAnalyticsCategory` オペレーションでこれらの基準を使用する詳細については、「通話後分析カテゴリのルールの条件 セクション」を参照してください。

メディア内のコンテンツが、指定したカテゴリのすべてのルールに一致する場合、 Amazon Transcribe は出力にそのカテゴリのラベル付けを行います。JSON 出力のカテゴリマッチの例については、「通話の分類出力」を参照してください。

カスタムカテゴリを使用してできるその他の例を紹介します。

- ネガティブなカスタマー感情で終わった通話など、特定の特徴を持つ通話を分離します。
- 特定のキーワードセットにフラグを付けて追跡することで、お客様の問題の傾向を特定します。
- エージェントが通話の最初の数秒間に特定のフレーズを話す (または省略する) などのコンプライアンスをモニタリングします。
- エージェントによる中断やお客様からのネガティブな感情が多い通話にフラグを立てることで、カスタマーエクスペリエンスに関するインサイトを得られます。
- 複数のカテゴリを比較して相関関係を測定します。たとえば、ウェルカムフレーズを使用するエージェントがお客様のポジティブな感情と相関関係があるかどうかの分析などです。

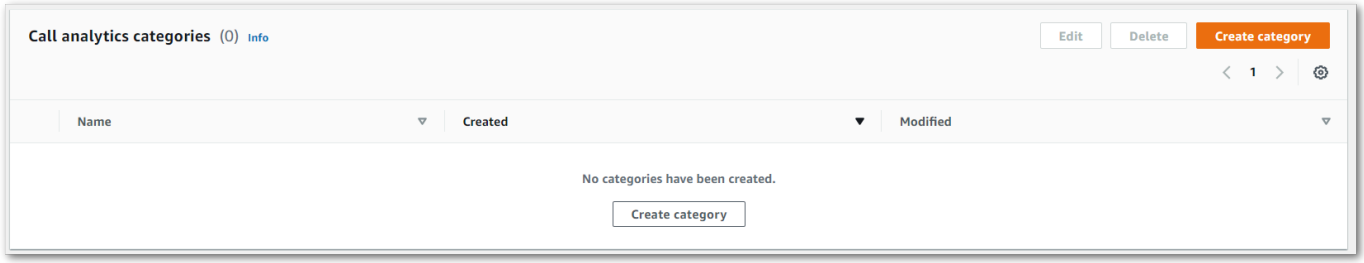
通話後カテゴリとリアルタイムカテゴリ

新しいカテゴリを作成する場合、通話後分析カテゴリ (POST_CALL) として作成するか、リアルタイム通話分析カテゴリ (REAL_TIME) として作成するかを指定できます。オプションを指定しない場合、カテゴリはデフォルトで通話後カテゴリとして作成されます。通話後分析カテゴリマッチは、通話後分析の文字起こしが完了すると、出力に表示されます。

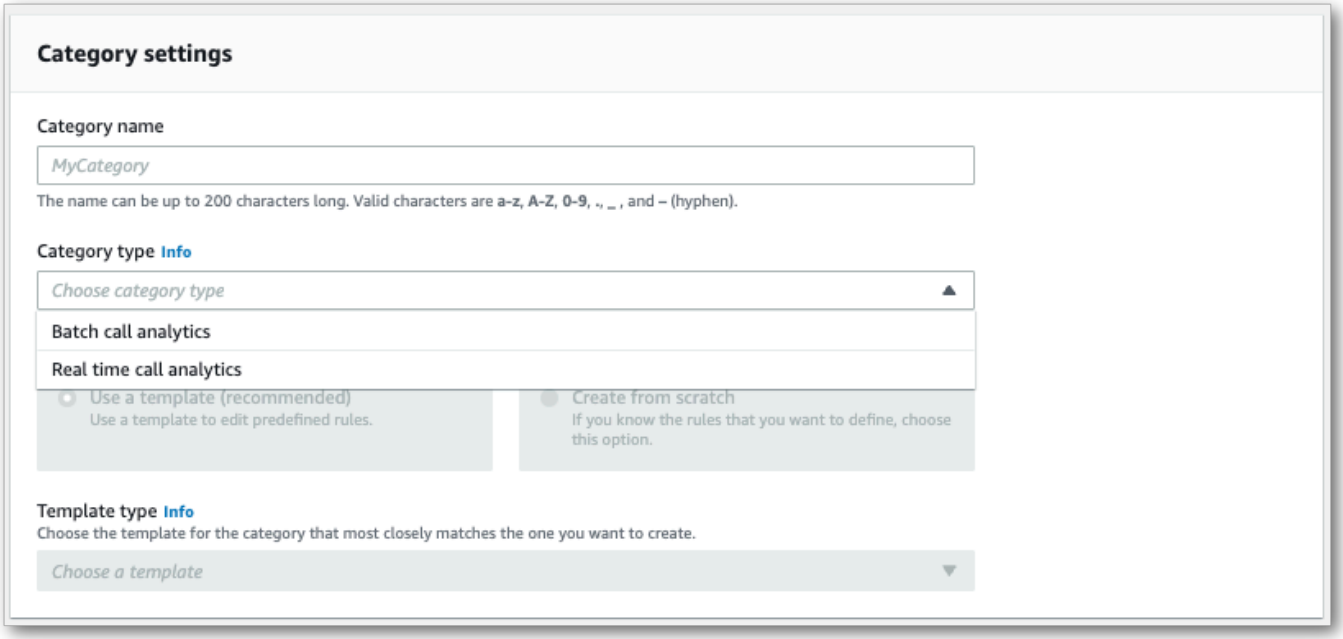
通話後分析用の新しいカテゴリを作成するには、AWS Management Console、AWS CLI、または AWS SDK を使用できます。例については以下を参照してください。

AWS Management Console

1. ナビゲーションペインの で Amazon Transcribe、Amazon Transcribe 分析の呼び出し を選択します。
2. [コール分析カテゴリ] を選択すると、[コール分析カテゴリ] ページに移動します。[カテゴリの作成] を選択します。

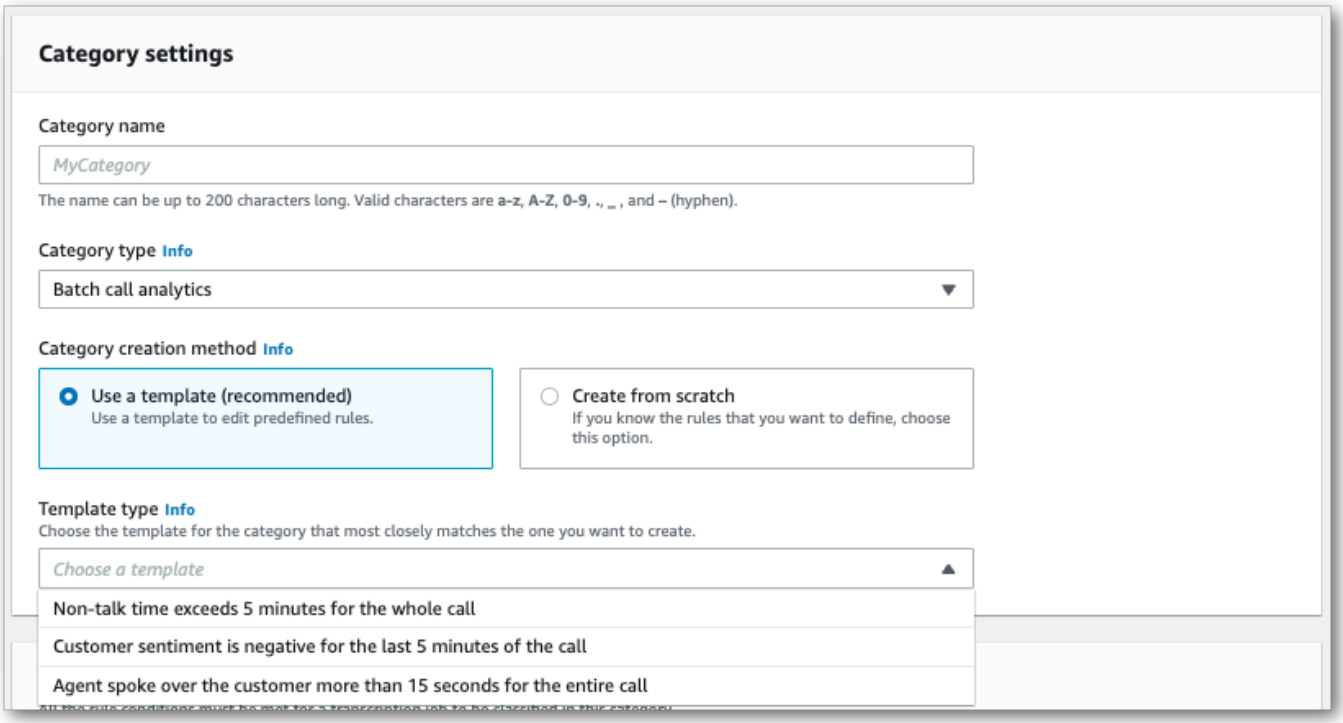


3. [カテゴリの作成ページ] が表示されます。カテゴリの名前を入力し、カテゴリタイプのドロップダウンメニューで [バッチコール分析] を選択します。

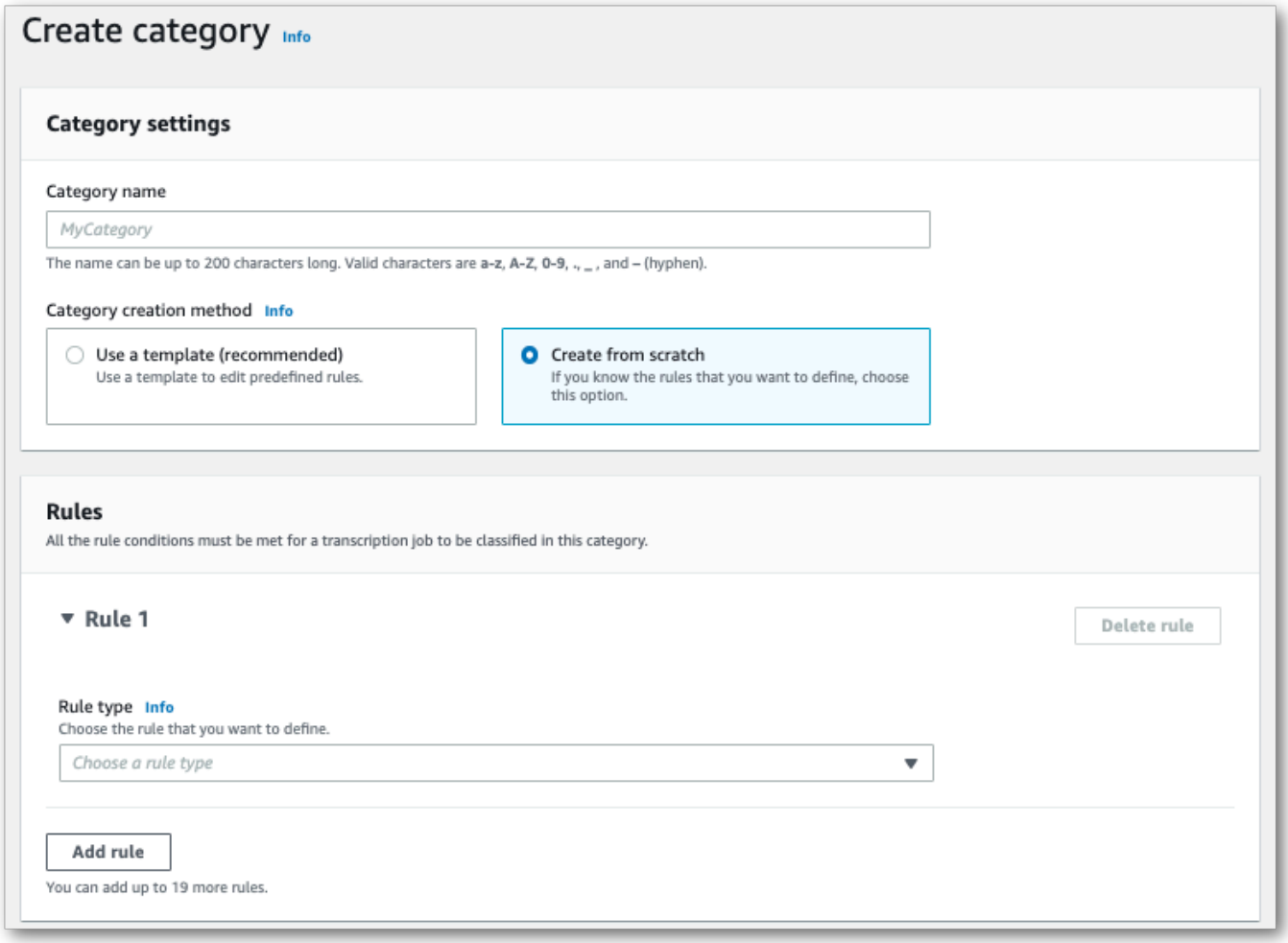


4. テンプレートを選択してカテゴリを作成することも、一から作成することもできます。

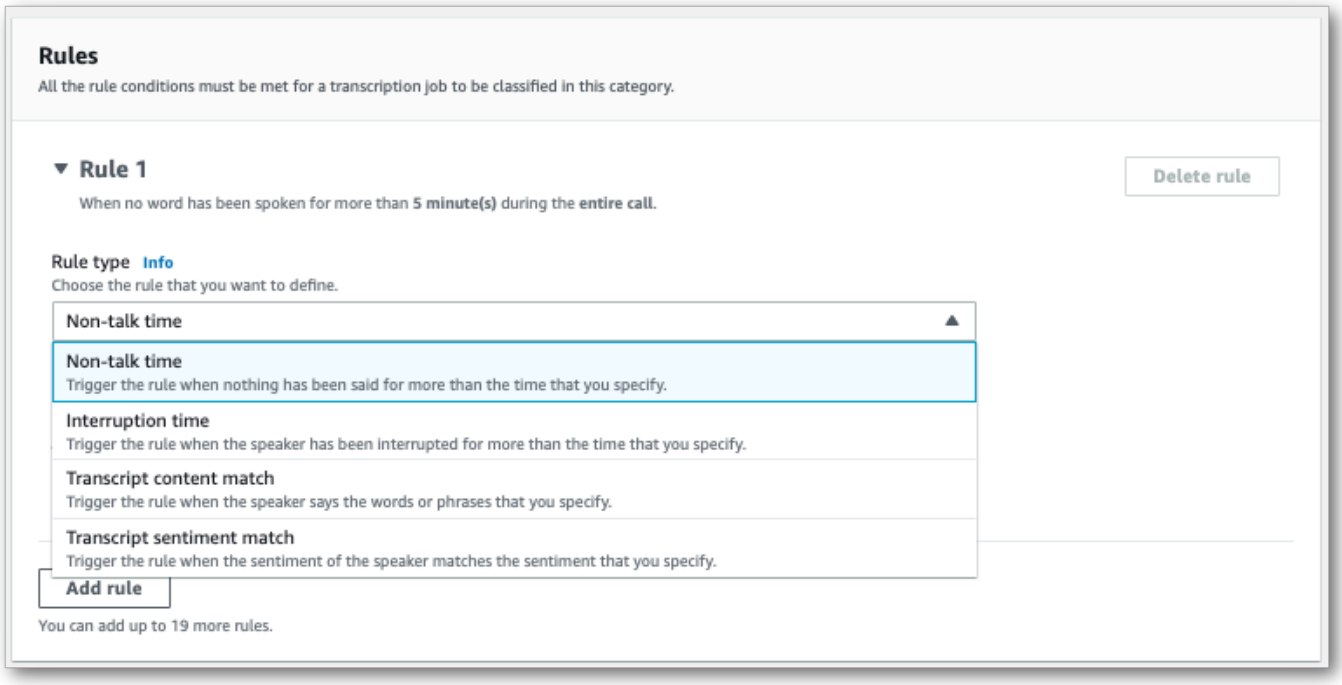
   テンプレートを使用する場合: [テンプレートを使用する (推奨)] を選択し、必要なテンプレートを選択してから [カテゴリの作成] を選択します。



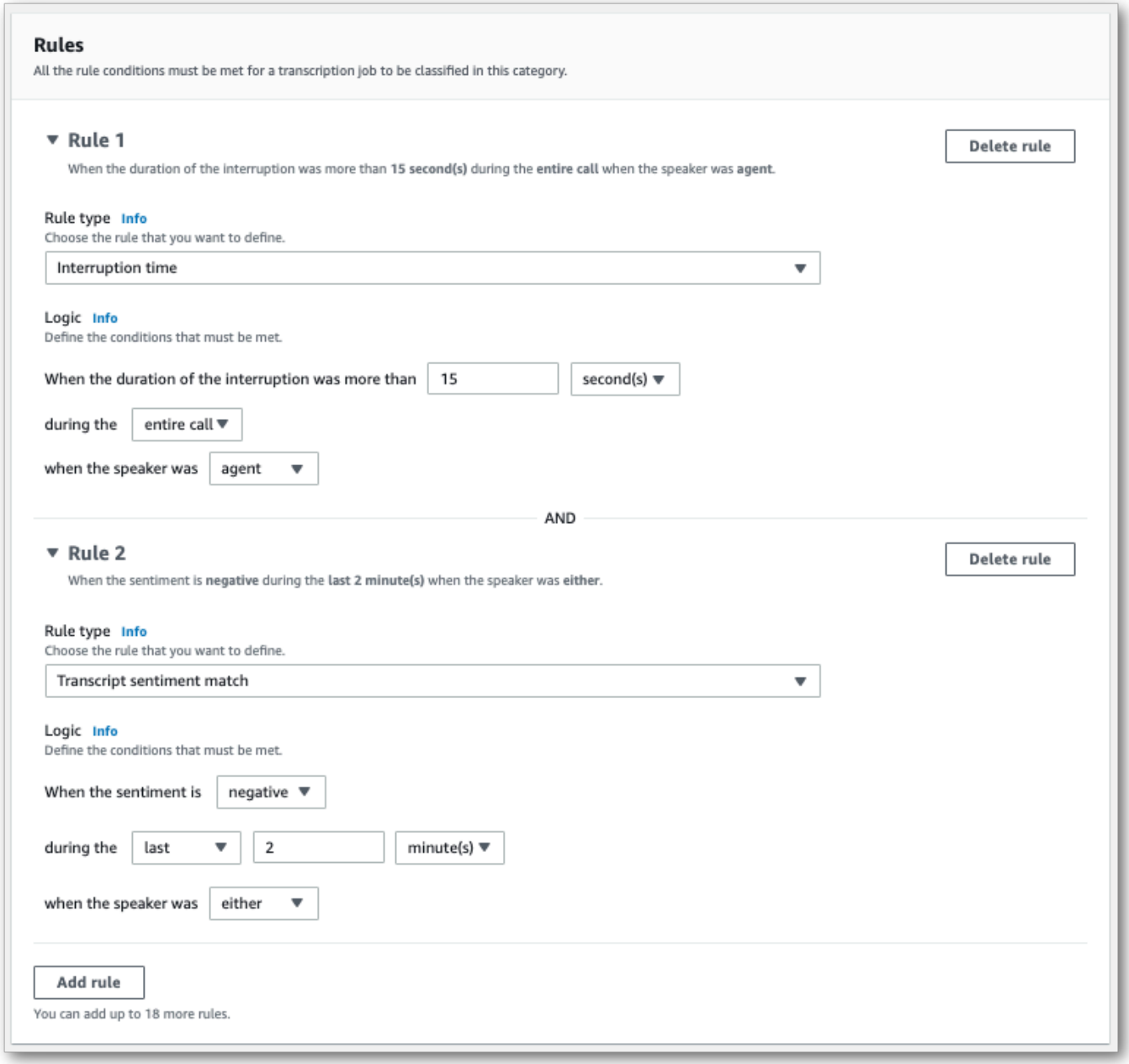
5. カスタムカテゴリを作成する場合: [最初から作成] を選択します。



6. ドロップダウンメニューを使用して、カテゴリにルールを追加します。1 つのカテゴリには最大 20 ルールまで追加できます。



7. これは、2 つのルールがあるカテゴリの例です。1 つは、通話中に 15 秒以上お客様の話を中断したエージェントと、もう 1 つは通話の最後の 2 分間にお客様またはエージェントが感じたネガティブな感情です。



8. カテゴリにルールを追加し終えたら、[カテゴリの作成] を選択します。

AWS CLI

この例では、 create-call-analytics-category コマンドを使用します。詳細については、CreateCallAnalyticsCategory、CategoryProperties、およびRuleを参照してください。

以下の例では、ルールを含むカテゴリを作成します。

- カスタマーは最初の 60 秒間で中断されました。これらの中断時間は少なくとも 10 秒続きました。
- 通話の 10% から 80% の間では少なくとも20 秒の無音が続きました。
- エージェントは、通話中のある時点でネガティブな感情を抱いていました。
- 通話の最初の 10 秒の間に、「ようこそ」や「こんにちは」という単語は使われていません。

この例では、 create-call-analytics-category コマンドと、カテゴリに複数のルールを追加するリクエスト本文を使用します。

```
aws transcribe create-call-analytics-category \
--cli-input-json file://filepath/my-first-analytics-category.json
```

ファイル my-first-analytics-category.json には、次のリクエスト本文が含まれています。

```
{
  "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)

この例では、 を使用して AWS SDK for Python (Boto3) 、create_call_analytics_category メソッドの 引数CategoryNameと Rules引数を使用してカテゴリを作成します。詳細については、CreateCallAnalyticsCategory、CategoryProperties、およびRuleを参照してください。

機能固有の例、シナリオ例、クロスサービス例など、 AWS SDKsSDK を使用した Amazon Transcribe のコード例 AWS SDKs「」の章を参照してください。

以下の例では、ルールを含むカテゴリを作成します。

- カスタマーは最初の 60 秒間で中断されました。これらの中断時間は少なくとも 10 秒続きました。
- 通話の 10% から 80% の間では少なくとも20 秒の無音が続きました。
- エージェントは、通話中のある時点でネガティブな感情を抱いていました。
- 通話の最初の 10 秒の間に、「ようこそ」や「こんにちは」という単語は使われていません。

```
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)
```

通話後分析カテゴリのルールの条件

このセクションでは、CreateCallAnalyticsCategory API オペレーションを使用して作成できるカスタム POST_CALL ルールのタイプについて説明します。

中断マッチ

中断 (InterruptionFilter データタイプ) を使用するルールは、以下と一致するように設計されています。

- エージェントがお客様を中断させるインスタンス
- お客様がエージェントを中断させるインスタンス
- いずれかの参加者が他の参加者を中断させる
- 中断がないこと

InterruptionFilter で使用できるパラメータの例を以下に示します。

```
"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
```

```
},
```

これらのパラメータとそれぞれに関連する有効な値の詳細については、「CreateCallAnalyticsCategory」および「InterruptionFilter」を参照してください。

キーワードマッチ

キーワード (TranscriptFilter データタイプ) を使用するルールは、以下と一致するように設計されています。

- エージェント、お客様、あるいはその両方が話すカスタム単語またはフレーズ
- エージェント、お客様、あるいはその両方が口にしないカスタム単語またはフレーズ
- 特定の期間に出現するカスタム単語またはフレーズ

TranscriptFilter で使用できるパラメータの例を以下に示します。

```
"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
}
```

これらのパラメータとそれぞれに関連する有効な値の詳細については、「CreateCallAnalyticsCategory」および「TranscriptFilter」を参照してください。

非通話時間マッチ

非通話時間 (NonTalkTimeFilter データタイプ) を使用するルールは、以下と一致するように設計されています。

- 通話中、指定された時間帯に無音状態が続いていること

- 通話中、指定された時間帯に発話状態が続いていること

NonTalkTimeFilter で使用できるパラメータの例を以下に示します。

```
"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
},
```

これらのパラメータとそれぞれに関連する有効な値の詳細については、「CreateCallAnalyticsCategory」および「NonTalkTimeFilter」を参照してください。

感情マッチ

感情 (SentimentFilter データタイプ) を使用するルールは、以下と一致するように設計されています。

- 通話中の特定の時点で、お客様、エージェント、あるいはその両方が表現したポジティブな感情の有無
- 通話中の特定の時点で、お客様、エージェント、あるいはその両方が表明したネガティブな感情の有無
- 通話中の特定の時点で、お客様、エージェント、あるいはその両方が表明した中立的な感情の有無
- 通話中の特定の時点で、お客様、エージェント、あるいはその両方が表明したさまざまな感情の有無

SentimentFilter で使用できるパラメータの例を以下に示します。

```
"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 ]
},
```

これらのパラメータとそれぞれに関連する有効な値の詳細については、「CreateCallAnalyticsCategory」および「SentimentFilter」を参照してください。

通話後分析の文字起こしを開始する

通話後分析文字起こしを開始する前に、音声で一致 Amazon Transcribe させるすべてのカテゴリを作成する必要があります。

> **Note**
>
> コール分析トランスクリプトを新しいカテゴリと遡及的に一致させることはできません。コール分析文字起こしを開始する前に作成したカテゴリのみ、その文字起こし出力に適用することができます。

1 つ以上のカテゴリを作成し、音声が少なくとも 1 つのカテゴリですべてのルールに一致する場合、Amazon Transcribe は一致するカテゴリで出力にフラグ付けを行います。カテゴリを使用しないことを選択した場合、または音声がカテゴリで指定したルールに一致しない場合、トランスクリプトにフラグ付けは行われません。

通話後分析文字起こしを開始するには、AWS Management Console、AWS CLI、または AWS SDK を使用できます。例については以下を参照してください。

AWS Management Console

通話後分析ジョブをスタートするには、次の手順を実行します。カテゴリで定義されたすべての特性に一致する通話は、該当するカテゴリでラベル付けされます。

1. ナビゲーションペインの Amazon Transcribe「コール分析」で、「コール分析ジョブ」を選択します。

2. [ジョブの作成] を選択します。

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 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 (1) Info

Search

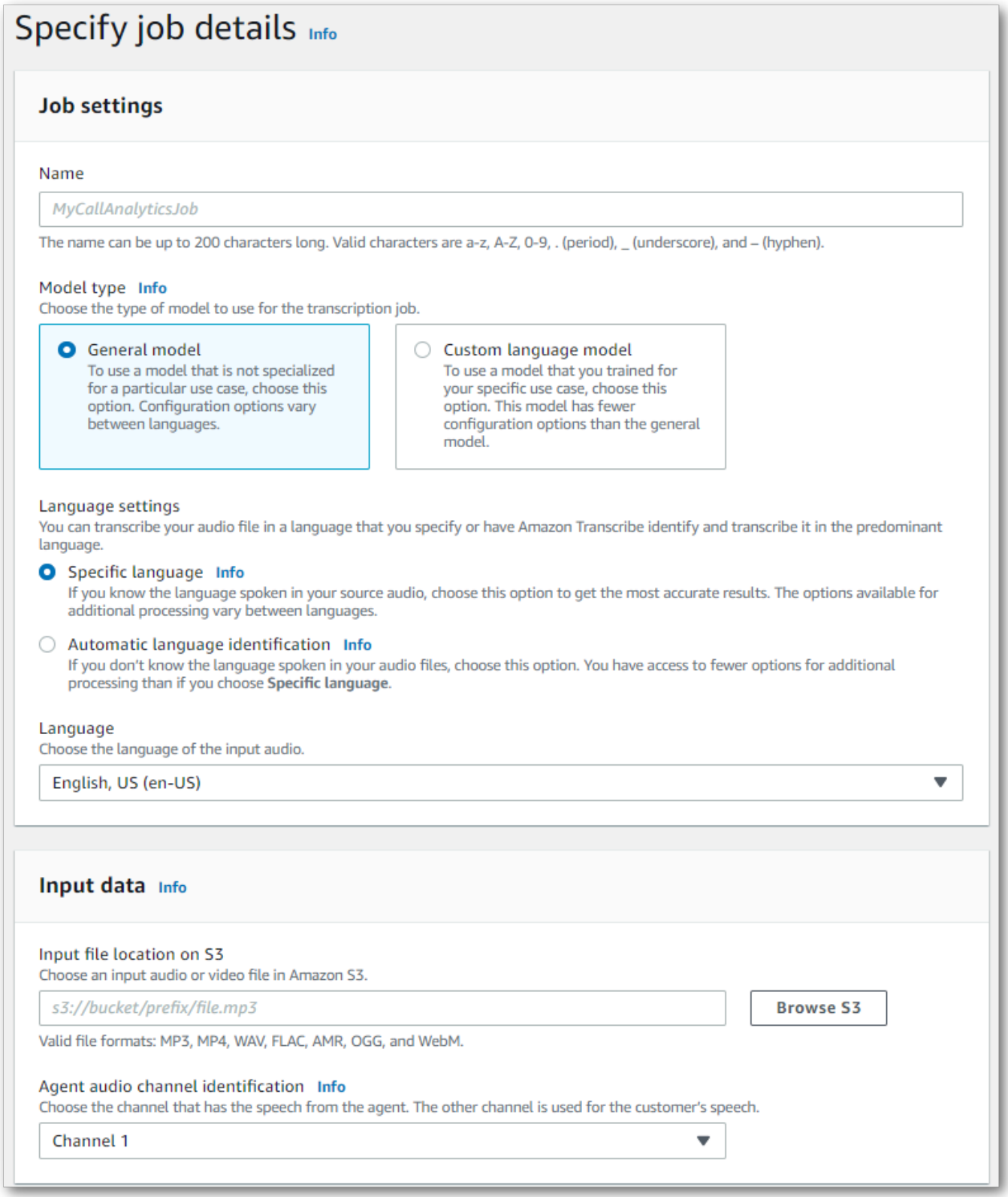
1

	Name	Type	Created	Modified
○	CatchNegativeSentiment	POST_CALL	February 17 2023, 10:43 (UTC-08:00)	February 17 2023, 10:43 (UTC-08:00)

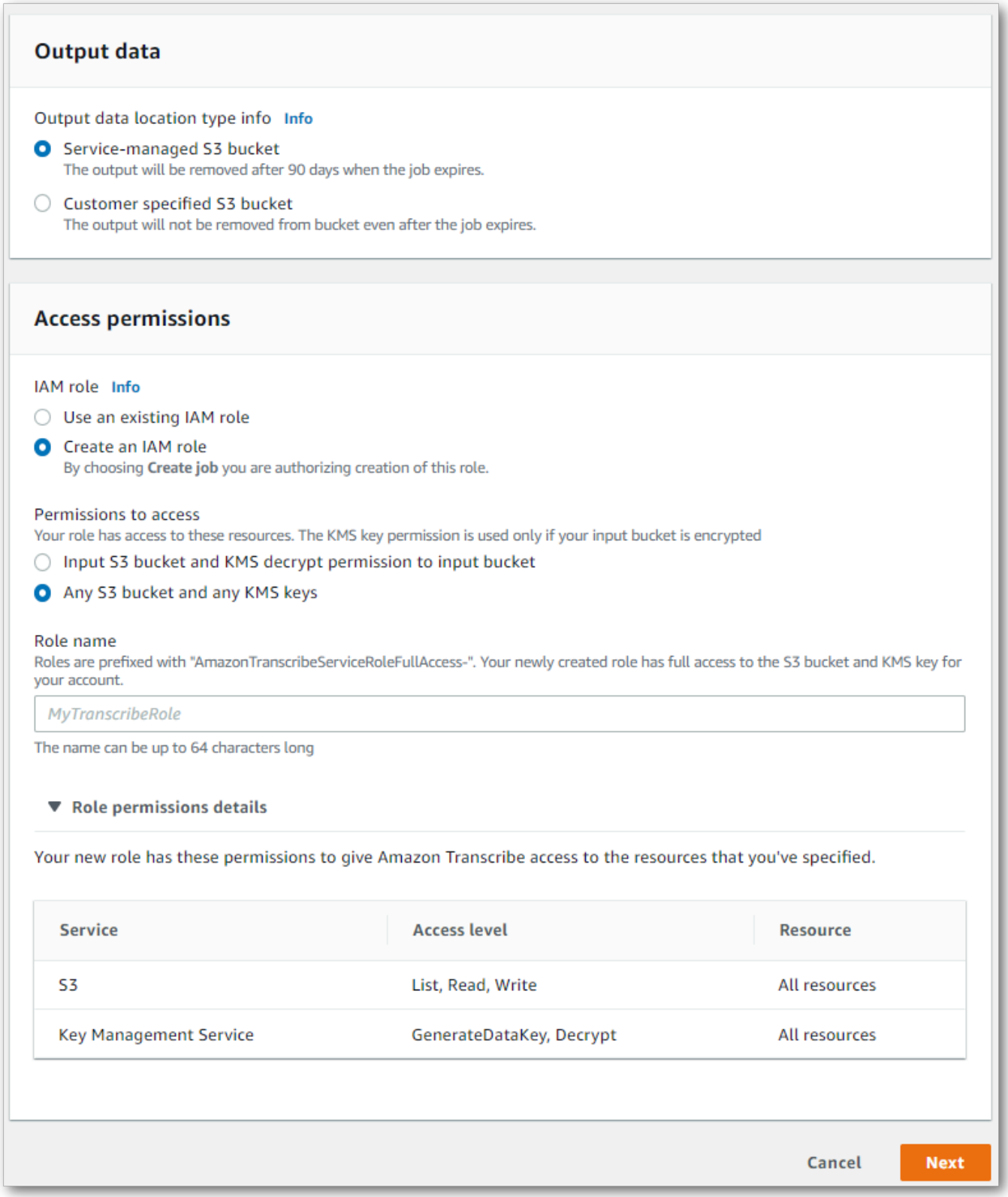
If the above categories aren't relevant to your use case, you can create a new category. Create a new category.

Cancel Previous Create job

3. [ジョブ詳細の指定] ページで、入力データの場所など、コール分析ジョブに関する情報を提供します。



出力データの目的 Amazon S3 の場所と使用する IAM ロールを指定します。



4. [次へ] をクリックします。
5. ジョブの設定で、コール分析ジョブに含めたいオプション機能をオンにします。以前にカテゴリを作成した場合、そのカテゴリはカテゴリパネルに表示され、コール分析ジョブに自動的に適用されます。

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 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 (1) Info

Search

1

	Name	Type	Created	Modified
○	CatchNegativeSentiment	POST_CALL	February 17 2023, 10:43 (UTC-08:00)	February 17 2023, 10:43 (UTC-08:00)

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. [ジョブの作成] を選択します。

AWS CLI

この例では、start-call-analytics-job コマンドと channel-definitionsパラメータを使用します。詳細については、「StartCallAnalyticsJob」および「ChannelDefinition」を参照してください。

```
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
```

コマンドとstart-call-analytics-job、そのジョブのコール分析を有効にするリクエスト本文を使用する別の例を次に示します。

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

ファイル my-call-analytics-job.json には、次のリクエスト本文が含まれています。

```
{
      "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)

この例では、 を使用して AWS SDK for Python (Boto3) start_call_analytics_job メソッドを使用してコール分析ジョブを開始します。詳細については、「StartCallAnalyticsJob」および「ChannelDefinition」を参照してください。

機能固有の例、シナリオ例、クロスサービス例など、 AWS SDKs「」のSDK を使用した Amazon Transcribe のコード例 AWS SDKs章を参照してください。

```
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)
```

通話後分析出力

通話後分析のトランスクリプトは、セグメントごとに turn-by-turn フォーマットで表示されます。これらには、通話の分類、コールの特性 (ラウドネススコア、中断、非通話時間、通話速度)、コールサマリー (問題、結果、アクションアイテム)、リダクション、感情が含まれます。さらに、会話の特徴の概要はトランスクリプトの最後に記載されています。

精度を高め、業界固有の用語など、ユースケースに合わせてトランスクリプトをさらにカスタマイズするには、コール分析リクエストにカスタム語彙またはカスタム言語モデルを使用します。冒涜的な言葉など、文字起こし結果に表示したくない言葉をマスキング、削除、またはタグ付けするには、語彙フィルタリングを追加します。メディアファイルに渡す言語コードがわからない場合は、バッチ言語識別を有効にして、メディアファイルの言語を自動的に識別できます。

以下のセクションでは、インサイトレベルでの JSON 出力の例を示します。コンパイルされた出力については、「コンパイル済み通話後分析出力」を参照してください。

通話の分類

カテゴリマッチは、次のように文字起こし出力で表示されます。この例は、40040 ミリ秒のタイムスタンプから 42460 ミリ秒のタイムスタンプまでの音声が「ポジティブ解決」カテゴリと一致していることを示しています。この場合、カスタムの「ポジティブ解決」カテゴリでは、音声の最後の数秒間はポジティブな感情が必要でした。

```
"Categories": {
    "MatchedDetails": {
        "positive-resolution": {
            "PointsOfInterest": [
                {
                    "BeginOffsetMillis":  40040,
                    "EndOffsetMillis":  42460
                }
            ]
        }
    },
    "MatchedCategories": [
        " positive-resolution"
```

```
    ]
},
```

コールの特性

文字起こし出力では、コールの特性は次のように表示されます。ラウドネススコアは会話のターンごとに表示され、他のすべての特性はトランスクリプトの最後に表示されます。

```
"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
    }
},
```

問題、アクション項目、次のステップ

- 次の例では、問題は文字 7 から始まり、文字 51 で終わるものとして識別されます。これは、「レシピのサブスクリプションをキャンセルしたい」というテキストのこのセクションを指しています。

```
"Content": "Well, I would like to cancel my recipe subscription.",

"IssuesDetected": [
    {
        "CharacterOffsets": {
            "Begin": 7,
```

```
            "End": 51
        }
    }
],
```

- 次の例では、結果は文字 12 で始まり、文字 78 で終わると識別されます。これは、「アカウントにすべての変更を加えたので、この割引が適用されました」というテキストのこのセクションを指します。

```
"Content": "Wonderful. I made all changes to your account and now this discount is
 applied, please check.",

"OutcomesDetected": [
    {
        "CharacterOffsets": {
            "Begin": 12,
            "End": 78
        }
    }
],
```

- 次の例では、アクションアイテムは文字 0 で始まり、文字 103 で終わるように識別されます。これは、「本日、すべての詳細を記載したメールをお送りします。フォローアップのため、来週折り返しお電話させていただきます。」というテキストのこのセクションを指します。

```
"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
        }
    }
],
```

通話の生成要約

通話の生成要約は、トランスクリプション出力に次のように表示されます。

```
"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."
        }
    }
}
```

次の場合、分析ジョブはサマリー生成なしで完了します。

- 会話コンテンツが不十分: 会話には、エージェントと顧客の両方からのターンが少なくとも 1 回含まれている必要があります。会話コンテンツが不十分な場合、サービスはエラーコード INSUFFICIENT_CONVERSATION_CONTENT を返します。
- 安全ガードレール: 会話は、適切な概要が生成されるように、所定の安全ガードレールを満たす必要があります。これらのガードレールが満たされない場合、サービスはエラーコード FAILED_"TY_GUIDELINES を返します。

エラーコードは、出力の 内の Skipped セクションAnalyticsJobDetailsにあります。また、GetCallAnalyticsJob API レスポンスの CallAnalyticsJobDetailsにエラーの理由が表示される場合もあります。

サンプルエラー出力

```
{
    "JobStatus": "COMPLETED",
    "AnalyticsJobDetails": {
        "Skipped": [
            {
                "Feature": "GENERATIVE_SUMMARIZATION",
                "ReasonCode": "INSUFFICIENT_CONVERSATION_CONTENT",
                "Message": "The conversation needs to have at least one turn from both
 the participants to generate summary"
            }
        ]
    },
    "LanguageCode": "en-US",
    "AccountId": "***************",
    "JobName": "Test2-copy",
    ...
```

```
}
```

感情分析

感情分析は、トランスクリプション出力に次のように表示されます。

- 定性的 turn-by-turn 感情値：

```
"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"
```

- コール全体の定量的感情値

```
"Sentiment": {
    "OverallSentiment": {
        "AGENT": 2.5,
        "CUSTOMER": 2.1
    },
```

- 参加者一人当たりおよびコール四半期ごとの定量的感情値

```
"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 のマスキング

PII のマスキングは、トランスクリプション出力に次のように表示されます。

```
"Content": "[PII], my name is [PII], how can I help?",
"Redaction": [{
    "Confidence": "0.9998",
    "Type": "NAME",
    "Category": "PII"
}]
```

詳細については、「バッチジョブの PII のマスキング」を参照してください。

言語識別

言語識別機能が有効になっている場合、言語識別はトランスクリプション出力に次のように表示されます。

```
"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"
}]
```

上記の出力例では、言語識別によって言語コードが信頼度スコアと共に表示されます。スコアが最も高い結果が、トランスクリプションの言語コードとして選択されます。モードの詳細については、「メディアの主要言語の特定」を参照してください。

コンパイル済み通話後分析出力

簡潔にするために、次の文字起こし出力では一部の内容が省略記号に置き換えられています。

このサンプルには、オプションの機能 - 生成通話の要約が含まれています。


```
{
    "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
        },
        "ContactSummary": { // Optional feature - Generative call summarization
            "AutoGenerated": {
                "OverallSummary": {
                    "Content": "The customer initially wanted to cancel but the agent
 convinced them to stay by offering a 50% discount, which the customer accepted after
 reconsidering cancelling given the significant savings. The agent ensured the discount
 was applied and said they would follow up to ensure the customer remained happy with
 the revised subscription."
                }
            }
        }
    },
    "AnalyticsJobDetails": {
        "Skipped": []
    },
    ...
}
```

通話の生成要約の有効化

> **Note**
>
> Amazon Bedrock を搭載： AWS 自動不正使用検出を実装。生成 AI によるコンタクト後の要約は Amazon Bedrock 上に構築されているため、ユーザーは Amazon Bedrock に実装されている制御を最大限に活用して、人工知能 (AI) の安全性、セキュリティ、責任ある使用を適用できます。

通話の生成要約を通話後の分析ジョブで使用するには、以下の例を参照してください。

AWS Management Console

[要約] パネルで、[通話の生成要約] を有効にすると、概要が出力に表示されます。

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 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 (1) Info

Search

1

	Name	Type	Created	Modified
○	CatchNegativeSentiment	POST_CALL	February 17 2023, 10:43 (UTC-08:00)	February 17 2023, 10:43 (UTC-08:00)

If the above categories aren't relevant to your use case, you can create a new category. Create a new category.

AWS CLI

この例では、 Summarization サブSettingsパラメータで start-call-analytics-job コマンドと パラメータを使用します。詳細については、「StartCallAnalyticsJob」を参照してください。

```
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}}'
```

コマンドとstart-call-analytics-job、そのジョブの要約を有効にするリクエスト本文を使用する別の例を次に示します。

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

ファイル my-call-analytics-job.json には、次のリクエスト本文が含まれています。

```
{
  "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)

この例では、 を使用して AWS SDK for Python (Boto3) 、start_call_analytics_job メソッドを使用して要約を有効にしたコール分析を開始します。詳細については、「StartCallAnalyticsJob」を参照してください。

機能固有の例、シナリオ例、クロスサービス例など、 AWS SDKs「」のSDK を使用した Amazon Transcribe のコード例 AWS SDKs章を参照してください。

```
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)
```

リアルタイムコール分析

リアルタイムコール分析では、問題への対処や問題の発生時のエスカレーションの緩和に役立つリアルタイムのインサイトが得られます。

リアルタイムコール分析では、次のようなインサイトが得られます。

- ルールを使用して特定のキーワードやフレーズにフラグを付ける[カテゴリイベント]; カテゴリイベントを使用して[リアルタイムアラート]を作成する
- [問題検出]では、各音声セグメント内で話されている問題を特定する
- テキストトランスクリプト内の [PII (機密データ) の識別]
- テキストトランスクリプト内の [PII (機密データ) リダクション]
- 各音声セグメントの[感情分析]

リアルタイムコール分析に加えて、 Amazon Transcribe はメディアストリームで[通話後分析]を実行することもできます。[PostCallAnalyticsSettings] パラメータを使用して、通話後分析をリアルタイムコール分析リクエストに含めることができます。

リアルタイムインサイト

このセクションでは、リアルタイムコール分析の文字起こしで得られるインサイトについて詳しく説明します。

イベントカテゴリ

カテゴリイベントを使用すると、正確なキーワードまたはフレーズに基づいて文字起こしを一致させることができます。例えば、「マネージャーと話したい」というフレーズにフィルターを設定すると、その正確なフレーズが Amazon Transcribe フィルタリングされます。

以下は出力例です。

リアルタイムコール分析カテゴリの作成について詳しくは、「リアルタイム文字起こしのカテゴリの作成」を参照してください。

> **Tip**
>
> カテゴリイベントでは、リアルタイムアラートを設定できます。詳細については、「カテゴリマッチに関するリアルタイムアラートの作成」を参照してください。

問題検出

問題検出では、各音声セグメント内で検出された問題の簡潔な概要が表示されます。問題検出機能を使うと、以下のことができます。

- 通話中や通話後に手動でメモを取る手間を減らします。
- エージェントの効率を向上させて、お客様への対応を迅速に行えるようにします。

> **Note**
>
> 問題検出は、オーストラリア (en-AU)、英国 (en-GB)、米国 (en-US) の英語の方言でサポートされています。

問題検出機能はあらゆる業界や業種に対応し、コンテキストに基づいています。機能 out-of-the-box するため、モデルトレーニングやカスタムカテゴリなどのカスタマイズはサポートされていません。

リアルタイムコール分析による問題検出は、完全な音声セグメントごとに実行されます。

以下は出力例です。

PII (機密データ) の識別

機密データ識別は、テキストトランスクリプト内の個人を特定できる情報 (PII) をラベル付けします。このパラメータは、お客様の情報を保護するのに役立ちます。

> **Note**
>
> リアルタイムの PII 識別は、オーストラリア (en-AU) 、英国 ()、米国 (en-GB) 、スペイン語 (en-US) の英語の方言でサポートされていますes-US。

リアルタイムコール分析による PII 識別は、完全な音声セグメントごとに実行されます。

この機能を使用して識別される PII のリストを表示する場合、または を使用した PII 識別の詳細については Amazon Transcribe、「」を参照してください個人を特定できる情報の編集または特定。

以下は出力例です。

PII (機密データ) リダクション

機密データのリダクションでは、テキストトランスクリプトの個人を特定できる情報 (PII) が PII のタイプ (例: [NAME]) に置き換えられます。このパラメータは、お客様の情報を保護するのに役立ちます。

> **Note**
>
> リアルタイムの PII リダクションは、オーストラリア (en-AU) 、英国 ()、米国 (en-GB) 、スペイン語 (en-US) の英語の方言でサポートされていますes-US。

リアルタイムコール分析による PII リダクションは、完全な音声セグメントごとに実行されます。

この機能を使用して編集された PII のリストの表示、または Amazon Transcribeでのリダクションの詳細については、「個人を特定できる情報の編集または特定」を参照してください。

以下は出力例です。

感情分析

感情分析では、コール全体を通してカスタマーとエージェントがどのように感じているかを推定します。このメトリックは音声セグメントごとに提供され、定性値 (positive、neutral、mixed、または negative) で表されます。

このパラメータを使用すると、各通話参加者の全体的な感情と、各音声セグメントの各参加者の感情を定性的に評価できます。このメトリクスは、コールが終了するまでに、エージェントが怒っているカスタマーを喜ばすことができるかどうかを識別するのに役立ちます。

リアルタイムコール分析による感情分析は、完全な音声セグメントごとに実行されます。

感情分析は機能 out-of-the-box するため、モデルトレーニングやカスタムカテゴリなどのカスタマイズはサポートされていません。

以下は出力例です。

リアルタイム文字起こしのカテゴリの作成

リアルタイムコール分析はカスタムカテゴリの作成をサポートしており、これを使用して特定のビジネスニーズに合わせてトランスクリプト分析を調整できます。

さまざまなシナリオをカバーするカテゴリをいくつでも作成できます。作成するカテゴリごとに、1 から 20 のルールを作成する必要があります。リアルタイムコール分析の文字起こしでは、`TranscriptFilter` (キーワードマッチ) を使用するルールのみがサポートされます。`CreateCallAnalyticsCategory` オペレーションでルールを使用する詳細については、「リアルタイムコール分析カテゴリのルールの条件 セクション」を参照してください。

メディア内のコンテンツが、指定したカテゴリのすべてのルールに一致する場合、 Amazon Transcribe は出力にそのカテゴリのラベル付けを行います。JSON 出力形式のカテゴリマッチの例については、「カテゴリイベント出力」を参照してください。

カスタムカテゴリを使用してできるその他の例を紹介します。

- 特定のキーワードセットにフラグを付けて追跡することで、早急な対応が必要な問題を特定できます。
- エージェントが特定のフレーズを話す (または省略) など、コンプライアンスをモニタリングする
- 特定の単語やフレーズにリアルタイムでフラグを付け、カテゴリマッチを設定して即時アラートを設定できます。たとえば、「マネージャーと話す」と言うお客様についてのリアルタイムコール分

析カテゴリを作成した場合、このリアルタイムのカテゴリマッチに対して、勤務中のマネージャーに通知するイベントアラートを設定できます。

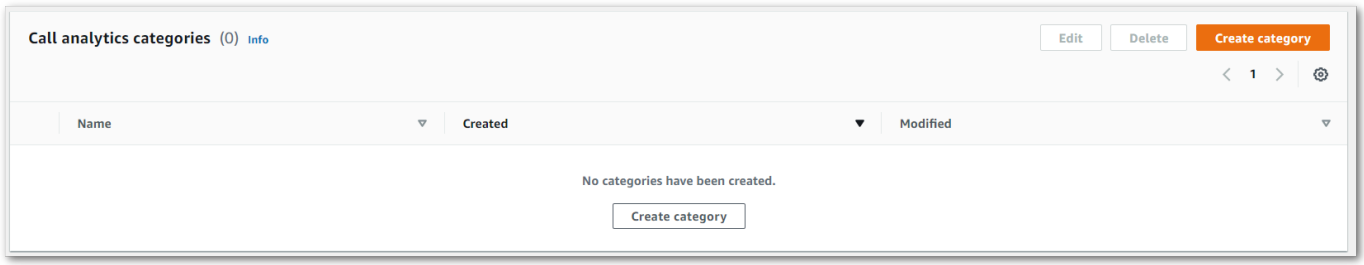
通話後カテゴリとリアルタイムカテゴリ

新しいカテゴリを作成する場合、通話後カテゴリ (POST_CALL) として作成するか、リアルタイムカテゴリ (REAL_TIME) として作成するかを指定できます。オプションを指定しない場合、カテゴリはデフォルトで通話後カテゴリとして作成されます。リアルタイムのカテゴリマッチは、リアルタイムのアラートを作成するために使用することができます。詳細については、「カテゴリマッチに関するリアルタイムアラートの作成」を参照してください。

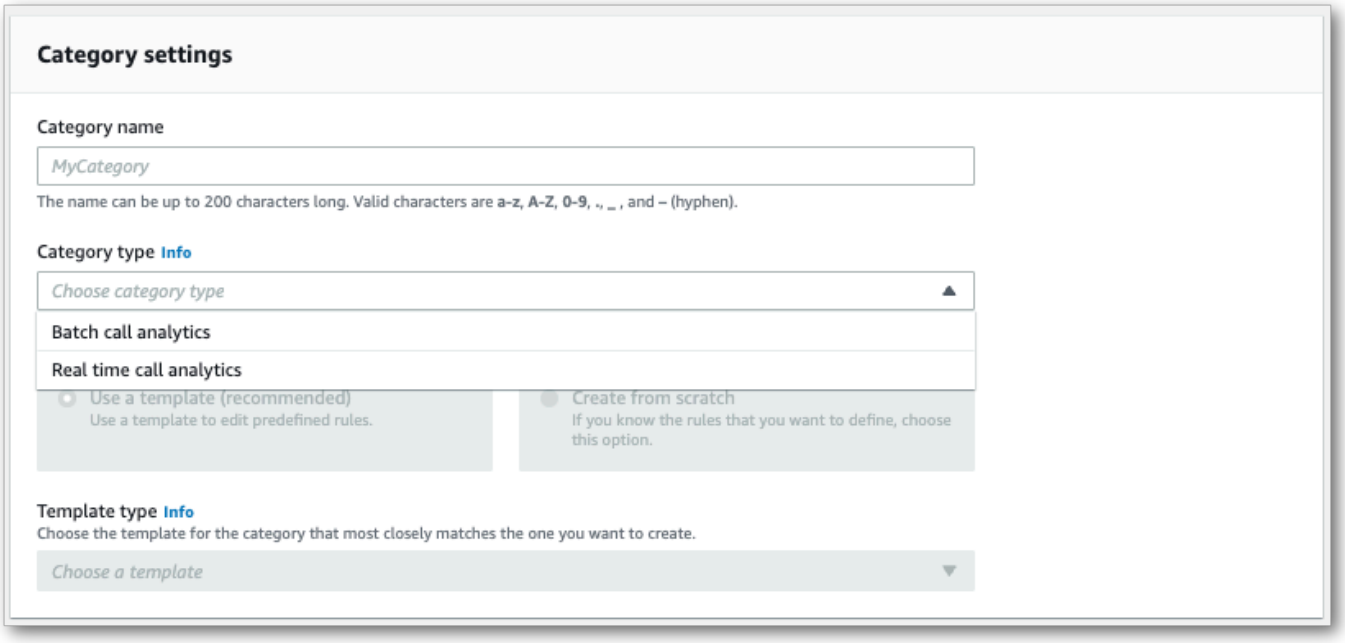
リアルタイムコール分析の新しいカテゴリを作成するには、AWS Management Console、AWS CLI、または AWS SDK を使用できます。例については以下を参照してください。

AWS Management Console

1. ナビゲーションペインの で Amazon Transcribe、Amazon Transcribe 分析の呼び出し を選択します。
2. [コール分析カテゴリ] を選択すると、[コール分析カテゴリ] ページに移動します。「カテゴリの作成」ボタンを選択します。

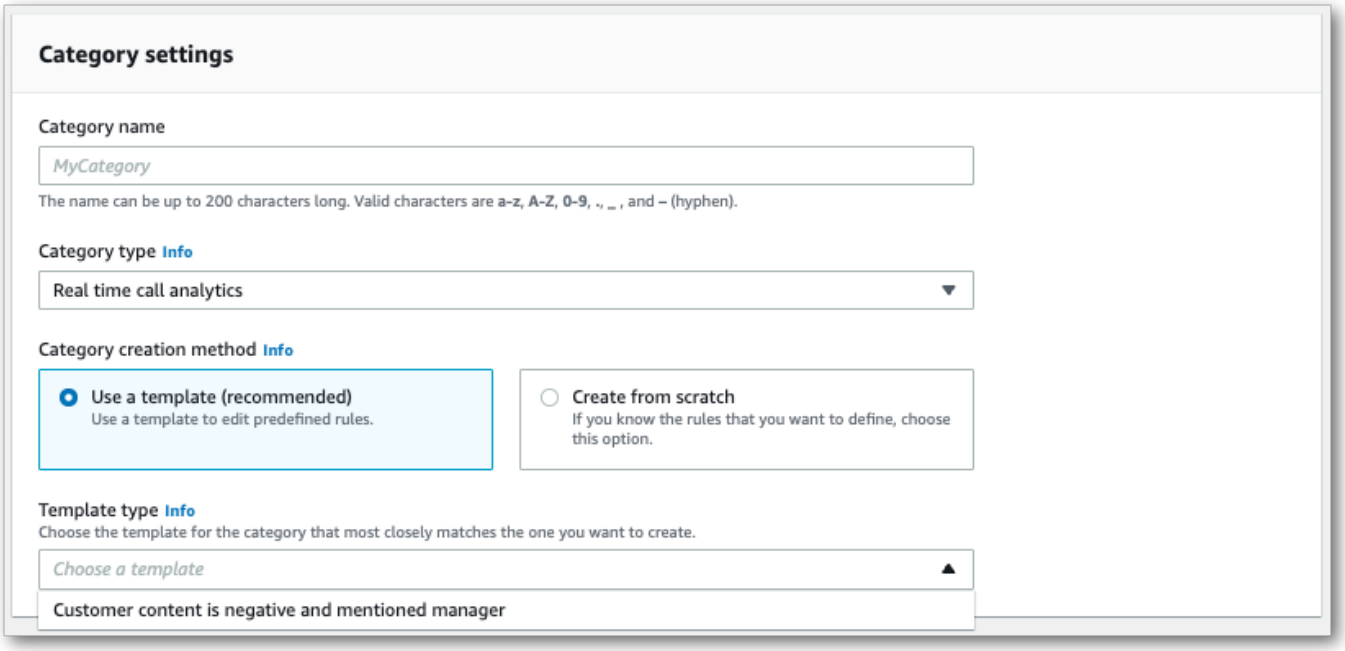


3. [カテゴリの作成ページ] が表示されます。カテゴリの名前を入力し、カテゴリタイプのドロップダウンメニューで [リアルタイムコール分析] を選択します。

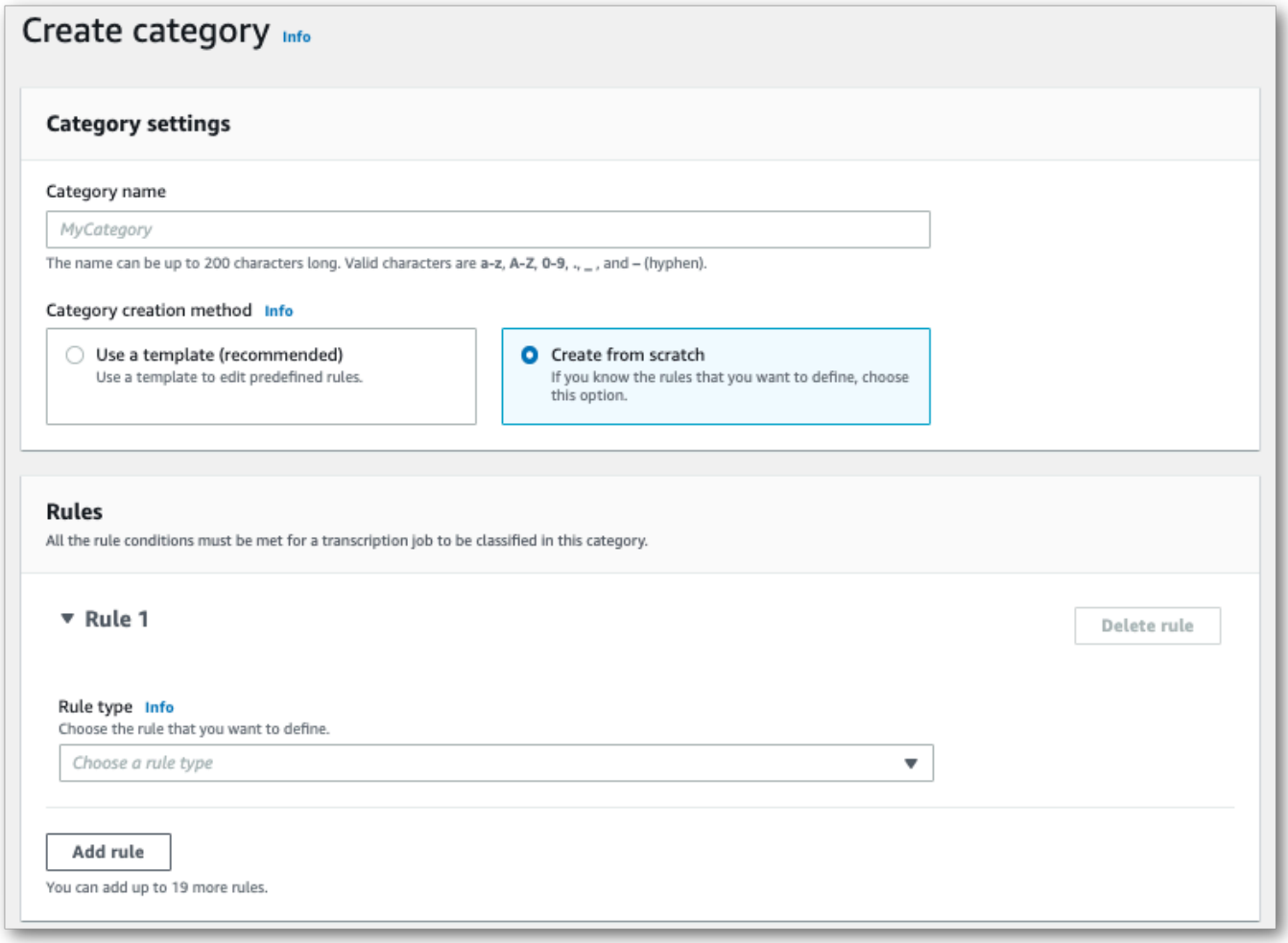


4. テンプレートを選択してカテゴリを作成することも、一から作成することもできます。

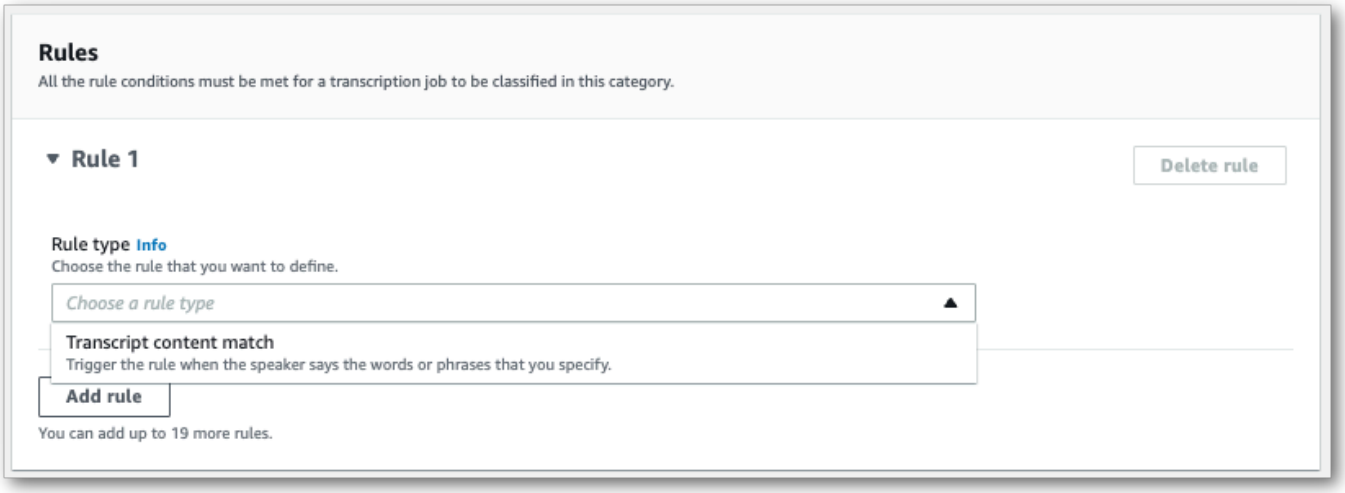
   テンプレートを使用する場合: [テンプレートを使用する (推奨)] を選択し、必要なテンプレートを選択してから [カテゴリの作成] を選択します。



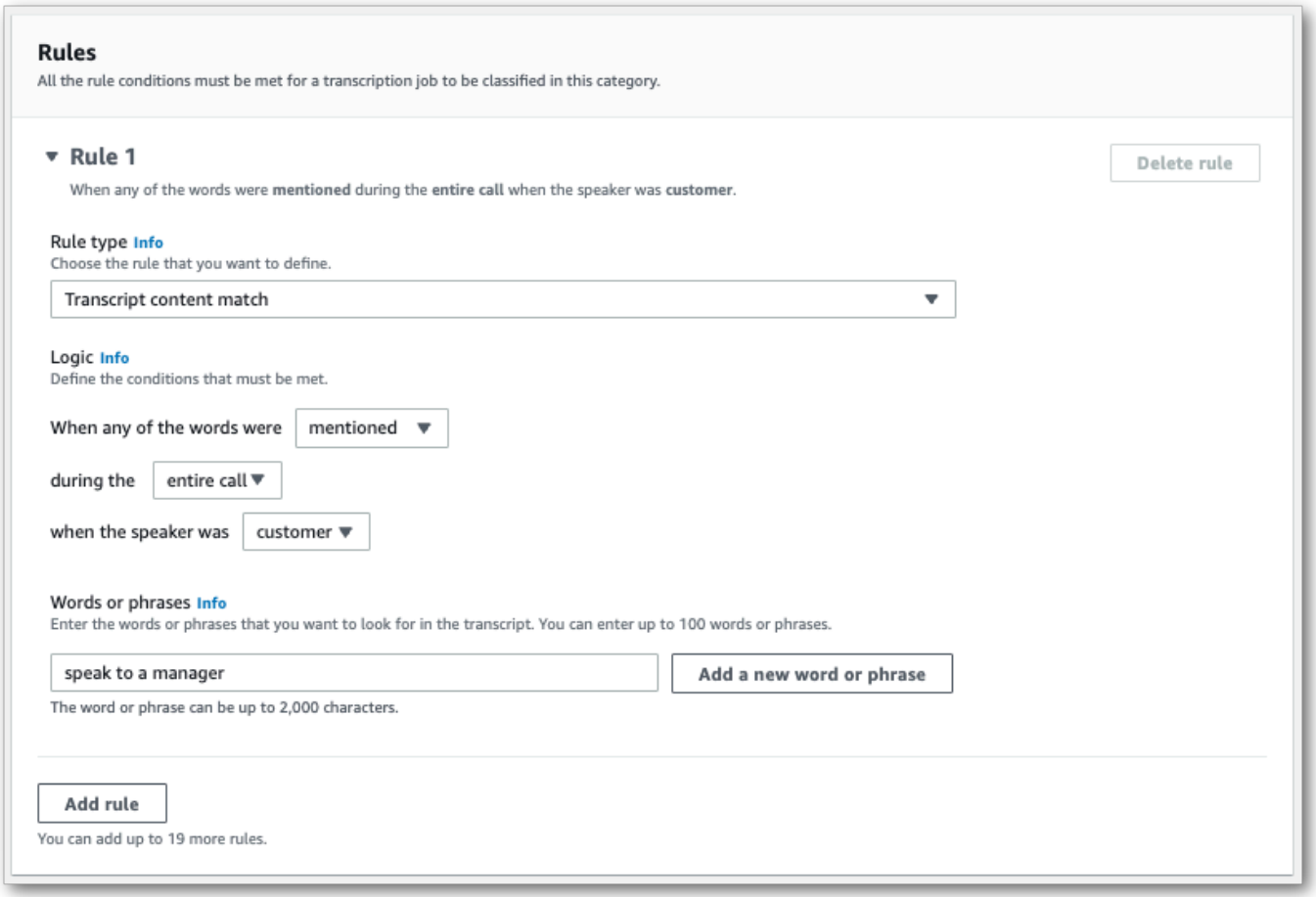
5. カスタムカテゴリを作成する場合: [最初から作成] を選択します。



6. ドロップダウンメニューを使用して、カテゴリにルールを追加します。1 つのカテゴリには最大 20 ルールまで追加できます。リアルタイムコール分析文字起こしでは、トランスクリプトの内容が一致するルールのみを含めることができます。一致した場合はリアルタイムでフラグが付けられます。



7. ルールが 1 つあるカテゴリの例を次に示します。お客様が通話中どの時点でも「マネージャーと話す」と言っている場合です。



8. カテゴリにルールを追加し終えたら、[カテゴリの作成] を選択します。

AWS CLI

この例では、 create-call-analytics-category コマンドを使用します。詳細については、「CreateCallAnalyticsCategory」、「CategoryProperties」、および「Rule」を参照してください。

以下の例では、ルールを含むカテゴリを作成します。

- お客様は、通話のどの時点でも「マネージャーと話す」というフレーズを口にしました。

この例では、 create-call-analytics-category コマンドと、カテゴリにルールを追加するリクエスト本文を使用します。

```
aws transcribe create-call-analytics-category \
--cli-input-json file://filepath/my-first-analytics-category.json
```

ファイル my-first-analytics-category.json には、次のリクエスト本文が含まれています。

```
{
  "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)

この例では、 を使用して AWS SDK for Python (Boto3) 、create_call_analytics_category メソッドの 引数CategoryNameと Rules引数を使用してカテゴリを作成します。詳細については、CreateCallAnalyticsCategory、CategoryProperties、およびRuleを参照してください。

機能固有の例、シナリオ例、クロスサービス例など、 AWS SDKs「」のSDK を使用した Amazon Transcribe のコード例 AWS SDKs章を参照してください。

以下の例では、ルールを含むカテゴリを作成します。

- お客様は、通話のどの時点でも「マネージャーと話す」というフレーズを口にしました。

```
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)
```

リアルタイムコール分析カテゴリのルールの条件

このセクションでは、CreateCallAnalyticsCategory API オペレーションを使用して作成できるカスタム REAL_TIME ルールのタイプについて説明します。

問題検出は自動的に行われるため、問題にフラグを付けるためのルールやカテゴリを作成する必要はありません。

リアルタイムコール分析の文字起こしでは、キーワードマッチのみがサポートされます。中断、無音、感情を含むカテゴリを作成したい場合は、「通話後分析カテゴリのルールの条件」を参照してください。

キーワードマッチ

キーワード (TranscriptFilter データタイプ) を使用するルールは、以下と一致するように設計されています。

- エージェント、お客様、あるいはその両方が話すカスタム単語またはフレーズ
- エージェント、お客様、あるいはその両方が口にしないカスタム単語またはフレーズ
- 特定の期間に出現するカスタム単語またはフレーズ

TranscriptFilter で使用できるパラメータの例を以下に示します。

```
"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
}
```

これらのパラメータとそれぞれに関連する有効な値の詳細については、「CreateCallAnalyticsCategory」および「TranscriptFilter」を参照してください。

通話後分析とリアルタイム文字起こし

通話後分析は、リアルタイムコール分析文字起こしで利用できるオプション機能です。通話後分析では、標準のリアルタイム分析インサイトに加えて、次の情報も得られます。

- アクションアイテム: 通話中に特定されたアクションアイテムをすべて一覧表示
- 中断: 一方の参加者がもう一方の参加者の発言を途中で中断したかどうかと、そのタイミングを測定
- 問題: 通話中に特定された問題が表示される

- ラウドネス: 各参加者が話している音量を測定
- 非通話時間: 音声が含まれていない時間を測定
- 結果: 通話中に特定された結果または解決策を提供
- 通話速度: 両方の参加者が話している速度を測定
- 通話時間: 通話中に各参加者が通話した時間 (ミリ秒単位) を測定

有効にすると、オーディオストリームからの通話後分析は、オーディオファイルからの通話後分析と同様のトランスクリプトを生成し、 で指定された Amazon S3 バケットに保存しますOutputLocation。さらに、通話後分析はオーディオストリームを記録し、同じ Amazon S3 バケットにオーディオファイル (WAV 形式) として保存します。秘匿化を有効にすると、秘匿化されたトランスクリプトと秘匿化されたオーディオファイルも指定された Amazon S3 バケットに保存されます。音声ストリームで通話後分析を有効にすると、以下に説明するように 2～4 つのファイルが作成されます。

- リダクションが有効になっていない場合、出力ファイルは次のようになります。
  1. 未編集のトランスクリプト
  2. 未編集の音声ファイル
- 未編集オプション (redacted) なしでリダクションを有効にすると、出力ファイルは次のようになります。
  1. 編集済みトランスクリプト
  2. 編集済み音声ファイル
- 未編集オプション (redacted_and_unredacted) ありでリダクションを有効にすると、出力ファイルは次のようになります。
  1. 編集済みトランスクリプト
  2. 編集済み音声ファイル
  3. 未編集のトランスクリプト
  4. 未編集の音声ファイル

リクエストで通話後分析 (PostCallAnalyticsSettings) を有効にしていて、FLAC または OPUS-OGG メディアを使用している場合、トランスクリプトに loudnessScore は表示されず、ストリームの音声録音も作成されないことに注意してください。Transcribe は、90 分以上続く長時間実行されるオーディオストリームに対して通話後分析を提供できない場合もあります。

音声ストリームの通話後分析で得られるインサイトについて詳しくは、「通話後分析インサイト」セクションを参照してください。

> **Tip**
>
> リアルタイムコール分析リクエストで通話後分析を有効にすると、すべての POST_CALL および REAL-TIME カテゴリが通話後分析の文字起こしに適用されます。

通話後分析を有効にする

通話後分析を有効にするには、リアルタイムコール分析リクエストに PostCallAnalyticsSettings パラメータを含める必要があります。PostCallAnalyticsSettings を有効にする場合は、次のパラメータを含める必要があります。

- OutputLocation: 通話後のトランスクリプトを保存する Amazon S3 バケット。
- DataAccessRoleArn: 指定した Amazon S3 バケットへアクセスする権限を持つ Amazon S3 ロールの Amazon リソースネーム (ARN)。リアルタイム分析の信頼ポリシーも使用する必要があることに注意してください。

トランスクリプトの編集版が必要な場合は、リクエストに ContentRedactionOutput または ContentRedactionType を含めることができます。これらのパラメータの詳細については、「API リファレンス」の「StartCallAnalyticsStreamTranscription」を参照してください。

通話後分析を有効にしてリアルタイムコール分析文字起こしを開始するには、 AWS Management Console (デモのみ) 、HTTP/2、または を使用できますWebSockets。例については、「リアルタイムコール分析の文字起こしを開始する」を参照してください。

> **Important**
>
> 現在、 AWS Management Console は、オーディオサンプルがプリロードされたリアルタイムコール分析のデモのみを提供しています。独自のオーディオを使用する場合は、 API (HTTP/2、、 WebSocketsまたは SDK) を使用する必要があります。

通話後分析の出力例

通話後トランスクリプトはセグメントごとに turn-by-turn フォーマットで表示されます。これらには、コールの特性、感情、コールサマリー、問題検出、および (オプションで) PII リダクションが含まれます。通話後カテゴリのいずれかが音声コンテンツと一致する場合、そのカテゴリも出力に含まれます。

精度を高め、業界固有の用語など、ユースケースに合わせてトランスクリプトをさらにカスタマイズするには、コール分析リクエストにカスタム語彙またはカスタム言語モデルを使用します。冒涜的な言葉など、文字起こし結果に表示したくない言葉をマスキング、削除、またはタグ付けするには、語彙フィルタリングを追加します。

以下に、通話後分析の出力例を示します。

```
{
    "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
```

```
        }
    },
    ...
}
```

リアルタイムコール分析の文字起こしを開始する

リアルタイムコール分析文字起こしを開始する前に、通話で一致 Amazon Transcribe させるすべてのカテゴリを作成する必要があります。

> **Note**
>
> コール分析トランスクリプトを新しいカテゴリと遡及的に一致させることはできません。コール分析文字起こしを開始する前に作成したカテゴリのみ、その文字起こし出力に適用することができます。

1 つ以上のカテゴリを作成し、音声が少なくとも 1 つのカテゴリですべてのルールに一致する場合、Amazon Transcribe は一致するカテゴリで出力にフラグ付けを行います。カテゴリを使用しないことを選択した場合、または音声がカテゴリで指定したルールに一致しない場合、トランスクリプトにフラグ付けは行われません。

通話後分析をリアルタイムコール分析文字起こしに含めるには、`OutputLocation` パラメータを使用してリクエストに Amazon S3 バケットを指定する必要があります。また、指定したバケットへの書き込み権限を持つ `DataAccessRoleArn` も含める必要があります。リアルタイムコール分析ストリーミングセッションが完了すると、別のトランスクリプトが作成され、指定されたバケットに保存されます。

リアルタイムコール分析では、リアルタイムカテゴリアラートを作成するするオプションもあります。手順については、「カテゴリマッチに関するリアルタイムアラートの作成」を参照してください。

リアルタイムコール分析文字起こしを開始するには、、AWS Management ConsoleHTTP/2、またはを使用できますWebSockets。例については、以下を参照してください。

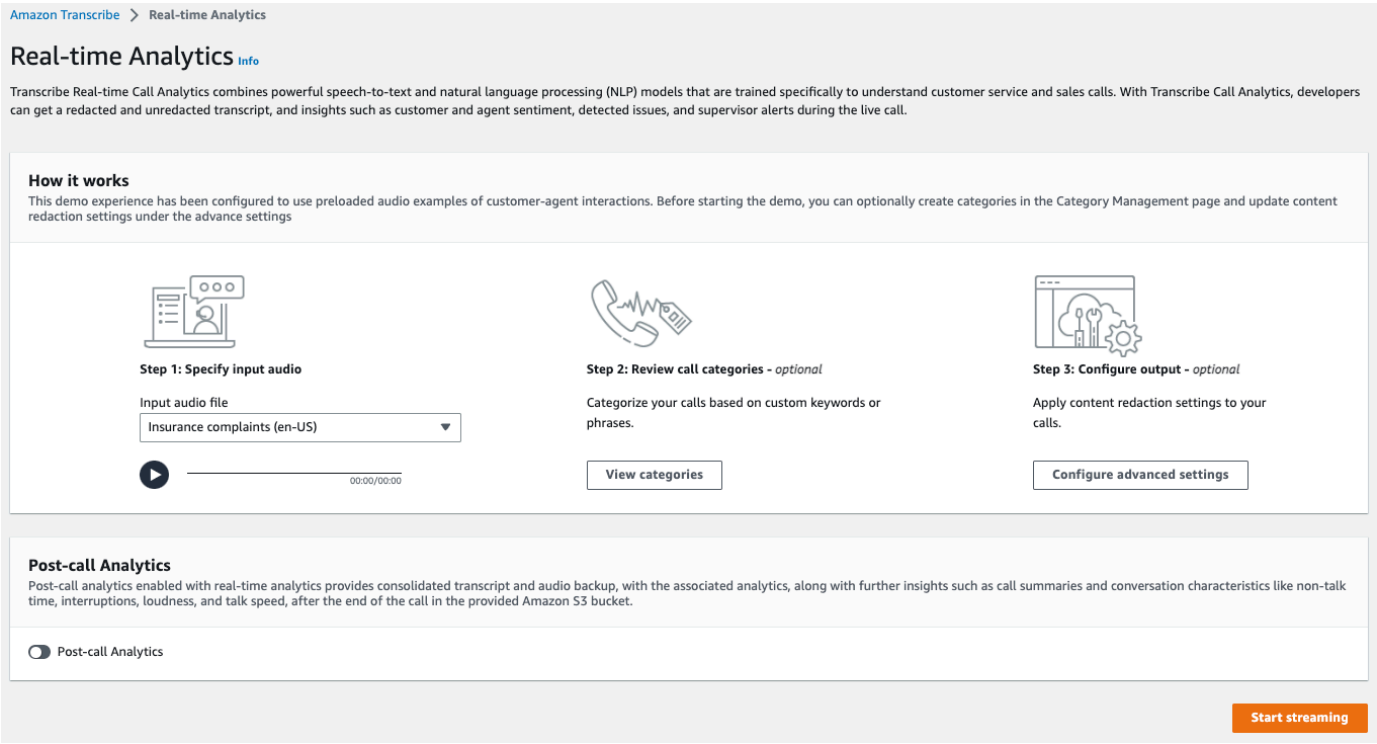
> **Important**
>
> 現在、 AWS Management Console は、オーディオサンプルがプリロードされたリアルタイムコール分析のデモのみを提供しています。独自のオーディオを使用する場合は、 API (HTTP/2、、 WebSocketsまたは SDK) を使用する必要があります。

AWS Management Console

コール分析リクエストをスタートするには、次の手順を実行します。カテゴリで定義されたすべての特性に一致する通話は、該当するカテゴリでラベル付けされます。

> **Note**
>
> AWS Management Consoleではデモのみが提供されています。カスタムリアルタイム分析文字起こしを開始するには、API を使用する必要があります。

1. ナビゲーションペインの Amazon Transcribe 「コール分析」で、「リアルタイムコールの分析」を選択します。



2. ステップ 1: 入力音声の指定では、ドロップダウンメニューから [デモテストファイル] を選択します。

Step 1: Specify input audio

Input audio file

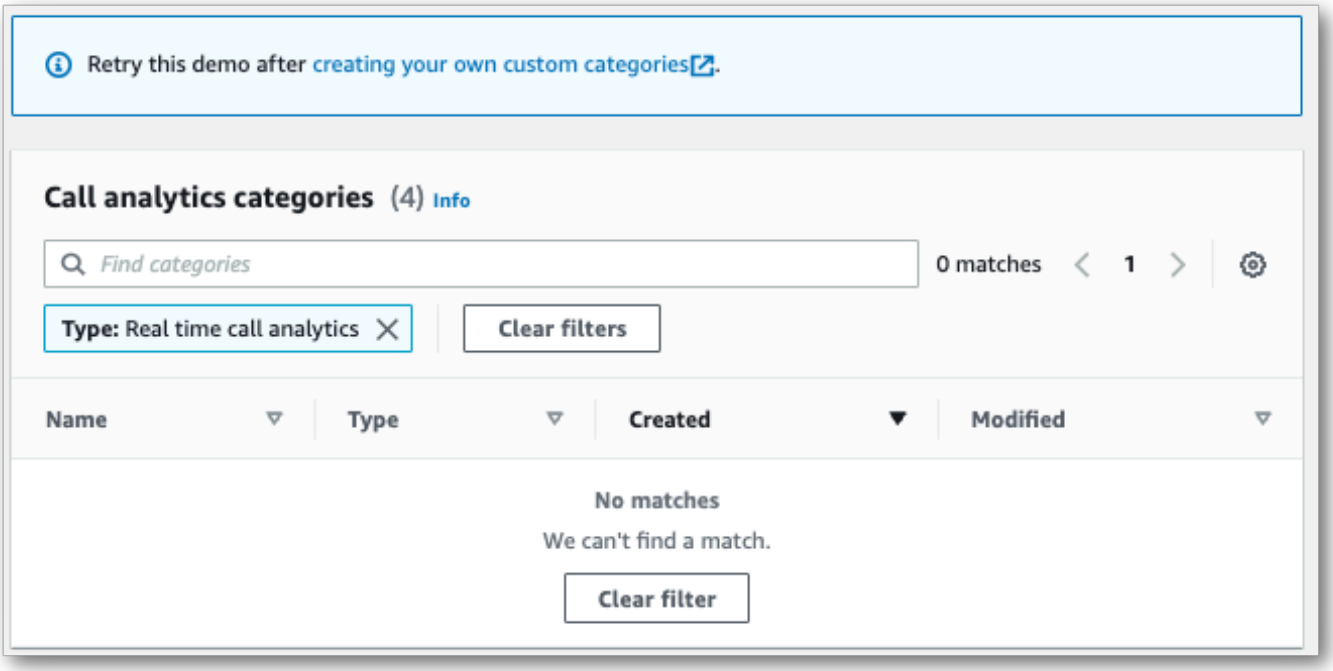
Insurance complaints (en-US)

Insurance complaints (en-US)

Hospitality complaints (en-US)

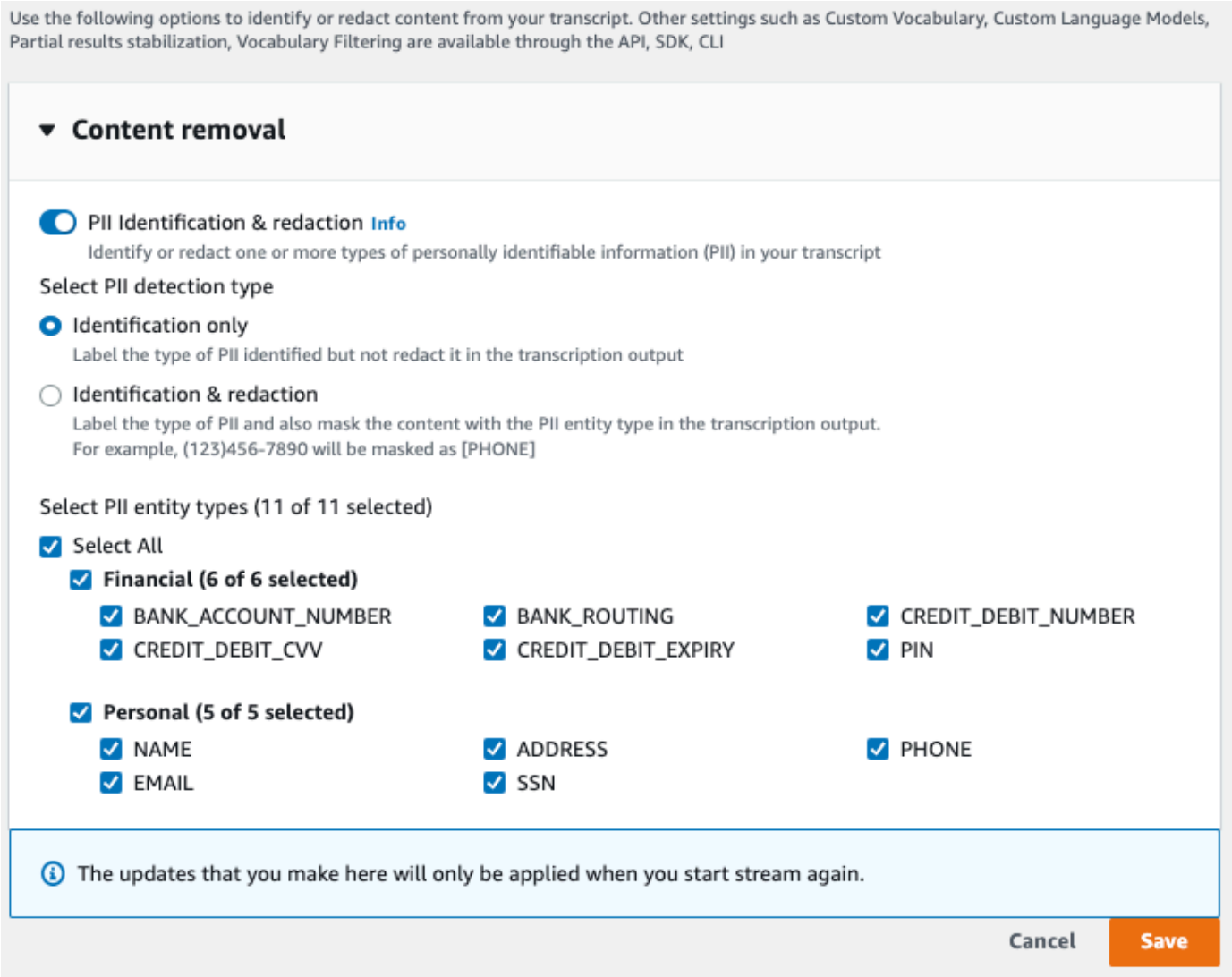
3. ステップ 2: 通話カテゴリの確認では、以前に作成したリアルタイムコール分析カテゴリを確認するオプションがあります。リアルタイムコール分析カテゴリはすべて、文字起こしに適用されます。

   [カテゴリを表示] を選択すると、新しいペインが開き、既存のリアルタイムコール分析カテゴリが表示され、新しいカテゴリを作成するためのリンクが表示されます。



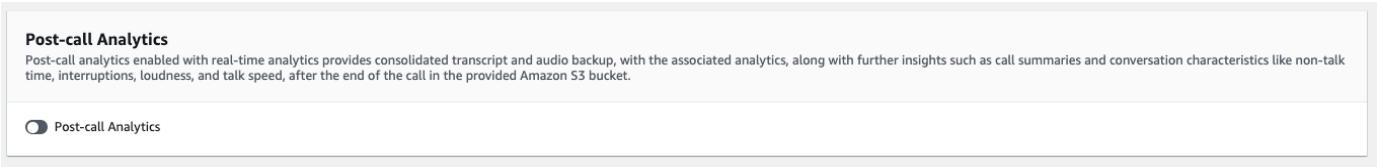
4. ステップ 3: 入力と出力を設定するでは、追加の設定を適用するオプションがあります。

   [詳細設定の構成] を選択すると、コンテンツリダクション設定を指定できる新しいペインが開きます。

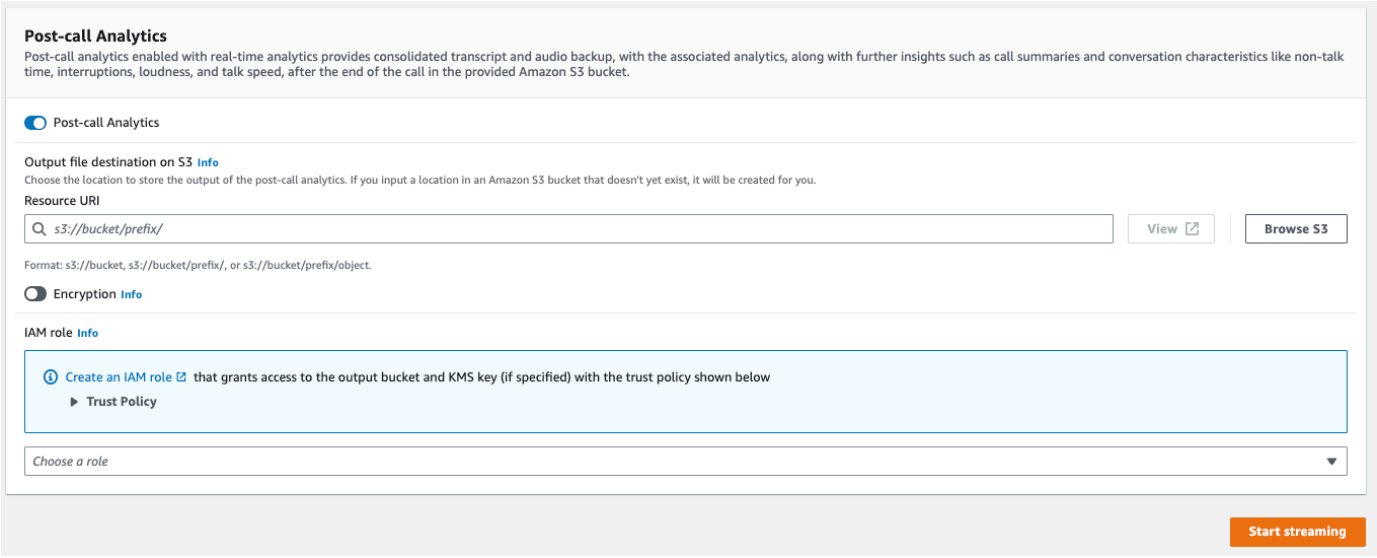


すべての選択を終えたら、[保存] を選択してメインページに戻ります。

5. 追加の分析を適用するには、「通話後分析」をオンに切り替えます。これにより、中断、ラウドネス、非通話時間、通話速度、通話時間、問題、アクションアイテム、結果など、通話後分析文字起こしと同じ分析が得られます。通話後分析出力は、リアルタイムコール分析トランスクリプトとは別のファイルに保存されます。



通話後分析を適用する場合は、 Amazon S3 出力ファイルの宛先と IAM ロールを指定する必要があります。オプションで出力を暗号化することもできます。



6. [ストリーミングの開始] を選択します。

HTTP/2 ストリーム

この例では、コール分析を有効にした状態で HTTP/2 リクエストを作成します。での HTTP/2 ストリーミングの使用の詳細については Amazon Transcribe、「」を参照してくださいHTTP/2 ストリームの設定。に固有のパラメータとヘッダーの詳細については、 Amazon Transcribe「」を参照してくださいStartCallAnalyticsStreamTranscription。

この例には通話後分析が含まれています。通話後分析が必要ない場合は、リクエストから PostCallAnalyticsSettings セクションを削除してます。

次の例に示す構成イベントは、ストリームの最初のイベントとして渡す必要があることに注意してください。

```
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"
            }
        }
    }
}
```

パラメータ定義は API リファレンス にあります。すべての API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。 AWS

WebSocket ストリーム

この例では、 WebSocket ストリームでコール分析を使用する署名付き URL を作成します。読みやすくするために、改行が追加されています。で WebSocket ストリームを使用する方法の詳細については、 Amazon Transcribe「」を参照してください WebSocket ストリームのセットアップ。パラメータの詳細については、「StartCallAnalyticsStreamTranscription」を参照してください。

この例には通話後分析が含まれています。通話後分析が必要ない場合は、リクエストから PostCallAnalyticsSettings セクションを削除してます。

次の例に示す構成イベントは、ストリームの最初のイベントとして渡す必要があることに注意してください。

```
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"
            }
        }
    }
}
```

パラメータ定義は API リファレンス にあります。すべての API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。 AWS

> **Tip**
>
> 上記の HTTP/2 と WebSocket の例には、通話後分析が含まれます。通話後分析が必要ない場合は、リクエストから PostCallAnalyticsSettings セクションを削除してます。PostCallAnalyticsSettings を有効にする場合は、最初のイベントとして構成イベントを送信する必要があります。構成イベントには、前述の例で示したように、ChannelDenifitions および PostStreamAnalyticsSettings の設定が含まれます。
> バイナリデータはでバイナリメッセージとして content-type application/octet-stream で渡され、構成イベントはでテキストメッセージとして content-type application/json で渡されます。
> 詳細については、「ストリーミング文字起こしの設定」を参照してください。

カテゴリマッチに関するリアルタイムアラートの作成

リアルタイムアラートを設定するには、まず REAL_TIME フラグが付いている TranscriptFilterType カテゴリを作成する必要があります。このフラグを使用すると、リアルタイムコール分析文字起こしにカテゴリを適用できます。

新しいカテゴリを作成する手順については、「リアルタイム文字起こしのカテゴリの作成」を参照してください。

リアルタイムコール分析文字起こしを開始すると、REAL_TIME フラグの付いたすべてのカテゴリが文字起こし出力にセグメントレベルで自動的に適用されます。TranscriptFilterType マッチがあると、その内容はトランスクリプトの CategoryEvent セクションに表示されます。その後、このパラメータとそのサブパラメータ、MatchedCategories および MatchedDetails を使用して、カスタムリアルタイムアラートを設定できます。

CategoryEvent マッチのリアルタイムコール分析文字起こしの出力例を次に示します。

```
"CategoryEvent": {
    "MatchedCategories": [ "shipping-complaint" ],
    "MatchedDetails": {
        "my package never arrived" : {
            "TimestampRanges": [
                {
                    "BeginOffsetMillis": 19010,
                    "EndOffsetMillis": 22690
```

```
                }
            ]
        }
    }
},
```

前述の例は、「荷物が届きません」という音声と完全に一致するテキストを表しています。これは「配送に関する苦情」カテゴリ内のルールを表しています。

リアルタイムアラートには、一覧表示されているパラメータを任意に組み合わせて含めるように設定できます。たとえば、一致したフレーズのみ (MatchedDetails) またはカテゴリ名 (MatchedCategories) のみを含むようにアラートを設定できます。または、すべてのパラメータを含むようにアラートを設定することもできます。

リアルタイムアラートをどのように設定するかは、組織のインターフェースと希望するアラートタイプによって異なります。たとえば、ポップアップ通知、電子メール、テキスト、またはシステムが受け付けることができるその他のアラートを送信するように CategoryEvent マッチを設定できます。

リアルタイムコール分析出力

リアルタイムコール分析のトランスクリプトは、セグメントごとに turn-by-turn フォーマットで表示されます。これらには、カテゴリイベント、問題検出、感情、PII の識別とリダクションが含まれます。カテゴリイベントでは、リアルタイムアラートを設定できます。詳細については、「カテゴリマッチに関するリアルタイムアラートの作成」を参照してください。

精度を高め、業界固有の用語など、ユースケースに合わせてトランスクリプトをさらにカスタマイズするには、コール分析リクエストにカスタム語彙またはカスタム言語モデルを使用します。冒涜的な言葉など、文字起こし結果に表示したくない言葉をマスキング、削除、またはタグ付けするには、語彙フィルタリングを追加します。

以下のセクションでは、リアルタイムコール分析文字起こしの JSON 出力の例を示します。

イベントカテゴリ

カテゴリの一致は、次のように文字起こし出力で表示されます。この例は、19010 ミリ秒のタイムスタンプから 22690 ミリ秒のタイムスタンプまでの音声が「ネットワークに関する苦情」カテゴリと一致していることを示しています。この場合、カスタムの「ネットワークに関する苦情」カテゴリでは、お客様が「ネットワークの問題」(単語が完全に一致) と言う必要がありました。

```
"CategoryEvent": {
```

```
    "MatchedCategories": [
        "network-complaint"
    ],
    "MatchedDetails": {
        "network issues" : {
            "TimestampRanges": [
                {
                    "BeginOffsetMillis": 9299375,
                    "EndOffsetMillis": 7899375
                }
            ]
        }
    }
},
```

問題検出

文字起こし出力では、問題検出マッチは次のように表示されます。この例では、文字 26 から文字 62 までのテキストが問題を説明してます。

```
"UtteranceEvent": {
    ...
    "Transcript": "Wang Xiulan I'm tired of the network issues my phone is having.",
    ...
    "IssuesDetected": [
        {
            "CharacterOffsets": {
                "BeginOffsetChar": 26,
                "EndOffsetChar": 62
            }
        }
    ]
},
```

感情

文字起こし出力では、感情分析は次のように表示されます。

```
"UtteranceEvent": {
    ...
    "Sentiment": "NEGATIVE",
    "Items": [{
```

```
        ...
```

PII 識別

文字起こし出力では、PII 識別は次のように表示されます。

```
"Entities": [
    {
        "Content": "Wang Xiulan",
        "Category": "PII",
        "Type": "NAME",
        "BeginOffsetMillis": 7999375,
        "EndOffsetMillis": 199375,
        "Confidence": 0.9989
    }
],
```

PII リダクション

文字起こし出力では、PII リダクションは次のように表示されます。

```
"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
        }
    ]
},
```

コンパイル済みのリアルタイムコール分析出力

簡潔にするために、次の文字起こし出力では一部の内容が省略記号に置き換えられています。

```
{
    "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
                        }
                    ]
                }
            }
        }
    }
}
```

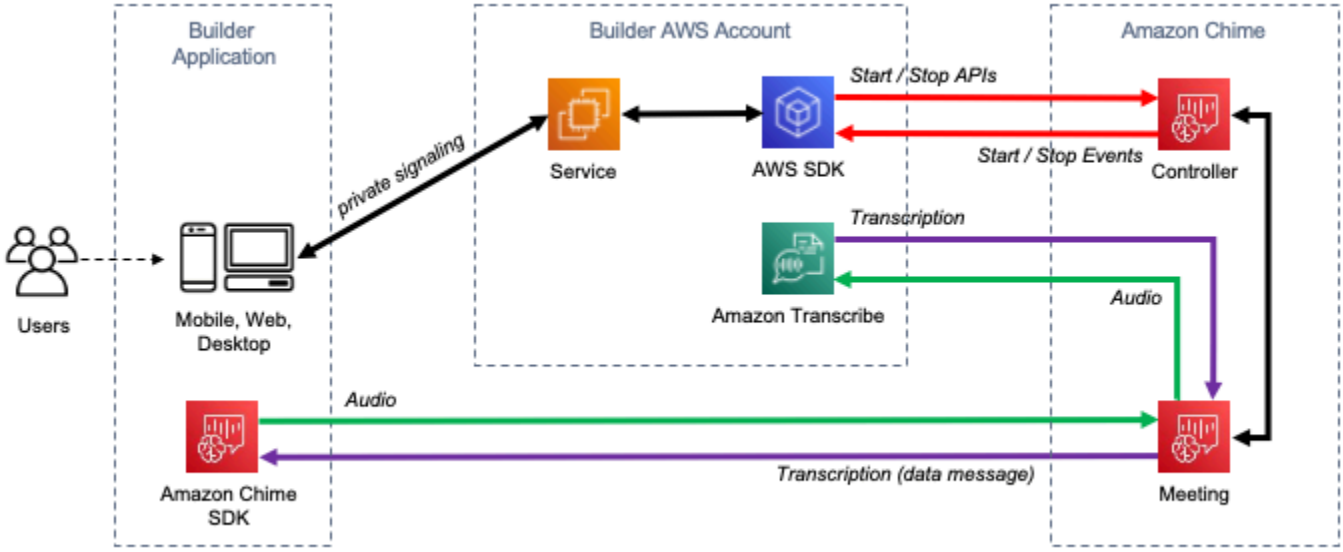
Amazon Chime通話内容をリアルタイムで文字起こし

Amazon TranscribeAmazon ChimeSDKと統合されているため、Amazon Chime通話のリアルタイムの文字起こしが容易になります。

Amazon ChimeSDK API を使用して文字起こしをリクエストすると、Amazon ChimeAmazon Transcribeへのオーディオのストリーミングが開始され、通話中もストリーミングが継続されます。

Amazon ChimeSDK は「アクティブトーカー」アルゴリズムを使用して上位 2 人のアクティブトーカーを選択し、そのオーディオを 1 つのストリームで 2Amazon Transcribe つの別々のチャンネルとして送信します。会議参加者は、Amazon Chime SDKデータメッセージを介してユーザー属性付きの文字起こしを受け取ります。Amazon ChimeSDK 開発者ガイドで配信例を確認できます。

Amazon Chimeトランスクリプションのデータフローを次の図に示します。



Amazon Chimeリアルタイム文字起こしを設定する方法の詳細と手順については、『SDK 開発者ガイド』の「Amazon ChimeSDK ライブ文字起こしの使用」を参照してください。Amazon ChimeAPI オペレーションについては、Amazon ChimeSDK API リファレンスを参照してください。

> **AWSMachine Learning ブログでさらに詳しく**
>
> リアルタイムの文字起こしによる精度の向上について詳しくは、以下をご覧ください。
>
> - Amazon ChimeSDK ミーティングでは、Amazon Transcribeおよびによるライブ文字起こしがサポートされるようになりましたAmazon Transcribe Medical

- Amazon Chime遠隔医療ソリューション用SDK

SDK を使用した Amazon Transcribe のコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で Amazon Transcribe を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、関連するシナリオやサービス間の例ではアクションのコンテキストが確認できます。

「シナリオ」は、同じサービス内で複数の関数を呼び出して、特定のタスクを実行する方法を示すコード例です。

クロスサービスの例は、複数の AWS のサービスで動作するサンプルアプリケーションです。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- SDK を使用した Amazon Transcribe のアクション AWS SDKs
  - AWS SDK または CLI CreateVocabularyで を使用する
  - AWS SDK または CLI DeleteMedicalTranscriptionJobで を使用する
  - AWS SDK または CLI DeleteTranscriptionJobで を使用する
  - AWS SDK または CLI DeleteVocabularyで を使用する
  - AWS SDK または CLI GetTranscriptionJobで を使用する
  - AWS SDK または CLI GetVocabularyで を使用する
  - AWS SDK または CLI ListMedicalTranscriptionJobsで を使用する
  - AWS SDK または CLI ListTranscriptionJobsで を使用する
  - AWS SDK または CLI ListVocabulariesで を使用する
  - AWS SDK または CLI StartMedicalTranscriptionJobで を使用する
  - AWS SDK または CLI StartStreamTranscriptionAsyncで を使用する
  - AWS SDK または CLI StartTranscriptionJobで を使用する
  - AWS SDK または CLI UpdateVocabularyで を使用する
- SDK を使用した Amazon Transcribe のシナリオ AWS SDKs

- AWS SDK を使用して Amazon Transcribe カスタム語彙を作成して絞り込む
- AWS SDK を使用して Amazon Transcribe で音声を書き起こし、ジョブデータを取得する
- AWS SDKs を使用した Amazon Transcribe のクロスサービスの例
  - Amazon Transcribe アプリを構築する
  - Amazon Transcribe ストリーミングアプリケーションを構築する
  - AWS SDK を使用してテキストを音声に変換し、テキストに戻す

SDK を使用した Amazon Transcribe のアクション AWS SDKs

次のコード例は、 AWS SDKs を使用して個々の Amazon Transcribe アクションを実行する方法を示しています。これらの抜粋は、Amazon Transcribe API を呼び出し、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。各例には GitHub、コードの設定と実行の手順を示す へのリンクが含まれています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細な一覧については、「Amazon Transcribe API リファレンス」を参照してください。

例

- AWS SDK または CLI CreateVocabularyで を使用する
- AWS SDK または CLI DeleteMedicalTranscriptionJobで を使用する
- AWS SDK または CLI DeleteTranscriptionJobで を使用する
- AWS SDK または CLI DeleteVocabularyで を使用する
- AWS SDK または CLI GetTranscriptionJobで を使用する
- AWS SDK または CLI GetVocabularyで を使用する
- AWS SDK または CLI ListMedicalTranscriptionJobsで を使用する
- AWS SDK または CLI ListTranscriptionJobsで を使用する
- AWS SDK または CLI ListVocabulariesで を使用する
- AWS SDK または CLI StartMedicalTranscriptionJobで を使用する
- AWS SDK または CLI StartStreamTranscriptionAsyncで を使用する
- AWS SDK または CLI StartTranscriptionJobで を使用する
- AWS SDK または CLI UpdateVocabularyで を使用する

AWS SDK または CLI **CreateVocabulary**で を使用する

以下のコード例は、CreateVocabulary の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- カスタム語彙を作成し改良する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <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;
    }
```

- API の詳細については、「 API リファレンスCreateVocabulary」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

カスタム語彙を作成するには

次の create-vocabulary 例は、カスタム語彙を作成します。カスタム語彙を作成するには、より正確に書き起こすべき用語のすべてを含むテキストファイルを作成しておく必要があります。には vocabulary-file-uri、そのテキストファイルの Amazon Simple Storage Service (Amazon S3) URI を指定します。language-code として、カスタム語彙の言語に対応する言語コードを指定します。vocabulary-name として、カスタムボキャブラリーに付ける名前を指定します。

```
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
```

出力:

```
{
    "VocabularyName": "cli-vocab-example",
    "LanguageCode": "language-code",
    "VocabularyState": "PENDING"
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「カスタムボキャブラリー」を参照してください。

- API の詳細については、「 コマンドリファレンスCreateVocabulary」の「」を参照してください。 AWS CLI

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 CreateVocabulary AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteMedicalTranscriptionJob**で を使用する

以下のコード例は、DeleteMedicalTranscriptionJob の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <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;
    }
```

- API の詳細については、「 API リファレンスDeleteMedicalTranscriptionJob」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

医療文字起こしジョブを削除するには

次の delete-medical-transcription-job の例は、医療文字起こしジョブを削除します。

```
aws transcribe delete-medical-transcription-job \
    --medical-transcription-job-name medical-transcription-job-name
```

このコマンドでは何も出力されません。

詳細については、Amazon Transcribe デベロッパーガイドDeleteMedicalTranscriptionJob」の「」を参照してください。

- API の詳細については、「 コマンドリファレンスDeleteMedicalTranscriptionJob」の「」を参照してください。 AWS CLI

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

クライアントを作成します。

```
import { TranscribeClient } from "@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 };
```

医療分野の文字起こしジョブを削除します。

```
// 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();
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスDeleteMedicalTranscriptionJob」の「」を参照してください。 AWS SDK for JavaScript

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteTranscriptionJob**で を使用する

以下のコード例は、DeleteTranscriptionJob の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- カスタム語彙を作成し改良する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <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;
    }
```

- API の詳細については、「 API リファレンスDeleteTranscriptionJob」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

文字起こしジョブの 1 つを削除するには

次の delete-transcription-job 例では、トランスクリプションジョブの 1 つを削除します。

```
aws transcribe delete-transcription-job \
    --transcription-job-name your-transcription-job
```

このコマンドでは何も出力されません。

詳細については、Amazon Transcribe デベロッパーガイドDeleteTranscriptionJob」の「」を参照してください。

- API の詳細については、「 コマンドリファレンスDeleteTranscriptionJob」の「」を参照してください。 AWS CLI

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

文字起こしジョブを削除します。

```
// 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();
```

クライアントを作成します。

```
import { TranscribeClient } from "@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 };
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスDeleteTranscriptionJob」の「」を参照してください。 AWS SDK for JavaScript

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 DeleteTranscriptionJob AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteVocabulary**で を使用する

以下のコード例は、DeleteVocabulary の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- カスタム語彙を作成し改良する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/// <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;
}
```

- API の詳細については、「 API リファレンスDeleteVocabulary」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

カスタム語彙を削除するには

次の delete-vocabulary の例は、カスタム語彙を削除します。

```
aws transcribe delete-vocabulary \
    --vocabulary-name vocabulary-name
```

このコマンドでは何も出力されません。

詳細については、「Amazon Transcribe デベロッパーガイド」の「カスタムボキャブラリー」を参照してください。

- API の詳細については、「 コマンドリファレンスDeleteVocabulary」の「」を参照してください。 AWS CLI

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、DeleteVocabularyAWS 「 SDK for Python (Boto3) API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetTranscriptionJob**で を使用する

以下のコード例は、GetTranscriptionJob の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- カスタム語彙を作成し改良する
- 音声の文字起こしとジョブデータを取得する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/// <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;
    }
```

- API の詳細については、「 API リファレンスGetTranscriptionJob」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

特定の文字起こしジョブに関する情報を取得するには

次の get-transcription-job 例では、特定の文字起こしジョブに関する情報を取得します。文字起こし結果にアクセスするには、 TranscriptFileUri パラメータを使用します。MediaFileUri パラメータを使用して、このジョブで文字起こししたオーディオファイルを確認します。Settings オブジェクトを使用して、文字起こしジョブで有効にしたオプション機能を確認できます。

```
aws transcribe get-transcription-job \
    --transcription-job-name your-transcription-job
```

出力:

```
{
    "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
    }
}
```

詳細については、Amazon Transcribe デベロッパーガイド」の「開始方法 (AWS コマンドラインインターフェイス) 」を参照してください。

- API の詳細については、「 コマンドリファレンスGetTranscriptionJob」の「」を参照してください。 AWS CLI

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 GetTranscriptionJob AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetVocabulary**で を使用する

以下のコード例は、GetVocabulary の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- カスタム語彙を作成し改良する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <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;
    }
```

- API の詳細については、「 API リファレンスGetVocabulary」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

カスタム語彙に関する情報を取得するには

次の get-vocabulary 例では、以前に作成したカスタム語彙に関する情報を取得します。

```
aws transcribe get-vocabulary \
    --vocabulary-name cli-vocab-1
```

出力:

```
{
    "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"
```

```
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「カスタムボキャブラリー」を参照してください。

- API の詳細については、「 コマンドリファレンスGetVocabulary」の「」を参照してください。 AWS CLI

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 GetVocabulary AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListMedicalTranscriptionJobs**で を使用する

以下のコード例は、ListMedicalTranscriptionJobs の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /// <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;
   }
```

- API の詳細については、「 API リファレンスListMedicalTranscriptionJobs」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

医療文字起こしジョブを一覧表示するには

次のlist-medical-transcription-jobs例では、 AWS アカウントとリージョンに関連付けられた医療文字起こしジョブを一覧表示します。特定の文字起こしジョブに関する詳細情報を取得するには、文字起こし出力で MedicalTranscriptionJobName パラメータの値をコピーし、その値を get-medical-transcription-job コマンドの MedicalTranscriptionJobNameオプションに指定します。文字起こしジョブをさらに表示するには、 NextToken パラメータの値をコピーし、 list-medical-transcription-jobs コマンドを再度実行して、 --next-tokenオプションでその値を指定します。

```
aws transcribe list-medical-transcription-jobs
```

出力:

```
{
    "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"
        }
    ]
}
```

詳細については、Amazon Transcribe デベロッパーガイド」のhttps://docs.aws.amazon.com/transcribe/latest/dg/batch-med-transcription.html>」を参照してください。

- API の詳細については、「 コマンドリファレンスListMedicalTranscriptionJobs」の「」を参照してください。 AWS CLI

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

クライアントを作成します。

```
import { TranscribeClient } from "@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 };
```

医療分野の文字起こしジョブを一覧表示します。

```
// 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();
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。

- API の詳細については、「 API リファレンスListMedicalTranscriptionJobs」の「」を参照してください。 AWS SDK for JavaScript

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListTranscriptionJobs**で を使用する

以下のコード例は、ListTranscriptionJobs の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <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;
    }
```

- API の詳細については、「 API リファレンスListTranscriptionJobs」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

文字起こしジョブを一覧表示するには

次のlist-transcription-jobs例では、 AWS アカウントとリージョンに関連付けられた文字起こしジョブを一覧表示します。

```
aws transcribe list-transcription-jobs
```

出力:

```
{
    "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"
        }
    ]
}
```

詳細については、Amazon Transcribe デベロッパーガイド」の「開始方法 (AWS コマンドラインインターフェイス) 」を参照してください。

- API の詳細については、「 コマンドリファレンスListTranscriptionJobs」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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("--------------");
               });
```

```
        }
    }
```

- API の詳細については、「 API リファレンスListTranscriptionJobs」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

文字起こしジョブを一覧表示します。

```
// 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();
```

クライアントを作成します。

```
import { TranscribeClient } from "@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 };
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスListTranscriptionJobs」の「」を参照してください。 AWS SDK for JavaScript

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 ListTranscriptionJobs AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListVocabularies**で を使用する

以下のコード例は、ListVocabularies の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- カスタム語彙を作成し改良する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <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;
    }
```

- API の詳細については、「 API リファレンスListVocabularies」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

カスタム語彙を一覧表示するには

次のlist-vocabularies例では、 AWS アカウントとリージョンに関連付けられているカスタム語彙を一覧表示します。

```
aws transcribe list-vocabularies
```

出力:

```
{
    "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"
        }
    ]
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「カスタムボキャブラリー」を参照してください。

- API の詳細については、「 コマンドリファレンスListVocabularies」の「」を参照してください。 AWS CLI

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、ListVocabulariesAWS 「 SDK for Python (Boto3) API リファレンス」の「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **StartMedicalTranscriptionJob**で を使用する

以下のコード例は、StartMedicalTranscriptionJob の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/// <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;
    }
```

- API の詳細については、「 API リファレンスStartMedicalTranscriptionJob」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

例 1: オーディオファイルとして保存されている医療ディクテーションを文字起こしするには

次の start-medical-transcription-job の例は、オーディオファイルの文字起こしを行います。トランスクリプション出力の場所を OutputBucketName パラメータで指定します。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfile.json
```

myfile.json の内容:

```
{
    "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"
    }
}
```

出力:

```
{
    "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"
    }
}
```

詳細については、「Amazon Transcribe 開発者ガイド」の「バッチトランスクリプションの概要」を参照してください。

例 2: オーディオファイルとして保存されている臨床医と患者の対話を文字起こしするには

次の start-medical-transcription-job 例では、臨床医と患者の対話を含むオーディオファイルの文字起こしを行います。 OutputBucketName パラメータで文字起こし出力の場所を指定します。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysecondfile.json
```

mysecondfile.json の内容:

```
{
    "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"
    }
}
```

出力:

```
{
    "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"
    }
}
```

詳細については、「Amazon Transcribe 開発者ガイド」の「バッチトランスクリプションの概要」を参照してください。

例 3: 臨床医と患者の対話のマルチチャネルオーディオファイルを書き起こすには

次の start-medical-transcription-job 例では、オーディオオファイルの各チャネルの音声の文字起こしを行い、チャネル別の文字起こし結果を組み合わせて、単一の文字起こし出力にまとめます。文字起こしの出力の場所を OutputBucketName パラメータで指定します。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mythirdfile.json
```

mythirdfile.json の内容:

```
{
    "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
        }
}
```

出力:

```
{
```

```
    "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"
    }
}
```

詳細については、「Amazon Transcribe 開発者ガイド」の「チャネル識別」を参照してください。

例 4: 臨床医と患者の対話のオーディオファイルを文字起こしして、文字起こし出力の話者を特定するには

次の start-medical-transcription-job の例は、オーディオファイルを書き起こしして、文字起こし出力の各話者の発話にラベルを付けます。文字起こしの出力の場所を OutputBucketName パラメータで指定します。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfourthfile.json
```

myfourthfile.json の内容:

```
{
    "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
        }
}
```

出力:

```
{
    "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"
    }
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「話者の識別」を参照してください。

例 5: オーディオファイルとして保存されている医療会話を、最大 2 つの代替文字起こし結果に文字起こしするには

次の `start-medical-transcription-job` の例は、単一のオーディオファイルから最大 2 つの代替文字起こし結果を作成します。文字起こし結果ごとに信頼度レベルが関連付けられます。デフォルトでは、Amazon Transcribe は、信頼度レベルが最も高い文字起こし結果を返します。Amazon Transcribe で他の信頼度レベルがより低いトランスクリプションを返すようにも指定できます。文字起こしの出力の場所を `OutputBucketName` パラメータで指定します。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfifthfile.json
```

myfifthfile.json の内容:

```
{
    "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
    }
}
```

出力:

```
{
    "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"
    }
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「代替文字起こし」を参照してください。

例 6: 医療ディクテーションのオーディオファイルを、最大 2 つの代替文字起こし結果に文字起こしするには

次の start-medical-transcription-job の例は、オーディオファイルを文字起こしして、語彙フィルターを使用して不要な単語をマスクします。 OutputBucketName パラメータで文字起こし出力の場所を指定します。

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysixthfile.json
```

mysixthfile.json の内容:

```
{
    "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
    }
}
```

出力:

```
{
    "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"
    }
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「代替文字起こし」を参照してください。

例 7: カスタムボ語彙を使用して、医療ディクテーションのオーディオファイルをより正確に書き起こすには

次の start-medical-transcription-job の例は、オーディオファイルを文字起こしして、以前に作成した医療カスタム語彙を使用して文字起こし結果の精度を高めます。文字起こしの出力の場所を OutputBucketName パラメータで指定します。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myseventhfile.json
```

mysixthfile.json の内容:

```
{
    "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"
    }
}
```

出力:

```
{
    "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"
    }
}
```

詳細については、「Amazon Transcribe 開発者ガイド」の「医療カスタムボキャブラリー」を参照してください。

- API の詳細については、「 コマンドリファレンスStartMedicalTranscriptionJob」の「」を参照してください。 AWS CLI

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

クライアントを作成します。

```
import { TranscribeClient } from "@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 };
```

医療分野の文字起こしジョブを開始します。

```
// 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();
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。

- API の詳細については、「 API リファレンスStartMedicalTranscriptionJob」の「」を参照してください。 AWS SDK for JavaScript

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **StartStreamTranscriptionAsync**で を使用する

次の例は、StartStreamTranscriptionAsync を使用する方法を説明しています。

C++

SDK for C++

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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;
}
```

- API の詳細については、「 API リファレンスStartStreamTranscriptionAsync」の「」を参照してください。 AWS SDK for C++

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI StartTranscriptionJobで を使用する

以下のコード例は、StartTranscriptionJob の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- カスタム語彙を作成し改良する
- 音声の文字起こしとジョブデータを取得する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <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;
    }
```

- API の詳細については、「 API リファレンスStartTranscriptionJob」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

例 1: オーディオファイルを文字起こしするには

次の `start-transcription-job` の例は、音声ファイルの文字起こしを行います。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfile.json
```

`myfile.json` の内容:

```
{
    "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"
    }
```

```
}
```

詳細については、Amazon Transcribe デベロッパーガイド」の「開始方法 (AWS コマンドラインインターフェイス) 」を参照してください。

例 2: マルチチャネルのオーディオファイルを文字起こしするには

次の start-transcription-job の例は、マルチチャネルのオーディオファイルの文字起こしを行います。

```
aws transcribe start-transcription-job \
    --cli-input-json file://mysecondfile.json
```

mysecondfile.json の内容:

```
{
    "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
    }
}
```

出力:

```
{
    "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
        }
    }
}
```

詳細については、「Amazon Transcribe 開発者ガイド」の「マルチチャネル音声の書き起こし」を参照してください。

例 3: オーディオファイルを文字起こしして、複数の異なる話者を識別するには

次の start-transcription-job 例では、オーディオファイルを書き起こし、文字起こし出力の話者を識別します。

```
aws transcribe start-transcription-job \
    --cli-input-json file://mythirdfile.json
```

mythirdfile.json の内容:

```
{
    "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
    }
}
```

出力:

```
{
    "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
        }
    }
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「話者の識別」を参照してください。

例 4: オーディオファイルを文字起こしして、文字起こし出力内の不要な単語をすべてマスクするには

次の start-transcription-job 例では、オーディオファイルを書き起こし、以前に作成した語彙フィルターを使用して不要な単語をマスクします。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfourthfile.json
```

myfourthfile.json の内容:

```
{
    "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"
    }
}
```

出力:

```
{
    "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"
        }
    }
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「トランスクリプションのフィルタリング」を参照してください。

例 5: オーディオファイルを文字起こしし、文字起こし出力から不要な単語を削除するには

次の start-transcription-job 例では、オーディオファイルを書き起こし、以前に作成した語彙フィルターを使用して不要な単語をマスクします。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfifthfile.json
```

myfifthfile.json の内容:

```
{
    "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"
    }
}
```

出力:

```
{
    "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"
        }
    }
}
```

詳細については、「Amazon Transcribe 開発者ガイド」の「トランスクリプションのフィルタリング」を参照してください。

例 6: カスタム語彙を使用して、オーディオファイルをより正確に文字起こしするには

次の `start-transcription-job` 例では、オーディオファイルを書き起こし、以前に作成した語彙フィルターを使用して不要な単語をマスクします。

```
aws transcribe start-transcription-job \
    --cli-input-json file://mysixthfile.json
```

`mysixthfile.json` の内容:

```
{
    "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"
    }
}
```

出力:

```
{
    "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"
        }
    }
}
```

詳細については、「Amazon Transcribe 開発者ガイド」の「トランスクリプションのフィルタリング」を参照してください。

例 7: オーディオファイルの言語を識別して文字起こしするには

次の start-transcription-job 例では、オーディオファイルを書き起こし、以前に作成した語彙フィルターを使用して不要な単語をマスクします。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myseventhfile.json
```

myseventhfile.json の内容:

```
{
    "TranscriptionJobName": "cli-identify-language-transcription-job",
    "IdentifyLanguage": true,
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
}
```

出力:

```
{
    "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
    }
}
```

詳細については、「Amazon Transcribe 開発者ガイド」の「言語の特定」を参照してください。

例 8: 個人を特定できる情報をマスクしてオーディオファイルを文字起こしするには

次の start-transcription-job の例は、オーディオファイルを文字起こしして、文字起こし出力内の個人を特定できる情報をマスクします。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myeighthfile.json
```

myeigthfile.json の内容:

```
{
    "TranscriptionJobName": "cli-redaction-job",
    "LanguageCode": "language-code",
    "Media": {
        "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
    },
    "ContentRedaction": {
        "RedactionOutput":"redacted",
        "RedactionType":"PII"
    }
}
```

出力:

```
{
```

```
    "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"
        }
    }
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「コンテンツの自動マスキング」を参照してください。

例 9: 個人を特定できる情報 (PII) をマスクしたトランスクリプトとマスクしていないトランスクリプトを生成するには

次の start-transcription-job の例は、オーディオファイルの 2 つの文字起こしを生成します。1 つでは個人を特定できる情報をマスクし、別の 1 つではマスクしません。

```
aws transcribe start-transcription-job \
    --cli-input-json file://myninthfile.json
```

myninthfile.json の内容:

```
{
    "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"
    }
}
```

出力:

```
{
    "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"
        }
    }
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「自動コンテンツリダクション」を参照してください。

例 10: 以前に作成したカスタム言語モデルを使用してオーディオファイルを文字起こしするには

次の start-transcription-job の例は、以前に作成したカスタム言語モデルを使用してオーディオファイルを文字起こしします。

```
aws transcribe start-transcription-job \
    --cli-input-json file://mytenthfile.json
```

mytenthfile.json の内容:

```
{
    "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"
```

```
    }
}
```

出力:

```
{
    "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"
        }
    }
}
```

詳細については、「Amazon Transcribe 開発者ガイド」の「カスタム言語モデルを使用したドメイン固有のトランスクリプション精度の向上」を参照してください。

- API の詳細については、「 コマンドリファレンスStartTranscriptionJob」の「」を参照してください。 AWS CLI

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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();
        }
    }
}
```

- API の詳細については、「 API リファレンスStartTranscriptionJob」の「」を参照してください。 AWS SDK for Java 2.x

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

文字起こしジョブを開始します。

```
// 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();
```

クライアントを作成します。

```
import { TranscribeClient } from "@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 };
```

- 詳細については、「AWS SDK for JavaScript デベロッパーガイド」を参照してください。
- API の詳細については、「 API リファレンスStartTranscriptionJob」の「」を参照してください。 AWS SDK for JavaScript

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 StartTranscriptionJob AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **UpdateVocabulary**で を使用する

以下のコード例は、UpdateVocabulary の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- カスタム語彙を作成し改良する

.NET

AWS SDK for .NET

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <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;
    }
```

- API の詳細については、「 API リファレンスUpdateVocabulary」の「」を参照してください。 AWS SDK for .NET

CLI

AWS CLI

カスタム語彙を新しい用語で更新するには

次の update-vocabulary の例は、カスタム語彙の作成に使用した用語を、指定した新しい用語で上書きします。前提条件: カスタム語彙の用語を置き換えるには、新しい用語を含むファイルが必要です。

```
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
```

出力:

```
{
    "VocabularyName": "custom-vocabulary",
    "LanguageCode": "language",
    "VocabularyState": "PENDING"
```

```
}
```

詳細については、「Amazon Transcribe デベロッパーガイド」の「カスタムボキャブラリー」を参照してください。

- API の詳細については、「 コマンドリファレンスUpdateVocabulary」の「」を参照してください。 AWS CLI

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

- API の詳細については、 UpdateVocabulary AWS SDK for Python (Boto3) API リファレンスの「」を参照してください。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した Amazon Transcribe のシナリオ AWS SDKs

次のコード例は、 AWS SDKs を使用して Amazon Transcribe で一般的なシナリオを実装する方法を示しています。これらのシナリオは、Amazon Transcribe 内で複数の機能を呼び出すことによって特定のタスクを実行する方法を示しています。各シナリオには GitHub、コードのセットアップと実行の手順を示す へのリンクが含まれています。

例

- AWS SDK を使用して Amazon Transcribe カスタム語彙を作成して絞り込む
- AWS SDK を使用して Amazon Transcribe で音声を書き起こし、ジョブデータを取得する

AWS SDK を使用して Amazon Transcribe カスタム語彙を作成して絞り込む

次のコードサンプルは、以下の操作方法を示しています。

- Amazon S3 に音声ファイルをアップロードします。
- Amazon Transcribe ジョブを実行してファイルを文字起こしし、結果を取得します。
- カスタム語彙を作成して改良し、文字起こしの精度を向上させます。
- カスタム語彙を使ってジョブを実行し、結果を取得します。

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ルイス キャロルによる「ジャバウォッキー」の朗読を収録した音声ファイルを文字起こしします。まず、Amazon Transcribe アクションをラップする関数を作成します。

```
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
```

ラッパー関数を呼び出して、カスタム語彙なしで音声を文字起こししてから、カスタム語彙の異なるバージョンで文字起こしを行うと、よりよい結果が取得できます。

```
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!")
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - CreateVocabulary
  - DeleteTranscriptionJob
  - DeleteVocabulary
  - GetTranscriptionJob
  - GetVocabulary
  - ListVocabularies
  - StartTranscriptionJob
  - UpdateVocabulary

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon Transcribe で音声を書き起こし、ジョブデータを取得する

次のコード例は、以下を実行する方法を示しています。

- Amazon Transcribe で文字起こしジョブを開始します。
- ジョブが完了するまで待ちます。

- 書き起こしが保存されている URI を取得します。

詳細については、「Amazon Transcribe の開始方法」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

PCM ファイルを書き起こします。

```
/**
 * 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();
        }
    }
}
```

コンピュータのマイクからのストリーミングオーディオを文字起こしします。

```
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();
```

```
        }
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - GetTranscriptionJob
  - StartTranscriptionJob

Python

SDK for Python (Boto3)

> **Note**
>
> については、「」を参照してください GitHub。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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()
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - GetTranscriptionJob
  - StartTranscriptionJob

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs を使用した Amazon Transcribe のクロスサービスの例

次のサンプルアプリケーションでは、 AWS SDKsを使用して Amazon Transcribe を他の と組み合わせます AWS のサービス。各例には GitHub、アプリケーションのセットアップと実行の手順を示すへのリンクが含まれています。

例

- Amazon Transcribe アプリを構築する

- Amazon Transcribe ストリーミングアプリケーションを構築する
- AWS SDK を使用してテキストを音声に変換し、テキストに戻す

Amazon Transcribe アプリを構築する

次のコード例は、Amazon Transcribe を使用して音声録音を文字起こしし、ブラウザに表示する方法を示しています。

JavaScript

SDK for JavaScript (v3)

Amazon Transcribe を使用して音声録音を文字起こしし、ブラウザに表示するアプリを作成します。このアプリは 2 つの Amazon Simple Storage Service (Amazon S3) バケットを使用します。1 つはアプリケーションコードをホストし、もう 1 つは文字起こしを保存します。このアプリは、Amazon Cognito ユーザープールを使用してユーザーを認証します。認証されたユーザーには、必要な AWS サービスにアクセスするための AWS Identity and Access Management (IAM) アクセス許可があります。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例は、AWS SDK for JavaScript v3 デベロッパーガイドでも使用できます。

この例で使用されているサービス

- Amazon Cognito ID
- Amazon S3
- Amazon Transcribe

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Transcribe ストリーミングアプリケーションを構築する

次のコード例は、ライブ音声をリアルタイムで記録、転写、翻訳し、結果を E メールで送信するアプリケーションを構築する方法を示しています。

JavaScript

SDK for JavaScript (v3)

Amazon Transcribe を使用して、ライブ音声をリアルタイムで記録、転写、翻訳し、Amazon Simple Email Service (Amazon SES) を使用して結果を E メールで送信するアプリケーションを構築する方法について説明します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用してテキストを音声に変換し、テキストに戻す

次のコードサンプルは、以下の操作方法を示しています。

- Amazon Polly を使用して、プレーンテキスト (UTF-8) 入力ファイルを音声ファイルに合成します。
- Amazon S3 バケットに音声ファイルをアップロードします。
- Amazon Transcribe を使用して、音声ファイルをテキストに変換します。
- テキストを表示します。

Rust

SDK for Rust

Amazon Polly を使用して、プレーンテキスト (UTF-8) 入力ファイルを音声ファイルに合成し、音声ファイルを Amazon S3 バケットにアップロードし、Amazon Transcribe を使用して音声ファイルをテキストに変換し、テキストを表示します。

完全なソースコードとセットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Polly
- Amazon S3
- Amazon Transcribe

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でこのサービスを使用する。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Transcribe でのセキュリティ

AWS では、クラウドのセキュリティが最優先事項です。AWS のお客様は、セキュリティを最も重視する組織の要件を満たすよう構築されたデータセンターとネットワークアーキテクチャを利用できます。

セキュリティは、AWS とユーザーの間の責任共有です。責任共有モデル では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

- クラウドのセキュリティ:AWSは、クラウドのセキュリティ:は、AWSクラウドのセキュリティ:AWS クラウドを保護する責任を担います。 AWSまた、安全に使用できるサービスも用意されています。AWS コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。Amazon Transcribe に適用するコンプライアンスプログラムの詳細については、コンプライアンスプログラムによる対象範囲内の AWS のサービスをご参照ください。
- クラウド内のセキュリティ:AWSお客様の責任は使用するのサービスによって決まります。また、お客様は、データの機密性、お客様の会社の要件、および適用される法律および規制など、その他の要因についても責任を負います。

このドキュメントは、Amazon Transcribe を使用する際に責任共有モデルを適用する方法を理解するのに役立ちます。次のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために Amazon Transcribe を設定する方法を示します。また、AWSAmazon Transcribeリソースの安全性を確保する方法も説明します。

トピック

- Amazon Transcribe 向けの Identity and Access Management
- Amazon Transcribe でのデータ保護
- モニタリング Amazon Transcribe
- のコンプライアンス検証 Amazon Transcribe
- Amazon Transcribe での耐障害性
- Amazon Transcribe でのインフラストラクチャセキュリティ
- Amazon Transcribe の脆弱性分析と管理
- Amazon Transcribe のセキュリティのベストプラクティス

Amazon Transcribe 向けの Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービスです。IAM 管理者は、誰をわ認証 (サインイン) し、誰に Amazon Transcribe リソースの使用を許可する (権限を持たせる) かを制御します。IAM は、無料で使用できる AWS のサービスです。

トピック

対象者

AWS Identity and Access Management (IAM) の用途は、Amazon Transcribe で行う作業によって異なります。

サービスユーザー - Amazon Transcribe サービスを使用してジョブを実行する場合は、必要な権限と認証情報を管理者が用意します。作業を実行するためにさらに多くの Amazon Transcribe 機能を使用するとき、追加の権限が必要になる場合があります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。Amazon Transcribe 機能にアクセスできない場合は、「Amazon Transcribe アイデンティティとアクセスに関するトラブルシューティング」を参照してください。

サービス管理者 - 社内の Amazon Transcribe リソースを担当している場合は、通常、Amazon Transcribe への完全なアクセスがあります。サービスのユーザーがどの Amazon Transcribe 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を確認して、IAM の基本概念を理解してください。お客様の会社で Amazon Transcribe で IAM を利用する方法の詳細については、「Amazon Transcribe と IAM の連携方法」を参照してください。

IAM 管理者 - 管理者は、Amazon Transcribe へのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる Amazon Transcribe アイデンティティベース

のポリシーの例を表示するには、「Amazon Transcribe アイデンティティベースポリシーの例」を参照してください。

アイデンティティを使用した認証

認証とは、アイデンティティ認証情報を使用して AWSにサインインする方法です。ユーザーは、AWS アカウントのルートユーザーもしくは IAM ユーザーとして、または IAM ロールを引き受けることによって、認証を受ける (AWS にサインインする) 必要があります。

ID ソースから提供された認証情報を使用して、フェデレーティッドアイデンティティとして AWS にサインインできます。 AWS IAM Identity Centerフェデレーティッドアイデンティティの例としては、(IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報などがあります。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWSにアクセスする場合、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。AWS へのサインインの詳細については、『AWS サインイン ユーザーガイド』の「AWS アカウント にサインインする方法」を参照してください。

プログラムで AWS にアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報でリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、『IAM ユーザーガイド』の「AWS API リクエストの署名」を参照してください。

使用する認証方法を問わず、追加のセキュリティ情報の提供が求められる場合もあります。例えば、AWS では、アカウントのセキュリティ強化のために多要素認証 (MFA) の使用をお勧めしています。詳細については、『AWS IAM Identity Center ユーザーガイド』の「Multi-factor authentication (多要素認証)」 および『IAM ユーザーガイド』の「AWS での多要素認証 (MFA) の使用」を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを作成する場合は、そのアカウントのすべての AWS のサービスとリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。このアイデンティティは AWS アカウントのルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要が

あるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッド ID

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに対し、ID プロバイダーとのフェデレーションを使用して、一時的な認証情報の使用により、AWS のサービスにアクセスすることを要求します。

フェデレーテッドアイデンティティ は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリのユーザーか、または ID ソースから提供された認証情報を使用して AWS のサービス にアクセスするユーザーです。フェデレーテッドアイデンティティ が AWS アカウント にアクセスすると、ロールが継承され、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Center を使用することをお勧めします。IAM アイデンティティセンター でユーザーとグループを作成するか、すべての AWS アカウント とアプリケーションで使用するために、独自の ID ソースで一連のユーザーとグループに接続して同期することもできます。IAM アイデンティティセンター の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM アイデンティティセンター?」(IAM アイデンティティセンター とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたは 1 つのアプリケーションに対して特定の権限を持つ AWS アカウント 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユー

ザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、『IAM ユーザーガイド』の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定の権限を持つ、AWS アカウント 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロールを切り替えることによって、AWS Management Console で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、『IAM ユーザーガイド』の「IAM ロールの使用」を参照してください。

一時的な認証情報を持った IAM ロールは、以下の状況で役立ちます。

- フェデレーションユーザーユーザーアクセス – フェデレーションアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー向けロールの作成」を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。権限セットの詳細については、『AWS IAM Identity Center ユーザーガイド』の「権限セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。
- クロスサービスアクセス - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスで

は、呼び出し元プリンシパルの権限、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。

- Forward access sessions (FAS) – IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、AWS のサービス を呼び出すプリンシパル権限と、リクエスト AWS のサービス を組み合わせ、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するための権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、『IAM ユーザーガイド』の「AWS のサービス に権限を委任するロールの作成」を参照してください。
- サービスリンクロール - サービスリンクロールは、AWS のサービス にリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスリンクロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールの権限を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用してアクセス許可を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、『IAM ユーザーガイド』の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセス権の管理

AWS でアクセス権を管理するには、ポリシーを作成して AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付け

て、これらの権限を定義します。AWS は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWSに保存されます。JSON ポリシードキュメントの構造と内容の詳細については、『IAM ユーザーガイド』の「JSON ポリシー概要」を参照してください。

管理者は AWSJSON ポリシーを使用して、だれが何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWS API からロール情報を取得できます。

アイデンティティベースポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。管理ポリシーは、AWS アカウント 内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーとカスタマー管理ポリシーがあります。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、『IAM ユーザーガイド』の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーの例には、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定

されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは IAM の AWSマネージドポリシーは使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかをコントロールします。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Simple Storage Service (Amazon S3)、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS では、他の一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- 権限の境界 - 権限の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティに権限の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとその権限の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、権限の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。権限の境界の詳細については、『IAM ユーザーガイド』の「IAM エンティティの権限の境界」を参照してください。
- サービスコントロールポリシー (SCP) - SCP は、AWS Organizations で組織や組織単位 (OU) の最大権限を指定する JSON ポリシーです。AWS Organizations は、顧客のビジネスが所有する複数の AWS アカウント をグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対する権限を制限します (各 AWS アカウントのルートユーザー など)。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果として

セッションの権限の範囲は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」をご参照ください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、「IAM ユーザーガイド」の「ポリシーの評価論理」を参照してください。

Amazon Transcribe と IAM の連携方法

IAM を使用して Amazon Transcribe へのアクセスを管理する前に、Amazon Transcribe で利用できる IAM の機能について学びます。

Amazon Transcribe で使用できる IAM 機能

IAM 機能	Amazon Transcribe サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	いいえ
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	なし
ABAC (ポリシー内のタグ)	部分的
一時的な認証情報	あり
プリンシパル権限	あり
サービスロール	あり

IAM 機能	Amazon Transcribe サポート
サービスリンクロール	いいえ

Amazon Transcribe およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、「IAM ユーザーガイド」の「IAM と連携する AWS のサービス」を参照してください。

Amazon Transcribe のアイデンティティベースのポリシー

アイデンティティベースポリシーをサポートする	あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それがアタッチされているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素について学ぶには、IAM ユーザーガイドのIAM JSON ポリシーの要素のリファレンスを参照してください。

Amazon Transcribe のアイデンティティベースのポリシーの例

Amazon Transcribe アイデンティティベースのポリシーの例を表示するには、「Amazon Transcribe アイデンティティベースポリシーの例」を参照してください。

Amazon Transcribe 内のリソースベースのポリシー

リソースベースのポリシーのサポート	なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーの例には、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがありま

す。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービス を含めることができます。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる AWS アカウント にある場合、信頼できるアカウントの IAM 管理者は、リソースにアクセスするための権限をプリンシパルエンティティ (ユーザーまたはロール) に付与する必要もあります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーを追加する必要はありません。詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

Amazon Transcribe のポリシーアクション

ポリシーアクションに対するサポート	あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連する AWS API オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

Amazon Transcribe アクションのリストを確認するには、サービス認可リファレンスの「Amazon Transcribe で定義されるアクション」を参照してください。

Amazon Transcribe のポリシーアクションは、アクションの前に transcribe プレフィックスを使用します。単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "transcribe:action1",
      "transcribe:action2"
]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、List という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "transcribe:List*"
```

Amazon Transcribe アイデンティティベースのポリシーの例を表示するには、「Amazon Transcribe アイデンティティベースポリシーの例」を参照してください。

Amazon Transcribe のポリシーリソース

ポリシーリソースに対するサポート	あり

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの Resource 要素は、アクションが適用される 1 つ以上のオブジェクトを指定します。ステートメントには、Resource または NotResource 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon Transcribe リソースのタイプとその ARN のリストを確認するには、「サービス認可リファレンス」の「Amazon Transcribe で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon Transcribe で定義されるアクション」を参照してください。

Amazon Transcribe アイデンティティベースのポリシーの例を表示するには、「Amazon Transcribe アイデンティティベースポリシーの例」を参照してください。

Amazon Transcribe 向けのポリシー条件キー

サービス固有のポリシー条件キーのサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

`Condition` 要素 (または `Condition` ブロック) を使用すると、ステートメントが有効になる条件を指定できます。`Condition` 要素はオプションです。equal や less than などの条件演算子を使用して条件式を作成することによって、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の `Condition` 要素を指定するか、1 つの `Condition` 要素に複数のキーを指定すると、AWS は `AND` 論理演算子を使用してそれらを評価します。単一の条件キーに複数の値を指定すると、AWS は `OR` 論理演算子を使用して条件を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシー要素: 変数およびタグ」を参照してください。

AWS はグローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」の「AWS グローバル条件コンテキストキー」を参照してください。

Amazon Transcribe の条件キーのリストを確認するには、サービス認可リファレンスの「Amazon Transcribe の条件キー」を参照してください。どのアクションおよびリソースと条件キーを使用できるかについては、「Amazon Transcribe で定義されるアクション」を参照してください。

Amazon Transcribe アイデンティティベースのポリシーの例を表示するには、Amazon Transcribe アイデンティティベースポリシーの例を参照してください。

Amazon Transcribe の ACL

ACL のサポート	なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon Transcribe による ABAC

ABAC (ポリシー内のタグ) のサポート	部分的

属性ベースのアクセスコントロール (ABAC) は、属性に基づいて権限を定義する認可戦略です。AWS では、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール)、および多数の AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合に操作を許可するように ABAC ポリシーを設計します。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの Condition 要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーのすべてをサポートする場合、そのサービスでのサポート状況の値は「はい」になります。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、『IAM ユーザーガイド』の「ABAC とは?」を参照してください。ABAC を設定する手順を示したチュートリアルを表示するには、IAM ユーザーガイドの 「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

Amazon Transcribe リソースのタグ付けの詳細については、「リソースのタグ付け」を参照してください。タグベースのアクセス制御の詳細については、「タグを使用した AWS リソースへのアクセスの制御」 を参照してください。

Amazon Transcribe での一時的な認証情報の使用

一時的な認証情報のサポート	あり

AWS のサービス には、一時的な認証情報を使用してサインインしても機能しないものがあります。一時的な認証情報で機能する AWS のサービスなどの詳細については、「IAM ユーザーガイド」の「IAM と連携する AWS のサービス」を参照してください。

ユーザー名とパスワード以外の方法で AWS Management Console にサインインする場合は、一時的な認証情報を使用していることになります。例えば、会社の Single Sign-On (SSO) リンクを使用して AWS にアクセスすると、そのプロセスは自動的に一時認証情報を作成します。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、『IAM ユーザーガイド』の「ロールへの切り替え (コンソール)」を参照してください。

一時認証情報は、AWS CLI または AWS API を使用して手動で作成できます。作成後、一時的な認証情報を使用して AWS にアクセスできるようになります。AWS は、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

Amazon Transcribe のクロスサービスプリンシパル権限

フォワードアクセスセッション (FAS) をサポート	はい

IAM ユーザーまたはロールを使用して AWSでアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行してから、別のサービスの別のアクションを開始することがあります。FAS は、AWS のサービス を呼び出すプリンシパル権限と、リクエスト AWS のサービス を組み合わせ、ダウンストリームのサービスに対してリクエストを行います。FAS リクエストは、サービスが、完了するために他の AWS のサービス またはリソースとのやりとりを必要とするリクエストを受け取ったときにのみ行われます。この場合、両方のアクションを実行するための権限が必要です。FASリクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Amazon Transcribe のサービスロール

サービスロールに対するサポート	あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービス にアクセス許可を委任するロールの作成」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、Amazon Transcribe の機能が破損する可能性があります。Amazon Transcribe が指示する場合以外は、サービスロールを編集しないでください。

Amazon Transcribe のサービスリンクロール

サービスにリンクされたロールのサポート	いいえ

サービスリンクロールは、AWS のサービスにリンクされているサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスリンクロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスリンクロールの権限を表示できますが、編集することはできません。

Amazon Transcribe はサービスにリンクされたロールをサポートしていません。

他のサービスに対するサービスにリンクされたロールの作成または管理の詳細については、IAM と提携する AWS のサービス」を参照してください。表の中から、[サービスにリンクされたロール] 列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

サービス間の混乱した代理の防止

混乱した代理人とは、別のエンティティから強制的にアクションを実行させられるエンティティ（サービスまたはアカウント）です。これには、クロスアカウントおよびクロスリ

AWSアカウントのリソースへのアクセスが付与されたサービスプリンシパルを持つすべてのサービスのデータを保護するのに役立つツールを提供していますAWS アカウント。このセクションでは、Amazon Transcribeに固有のサービス間混乱防止に焦点を当てていますが、このトピックの詳細については、IAMユーザーガイドの「混乱した代理人の問題」セクションを参照してください。

IAMAmazon Transcribeリソースへのアクセス権限を制限するには、aws:SourceArnaws:SourceAccountグローバル条件コンテキストキーとリソースポリシーを使用することをお勧めします。

これらのグローバル条件コンテキストキーを両方使用する場合、aws:SourceArn値に ID が付いた ID が付いた ID が付いたAWS アカウント ID を指定する場合、aws:SourceAccount値と、AWS アカウント同じポリシーステートメントで使用する場合、値と、それらが同じポリシーステートメントで使用する場合、値と、AWS アカウント値の IDaws:SourceArn を使用する必要があります。

クロスサービスのアクセスにリソースを 1 つだけ関連付けたい場合は、aws:SourceArn を使用します。AWS アカウントその中のリソースをクロスサービスアクセスに関連付ける場合は、を使用してくださいaws:SourceAccount。

> **Note**
>
> 混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して、リソースの ARN 全体が付いたaws:SourceArnグローバル条件コンテキストキーを使用することです。ARN 全体が不明または複数のリソースを指定する場合、ARN の未知部分にワイルドカード (*) が付いたaws:SourceArnグローバルコンテキスト条件キーを使用します。例えば、arn:aws:transcribe::*123456789012*:*。

代理人の混乱を防ぐ方法を示す役割を引き受けるポリシーの例については、を参照してください混乱した代理の防止。

Amazon Transcribe アイデンティティベースポリシーの例

デフォルトでは、 ユーザーおよびロールには、Amazon Transcribe リソースを作成または変更するアクセス許可はありません。また、AWS Management Console、AWS Command Line Interface (AWS CLI)、または AWS API を使用してタスクを実行することもできません。IAM 管理者は、リソースで必要なアクションを実行するためのアクセス許可をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、IAM ユーザーガイドの「IAM ポリシーの作成」を参照してください。

リソースタイプごとの ARN の形式を含む、Amazon Transcribe で定義されるアクションとリソースタイプの詳細については、「サービス認証リファレンス」の「Amazon Transcribe のアクション、リソース、および条件キー」を参照してください。

トピック

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰かが Amazon Transcribe リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、AWS アカウントに追加料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください。

- AWS マネージドポリシーを使用して開始し、最小特権の許可に移行する – ユーザーとワークロードへの許可の付与を開始するには、多くの一般的なユースケースのために許可を付与する AWS マネージドポリシーを使用します。これらは AWS アカウント で使用できます。ユースケースに応じた AWS カスタマーマネージドポリシーを定義することで、許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定するときは、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。

- IAM ポリシーで条件を使用してアクセスをさらに制限する – ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定することができます。また、AWS のサービス などの特定の AWS CloudFormation を介して使用する場合、条件を使用してサービスアクションへのアクセスを許可することもできます。詳細については、[IAM User Guide] (IAM ユーザーガイド) の [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素：条件) を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な許可を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM Access Analyzer は 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーを作成できるようサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – AWS アカウント で IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

AWS Management Consoleの使用

Amazon Transcribe コンソールにアクセスするには、許可の最小限のセットが必要です。これらのアクセス許可により、AWS アカウント の Amazon Transcribe リソースの詳細をリストおよび表示できます。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソール許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

エンティティ (ユーザーとロール) が確実に使用できるようにするにはAWS Management Console、AWS以下の管理ポリシーのいずれかをエンティティ (ユーザーとロール) にアタッチします。

- `AmazonTranscribeFullAccess`:Amazon Transcribe すべてのリソースの作成、更新、削除、および実行を行うためのフルアクセスを付与します。また、Amazon S3バケット名にが含まれるバケットへのアクセスも許可されます。`transcribe`
- `AmazonTranscribeReadOnlyAccess`: 字起こしジョブとカスタム語彙が表示できるように、Amazon Transcribeリソースへの読み取り専用アクセスを許可します。

> **Note**
>
> にサインインしてポリシー名で検索すると、管理対象アクセス許可ポリシーが表示されます。IAMAWS Management Console「転記」を検索すると、上記の両方のポリシー (AmazonTranscribeReadOnlyと AmazonTranscribeFullAccess) が返されます。

独自のカスタム IAM ポリシーを作成して、Amazon Transcribe API アクションにアクセス権限を付与することもできます。これらのカスタムポリシーは、それらのアクセス許可が必要なエンティティにアタッチできます。

IAMロールに必要なアクセス許可

IAM呼び出すロールを作成する場合Amazon Transcribe、Amazon S3そのロールにはバケットにアクセスする権限が必要です。該当する場合は、KMS keyを使用してバケットの内容を暗号化する必要もあります。ポリシーの例については、次のセクションを参照してください。

信頼ポリシー

IAM字起こしリクエストを行うために使用するエンティティには、Amazon Transcribeそのロールを引き受けることを可能にする信頼ポリシーが必要です。Amazon Transcribe以下の信頼ポリシーを使用してください。通話後分析を有効にしてリアルタイムの通話分析リクエストを行う場合は、「リアルタイム通話分析の信頼ポリシー」を使用する必要があることに注意してください。

の信頼ポリシー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:*"
        }
      }
    }
  ]
}
```

リアルタイム通話分析の信頼ポリシー

```
{
  "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入力バケットポリシー

次のポリシーは、IAM指定された入力バケットからファイルにアクセスする権限をロールに与えます。

```
{
    "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出力バケットポリシー

次のポリシーでは、IAM指定された出力バケットにファイルを書き込むためのロールに付与します。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-OUTPUT-BUCKET/*"
        ]
    }
}
```

Amazon S3暗号化キーに必要な権限

KMS keyAmazon S3を使用してバケットを暗号化する場合は、KMS keyポリシーに以下を含めてください。これにより、Amazon Transcribe にバケットの内容へのアクセスが許可されます。へのアク

セスを許可する方法の詳細についてはKMS keys、『AWS KMS開発者ガイド』KMS keyの「AWS アカウント外部からのアクセスの許可」を参照してください。

```
{
  "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"
    }
  ]
}
```

ユーザーが自分の許可を表示できるようにする

この例では、ユーザーアイデンティティに添付されたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーを作成する方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了するアクセス許可が含まれています。

```
{
    "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暗号化コンテキストポリシー

次のポリシーでは、IAMロール「ExampleRole」AWS KMS KMS keyにこの特定の復号化操作と暗号化操作を使用する権限を付与します。このポリシーは、少なくとも 1 つの暗号化コンテキストペア (この場合は "color:indigoBlue」) を持つリクエストに対してのみ機能します。AWS KMS暗号化コンテキストの詳細については、を参照してくださいAWS KMS の暗号化コンテキスト。

```
{
  "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"
```

```
            }
          }
        }
    ]
}
```

混乱した代理の防止

次に示すのは、役割を引き受けたポリシーの例です。このポリシーでは、aws:SourceArnaws:SourceAccountAmazon Transcribe代理人の案件が混乱するのを防ぐための使い方と使い方を示しています。混乱した副作用の防止の詳細については、を参照してくださいサービス間の混乱した代理の防止。

```
{
  "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:*"
        }
      }
    }
  ]
}
```

タグに基づく字起こしジョブの表示

アイデンティティベースのポリシーの条件を使用して、タグに基づいて Amazon Transcribe リソースへのアクセスをコントロールできます。この例では、字起こしジョブを表示できるポリシーを作成する方法を示します。ただし、Owner字起こしジョブタグにそのユーザーのユーザー名の値がある場

合のみ、アクセス許可は付与されます。このポリシーでは、を使用してこのアクションを実行するために必要なアクセス許可も付与しますAWS Management Console。

IAMこのポリシーをアカウントのエンティティにアタッチできます。test-role指定されたロールが文字起こしジョブを表示しようとすると、Owner=test-roleその文字起こしジョブにタグを付ける必要がありますowner=test-role (条件キー名は大文字と小文字が区別されません)。そうしないと、アクセスが拒否されます。詳細については、IAMIAMユーザーガイドの「JSON ポリシー要素:条件」を参照してください。

タグ付けの詳細についてはAmazon Transcribe、を参照してくださいリソースのタグ付け。

```
{
    "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}"}
            }
        }
    ]
}
```

Amazon Transcribe アイデンティティとアクセスに関するトラブルシューティング

以下の情報は、Amazon Transcribe と AWS Identity and Access Management (IAM) の使用に伴って発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- Amazon Transcribe でアクションを実行する権限がない

- iam:PassRole
- 自分の AWS アカウント 以外のユーザーに Amazon Transcribe リソースへのアクセスを許可したい

Amazon Transcribe でアクションを実行する権限がない

あるアクションを実行する権限がないというエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要な transcribe:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 transcribe:GetWidget on resource: my-example-widget
```

この場合、transcribe:*GetWidget* アクションを使用して *my-example-widget* リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。サインイン資格情報を提供した担当者が管理者です。

iam:PassRole

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon Transcribe にロールを渡すことができるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールやサービスにリンクされたロールを作成せずに、既存のロールをサービスに渡すことができます。そのためには、サービスにロールを渡す許可が必要です。

以下の例のエラーは、 marymajor という IAM ユーザーがコンソールを使用して Amazon Transcribe でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与されたアクセス許可が必要です。Mary には、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、メアリーのポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。サインイン資格情報を提供した担当者が管理者です。

自分の AWS アカウント 以外のユーザーに Amazon Transcribe リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外のユーザーが、リソースへのアクセスに使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定することができます。リソースベースのポリシーまたはアクセス制御リスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下をご参照ください。

- Amazon Transcribe がこれらの機能をサポートしているかどうかを確認するには、Amazon Transcribe と IAM の連携方法 をご参照ください。
- 所有している AWS アカウント 全体のリソースへのアクセス権を提供する方法については、IAM ユーザーガイドの「所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに提供」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「第三者が所有する AWS アカウント へのアクセス権を付与する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

Amazon Transcribe でのデータ保護

AWS 責任共有モデルは、Amazon Transcribe でのデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護するがあります。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービスのセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバシーのよく

ある質問」を参照してください。欧州でのデータ保護の詳細については、「AWS セキュリティブログ」に投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データを保護するため、AWS アカウント の認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみを各ユーザーに付与できます。また、次の方法でデータを保護することをおすすめします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須です。TLS 1.3 が推奨されます。
- AWS CloudTrail で API とユーザーアクティビティロギングをセットアップします。
- AWS のサービス内でデフォルトである、すべてのセキュリティ管理に加え、AWS の暗号化ソリューションを使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API により AWS にアクセスするときに FIPS 140-2 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、AWS CLI、または AWS SDK で Amazon Transcribe または他の AWS のサービス を使用する場合も同様です。タグ、または名前に使用される自由形式のテキストフィールドに入力されるデータは、請求または診断ログに使用される場合があります。外部サーバーへ URL を供給する場合は、そのサーバーへのリクエストを検証するために、認証情報を URL に含めないことを強くおすすめします。

ネットワーク間トラフィックのプライバシー

Amazon Transcribe の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントは、Amazon Transcribe のみへの接続を許可する VPC 内の論理エンティティです。Amazon VPC は リクエストを Amazon Transcribe にルーティングし、VPC に応答をルーティングします。詳細については、「AWS PrivateLink の概念」を参照してください。Amazon Transcribe での Amazon VPC エンドポイントの使用の詳細については、「Amazon Transcribe とインターフェース VPC エンドポイント (AWS PrivateLink)」を参照してください。

データ暗号化

データ暗号化とは、転送中と不使用時のいずれもデータの保護を指します。転送中は、Amazon S3KMS keysマネージドキーまたは保管中のデータを、標準のトランスポート層セキュリティ (TLS) と併用して保護できます。

保管中の暗号化

Amazon Transcribeは、Amazon S3Amazon S3バケットに配置された文字起こし内容のサーバー側の暗号化にデフォルトキー (SSE-S3) を使用します。

操作を使用するときは、KMS key独自の`StartTranscriptionJob`操作を指定して、文字起こしジョブからの出力を暗号化できます。

Amazon Transcribe では、デフォルトキーで暗号化された Amazon EBS ボリュームが使用されます。

転送中の暗号化

Amazon Transcribe は、TLS 1.2 を AWS 証明書で使用して、転送中のデータを暗号化します。ストリーミングの文字起こしも対象になります。

キーの管理

Amazon TranscribeによりKMS keys、データの暗号化を強化できます。を使用するとAmazon S3、文字起こしジョブを作成する際に、入力メディアを暗号化できます。AWS KMSとの統合により、`StartTranscriptionJob`リクエストからの出力を暗号化できます。

を指定しない場合KMS key、Amazon S3文字起こしジョブの出力はデフォルトキー (SSE-S3) で暗号化されます。

AWS KMS の詳細については、AWS Key Management Service デベロッパーガイド を参照してください。

を使用したキー管理AWS Management Console

文字起こしジョブの出力を暗号化するには、KMS keyAWS アカウントリクエストを行う側の「」を使用するか、別のユーザーの「」を使用するかを選択できますAWS アカウント。KMS key

を指定しない場合KMS key、Amazon S3文字起こしジョブの出力はデフォルトキー (SSE-S3) で暗号化されます。

出力暗号化を有効にするには:

1. [Output data (出力データ)] で、[Encryption (暗号化)] を選択します。

2. KMS keyAWS アカウント現在使用しているものか、別のものかを選択しますAWS アカウント。現在のキーを使用する場合はAWS アカウント、KMS keyID からキーを選択します。別のキーを使用する場合はAWS アカウント、キーの ARN を入力する必要があります。別のキーを使用するにはAWS アカウント、kms:Encrypt呼び出し元がに対する権限を持っている必要がありますKMS key。詳細については、「キーポリシーの作成」を参照してください。

API を使用したキーの管理

API で出力暗号化を使用するには、、StartCallAnalyticsJobStartMedicalTranscriptionJob、KMS keyOutputEncryptionKMSKeyIdStartTranscriptionJobまたはオペレーションのパラメータを使用してを指定する必要があります。

現在のキーを使用する場合はAWS アカウント、次の 4KMS key つの方法のいずれかでキーを指定できます。

1. KMS keyID 自体を使用してください。例えば、1234abcd-12ab-34cd-56ef-1234567890ab。
2. KMS keyID にはエイリアスを使用してください。例えば、alias/ExampleAlias。
3. KMS keyID には Amazon リソースネーム (ARN) を使用してください。例えば、arn:aws:kms:region:account-ID:key/1234abcd-12ab-34cd-56ef-1234567890ab。
4. KMS keyエイリアスには ARN を使用してください。例えば、arn:aws:kms:region:account-ID:alias/ExampleAlias。

AWS アカウント現在のキーとは異なるキーを使用する場合はAWS アカウント、次の 2KMS key つの方法のいずれかでキーを指定できます。

1. KMS keyID には ARN を使用してください。例えば、`arn:aws:kms:region:account-ID:key/1234abcd-12ab-34cd-56ef-1234567890ab`。
2. KMS keyエイリアスには ARN を使用してください。例えば、`arn:aws:kms:region:account-ID:alias/ExampleAlias`。

リクエストを行うエンティティには、指定したを使用するためのアクセス許可が必要であることに注意してくださいKMS key。

AWS KMS の暗号化コンテキスト

AWS KMS 暗号化コンテキストは、プレーンテキスト、非シークレットキー: 値ペアのマップです。このマップは、暗号化コンテキストペアと呼ばれる追加の認証データを表し、データのセキュリティレイヤーを追加できます。 Amazon Transcribeお客様指定のAmazon S3バケットに文字起こし出力を暗号化するための対称暗号化キーが必要です。詳細については、の「非対称キー」を参照してくださいAWS KMS。

暗号化コンテキストペアを作成するときは、機密情報を含めないでください。暗号化コンテキストは秘密ではありません。CloudTrailログ内でプレーンテキストで見ることができます (したがって、暗号化操作を特定し分類するために使用することができます)。

暗号化コンテキストのキーと値には、アンダースコア (_)、ダッシュ ()、スラッシュ (,-)、コロン (/\:) などの特殊文字を含めることができます。

> **Tip**
>
> 暗号化コンテキストペアの値を暗号化されるデータに関連付けると便利です。必須ではありませんが、ファイル名、ヘッダー値、暗号化されていないデータベースフィールドなど、暗号化されたコンテンツに関連する機密性のないメタデータを使用することをお勧めします。

API による出力暗号化を使用するには、`KMSEncryptionContext` パラメータを `StartTranscriptionJob` オペレーションに設定します。出力暗号化操作に暗号化コンテキストを提供するために、`OutputEncryptionKMSKeyId`パラメータは対称型KMS key ID を参照する必要があります。

IAMポリシーでAWS KMS条件キーを使用すると、KMS key暗号化操作のリクエストで使用された暗号化コンテキストに基づいて、対称暗号化へのアクセスを制御できます。暗号化コンテキストポリシーの例については、を参照してくださいAWS KMS暗号化コンテキストポリシー。

暗号化コンテキストの使用は任意ですが、推奨されています。暗号化コンテキストの詳細については、「Encription context」を参照してください。

サービス改善のためのデータの使用をオプトアウトする

デフォルトでは、Amazon Transcribe処理した音声入力を保存および使用して、サービスを開発し、お客様のエクスペリエンスを継続的に改善します。Amazon TranscribeAWS Organizationsオプトアウトポリシーを使用することで、お客様のコンテンツが開発および改善のために使用されることを拒否することができます。オプトアウトの方法の詳細については、「AI サービスのオプトアウトポリシー」を参照してください。

モニタリング Amazon Transcribe

モニタリングは、 Amazon Transcribe およびその他の AWS ソリューションの信頼性、可用性、およびパフォーマンスを維持する上で重要な部分です。 は、 をモニタリングし Amazon Transcribe、問題が発生したときに報告し、必要に応じて自動アクションを実行するために、以下のモニタリングツール AWS を提供します。

- Amazon CloudWatch は、 リソースと、 で実行しているアプリケーションを AWS リアルタイムでモニタリングします AWS 。メトリクスを収集および追跡し、カスタマイズされたダッシュボードを作成し、指定されたメトリックが指定したしきい値に達したときに通知またはアクションを実行するアラームを設定できます。例えば、 で Amazon EC2 インスタンスの CPU 使用率やその他のメトリクス CloudWatch を追跡し、必要に応じて新しいインスタンスを自動的に起動できます。
- Amazon CloudWatch Logs は、 Amazon EC2 インスタンスやその他のソースからログファイルをモニタリング、保存 CloudTrail、およびアクセスできます。 CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値に達したときに通知できます。高い耐久性を備えたストレージにログデータをアーカイブすることもできます。
- AWS CloudTrail は、 によって、または に代わって行われた API コールおよび関連イベントをキャプチャ AWS アカウント し、指定した Amazon S3 バケットにログファイルを配信します。を呼び出したユーザーとアカウント AWS、呼び出し元のソース IP アドレス、呼び出しが発生した日時を特定できます。

詳細については、『Amazon CloudWatch ユーザーガイド』を参照してください。

Amazon EventBridge は、イベントを使用してアプリケーションコンポーネントを接続できるサーバーレスサービスです。これにより、スケーラブルなイベント駆動型アプリケーションを簡単に構築できます。 は、独自のアプリケーション、Software as a Service (SaaS) アプリケーション、および AWS のサービスからリアルタイムデータのストリームを EventBridge 配信し、そのデータを などのターゲットにルーティングします Lambda。サービスで発生したイベントをモニタリングし、イベント駆動型アーキテクチャを構築できます。詳細については、『Amazon EventBridge ユーザーガイド』を参照してください。

トピック

- Amazon Transcribe によるモニタリング Amazon CloudWatch
- によるモニタリング Amazon TranscribeAWS CloudTrail
- Amazon EventBridge で を使用する Amazon Transcribe

Amazon Transcribe によるモニタリング Amazon CloudWatch

raw データを収集し CloudWatch、読み取り可能なほぼリアルタイムのメトリクスに処理する Amazon Transcribe を使用してモニタリングできます。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、『CloudWatch ユーザーガイド』を参照してください。

での Amazon CloudWatch メトリクスとディメンションの使用 Amazon Transcribe

Amazon Transcribe は、パフォーマンスのモニタリングに役立つデータである CloudWatch メトリクスとディメンションをサポートしています。サポートされているメトリクスカテゴリには、文字起こしジョブに関連するトラフィック、エラー、データ転送、レイテンシーなどがあります。サポートされているメトリクスは、AWS/Transcribe 名前空間 CloudWatch の を介して配置されます。

> **Note**
>
> CloudWatch モニタリングメトリクスは無料で、 CloudWatch サービスクォータにはカウントされません。

CloudWatch メトリクスの詳細については、「メトリクスの使用 Amazon CloudWatch」を参照してください。

によるモニタリング Amazon TranscribeAWS CloudTrail

Amazon Transcribe は AWS CloudTrail、 AWS Identity and Access Management （ IAM) ユーザーまたはロール、または service. CloudTrail captures Amazon Transcribe によって で実行されたアクションを記録する AWS サービスである と統合されています。 は、 のすべての API コールを取得します Amazon Transcribe。これには、 からの呼び出し AWS Management Console と、イベントとしての Amazon Transcribe APIsへのコード呼び出しが含まれます。証跡を作成することで、 の CloudTrail イベントを含むイベントの バケット Amazon Transcribeへの Amazon S3 継続的な配信を有効にすることができます。証跡を作成しない場合でも、イベント履歴の CloudTrail AWS Management Console で最新のイベントを表示することはできます。によって収集された情報を使用して CloudTrail、 に対して行われた各リクエスト Amazon Transcribe、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、「 AWS CloudTrail ユーザーガイド」を参照してください。

Amazon Transcribe および CloudTrail

CloudTrail アカウントを作成する AWS アカウント と、 で が有効になります。でアクティビティが発生すると Amazon Transcribe、そのアクティビティは CloudTrail イベント履歴 の他の AWS のサービス イベントとともにイベントに記録されます。 CloudTrail で最近のイベントを表示、検索、ダウンロードできます AWS アカウント。詳細については、「CloudTrail イベント履歴でのイベントの表示」を参照してください。

のイベントなど AWS アカウント、 のイベントの継続的な記録を取得するには Amazon Transcribe、証跡を作成します。証跡は、 がイベント CloudTrail をログファイルとして指定された Amazon S3 bucket. CloudTrail log files に配信できるようにする設定です。ログファイルには 1 つ以上のログエントリが含まれます。イベント は、任意の送信元からの単一の要求を表します。これには、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。 CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

デフォルトでは、 で証跡を作成すると AWS Management Console、証跡はすべての に適用されます AWS リージョン。証跡は AWS 、パーティション AWS リージョン 内のすべての からのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、他の を設定 AWS のサービス して、 CloudTrail ログで収集されたイベントデータをさらに分析し、それに基づく対応を行うことができます。詳細については、以下をご覧ください。

- 証跡を作成するための概要
- CloudTrail サポートされているサービスと統合

- Amazon SNS の通知の設定 CloudTrail
- 「複数のリージョンから CloudTrail ログファイルを受け取る」と「複数のアカウントから CloudTrail ログファイルを受け取る」

CloudTrail は、 API リファレンス に記載されているすべての Amazon Transcribe アクションを記録します。例えば、、CreateVocabulary、および StartTranscriptionJobオペレーションではGetTranscriptionJob、 CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。この情報は以下のことを確認するのに役立ちます:

- リクエストがルート認証情報と IAM ユーザー認証情報のどちらを使用して行われたか
- リクエストが、 IAM ロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報によって行われたか
- リクエストが別の によって行われたかどうか AWS のサービス

詳細については、「CloudTrail userIdentity 要素」を参照してください。

複数の AWS リージョン と複数の の Amazon Transcribe ログファイルを 1 つの Amazon S3 バケット AWS アカウント に集約することもできます。詳細については、「複数のリージョンからの CloudTrail ログファイルの受信」および「複数のアカウントからの CloudTrail ログファイルの受信」を参照してください。

例: Amazon Transcribe ログファイルエントリ

証跡は、指定した Amazon S3 bucket.log ファイルへのログファイルとしてイベントを配信できるようにする設定です。 CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは、任意の送信元からの単一の要求を表します。これには、リクエストされたアクションに関する情報がアクションの日時として含まれ、リクエストパラメータ. CloudTrail ログファイルはパブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

StartTranscriptionJob および GetTranscriptionJob API オペレーションを呼び出すと、以下のエントリが作成されます。

```
{
    "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"
        }
    ]
}
```

Amazon EventBridge で を使用する Amazon Transcribe

を使用すると Amazon EventBridge、他の で Amazon Transcribe イベントを開始することで、ジョブの状態の変化に対応できます AWS のサービス。文字起こしジョブの状態が変わると、 はイベント EventBridge をイベントストリームに自動的に送信します。イベントストリーム内でモニタリングするイベントと、それらのイベントが発生したときに EventBridge によって実行されるアクションを定義するルールを作成します。例えば、イベントを別のサービス (ターゲット) にルーティングし、そこからアクションが実行されるようにできます。例えば、文字起こしジョブが正常に完了したときにイベントを AWS Lambda 関数にルーティングするようにルールを設定できます。EventBridge ルールを定義するには、以下のセクションを参照してください。

イベントの通知は、E メール、AWS Chatbot チャット通知、AWS Console Mobile Application プッシュ通知など、複数のチャネルを通じて受け取ることができます。「コンソール通知センター」で通知を確認することもできます。通知を設定する場合は、. AWS User Notifications supports 集約を使用できますAWS User Notifications。これにより、特定のイベント中に受信する通知の数を減らすことができます。

EventBridge ルールの定義

EventBridge ルールを定義するには、 を使用しますAWS Management Console。ルールを定義するときは、サービス名として Amazon Transcribe を使用します。 EventBridge ルールの作成方法の例については、「 ルールAmazon EventBridge」を参照してください。

を使用する前に EventBridge、次の定義に注意してください。

- イベント - イベントは、文字起こしジョブのいずれかの状態の変化を示します。例えば、ジョブの `TranscriptionJobStatus` が `IN_PROGRESS` から `COMPLETED` に変わります。
- ターゲット - ターゲットは、イベントを処理する別の AWS のサービス です。例えば、 AWS Lambda または Amazon Simple Notification Service （ Amazon SNS) です。ターゲットは、JSON 形式のイベントを受け取ります。
- ルール – ルールは、監視する受信イベント EventBridge を照合し、処理のためにターゲットにルーティングします。ルールによってイベントが複数のターゲットにルーティングされた場合、すべてのターゲットではそのイベントが並行して処理されます。ルールでは、ターゲットに送信される JSON をカスタマイズできます。

Amazon EventBridge イベントはベストエフォートベースで発行されます。でのイベントの作成と管理の詳細については EventBridge、「 ユーザーガイド」の「 Amazon EventBridge イベントAmazon EventBridge 」を参照してください。

以下は、文字起こしジョブのステータス Amazon Transcribe が COMPLETEDまたは に変わったときに開始される の EventBridge ルールの例ですFAILED。

```
{
    "source": [
        "aws.transcribe"
    ],
    "detail-type": [
        "Transcribe Job State Change"
    ],
    "detail": {
        "TranscriptionJobStatus": [
            "COMPLETED",
            "FAILED"
        ]
    }
}
```

ルールには以下のフィールドがあります。

- source - イベントのソース。の場合 Amazon Transcribe、これは常に ですaws.transcribe。
- detail-type - イベントの詳細の識別子。 Amazon Transcribeの場合、これは常に Transcribe Job State Change です。
- detail - 文字起こしジョブの新しいステータス。この例のルールでは、ジョブのステータスが COMPLETED または FAILED に変わったときにイベントが開始されます。

Amazon Transcribe イベント

Amazon EventBridge はいくつかの Amazon Transcribe イベントを記録します。

- 文字起こしジョブイベント
- 言語識別イベント
- コール分析イベント
- コール分析通話後のイベント
- 語彙イベント

これらのイベントにはすべて、以下の共有フィールドが含まれます。

- version: イベントデータのバージョン。この値は常に 0 です。
- id: イベント EventBridge に対して によって生成された一意の識別子。
- detail-type: イベントの詳細の識別子。例えば Transcribe Job State Change です。
- source: イベントのソース。このため Amazon Transcribe 、これは常に で すaws.transcribe。
- account: API コールを生成したアカウントの AWS アカウント ID。
- time: イベントが配信された日時。
- region: リクエスト AWS リージョン が行われた 。
- resources: API コールによって使用されたリソース。の場合 Amazon Transcribe、このフィールドは常に空です。
- detail: イベントに関するその他の詳細
  - FailureReason: このフィールドは、状態またはステータスが FAILED に変化した場合に表示され、FAILED の状態またはステータスになった理由が説明されます。
  - 各イベントタイプには、detail の下に表示される追加の固有のフィールドがあります。これらの固有のフィールドは、各イベント例の後に続く以下のセクションで定義されています。

文字起こしジョブイベント

ジョブの状態が から COMPLETEDまたは IN_PROGRESSに変わるとFAILED、 はイベント Amazon Transcribe を生成します。ターゲットの状態を変更してイベントを開始させたジョブを特定するには、イベントの TranscriptionJobName フィールドを使用します。 Amazon Transcribe イベントには、次の情報が含まれます。文字起こしジョブのステータスが FAILED の場合、detail の下に FailureReason フィールドが追加されます。

このイベントは StartTranscriptionJob API オペレーションにのみ適用されることに注意してください。

```
{
    "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: 文字起こしジョブに選択した固有の名前。
- TranscriptionJobStatus : 文字起こしのジョブのステータス。これは、COMPLETED または FAILED です。

言語識別イベント

自動言語識別を有効にすると、 Amazon Transcribe は、言語識別の状態が COMPLETED または FAILED の場合にイベントを生成します。ターゲットの状態を変更してイベントを開始させたジョブを特定するには、イベントの JobName フィールドを使用します。 Amazon Transcribe イベントには、以下の情報が含まれています。言語識別ステータスが FAILED の場合、detail の下に FailureReason フィールドが追加されます。

このイベントは、LanguageIdSettings パラメータが含まれている場合の StartTranscriptionJob API オペレーションにのみ適用されることに注意してください。

```
{
    "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: 文字起こしジョブの場合、この値は TranscriptionJob でなければなりません。
- JobName: 文字起こしジョブの固有の名前。

- LanguageIdentificationStatus: 文字起こしジョブにおける言語識別のステータス。これは、COMPLETED または FAILED です。

コール分析イベント

コール分析ジョブの状態が IN_PROGRESS から COMPLETED または FAILED に変化すると、Amazon Transcribe はイベントを生成します。ターゲットで状態を変更してイベントを開始させたコール分析ジョブを特定するには、イベントの JobName フィールドを使用します。 Amazon Transcribe イベントには、以下の情報が含まれています。コール分析ジョブのステータスが FAILED の場合、detail の下に FailureReason フィールドが追加されます。

このイベントは StartCallAnalyticsJob API オペレーションにのみ適用されることに注意してください。

```
{
    "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"),
        "AnalyticsJobDetails": { // only when you enable optional features such as
 Generative Call Summarization
            "Skipped": []
        }
    }
}
```

- JobName: コール分析文字起こしジョブの固有の名前。
- JobStatus: コール分析文字起こしジョブのステータス。COMPLETED または FAILED のいずれかとなります。
- AnalyticsJobDetails: スキップされた分析機能に関する情報を含む、コール分析文字起こしジョブの詳細。

コール分析通話後のイベント

通話後分析文字起こしが IN_PROGRESS から COMPLETED または FAILED の状態に変化すると、Amazon Transcribe はイベントを生成します。ターゲットで状態を変更してイベントを開始させたコール分析通話後ジョブを特定するには、イベントの StreamingSessionId フィールドを使用します。

このイベントは、PostCallAnalyticsSettings パラメータが含まれている場合の StartCallAnalyticsStreamTranscription API オペレーションにのみ適用されることに注意してください。

COMPLETED イベントには、以下の情報が含まれています。

```
{
    "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 イベントには、以下の情報が含まれています。

```
{
```

```
    "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: リアルタイムコール分析文字起こしリクエストに割り当てられる識別番号。
- PostCallStatus: 通話後コール分析文字起こしジョブのステータス。COMPLETED または FAILED のいずれかとなります。
- Transcript: 編集済みおよび未編集のトランスクリプトの URI。
- Media: 編集済みおよび未編集の音声ファイルの URI。

語彙イベント

カスタム語彙の状態が から READYまたは PENDINGに変わるとFAILED、 はイベント Amazon Transcribe を生成します。ターゲットの状態を変更してイベントを開始させたカスタム語彙を特定するには、イベントの VocabularyName フィールドを使用します。 Amazon Transcribe イベントには、次の情報が含まれます。カスタム語彙の状態が FAILED の場合、detail の下に FailureReason フィールドが追加されます。

このイベントは CreateVocabulary API オペレーションにのみ適用されることに注意してください。

```
{
    "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: カスタム語彙の固有の名前です。
- VocabularyState: カスタム語彙の処理状態。これは、READY または FAILED です。

のコンプライアンス検証 Amazon Transcribe

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、コンプライアンスプログラムAWS のサービス による対象範囲内のコンプライアンスプログラムを参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「 でのレポートのダウンロード AWS Artifact」の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ — このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

> **Note**
>
> すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、「HIPAA 対応サービスのリファレンス」を参照してください。

- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。

- AWS カスタマーコンプライアンスガイド — コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス 、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「Security Hub のコントロールリファレンス」を参照してください。
- Amazon GuardDuty – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。
- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

Amazon Transcribe での耐障害性

AWS グローバルインフラストラクチャは AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。AWS リージョン には、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている物理的に独立・隔離された複数のアベイラビリティーゾーンがあります。アベイラビリティーゾーンを使用すると、中断することなくゾーン間で自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用できます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョン とアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

Amazon Transcribe でのインフラストラクチャセキュリティ

マネージドサービスとして、Amazon TranscribeAWSグローバルネットワークセキュリティによって保護されています。AWSセキュリティサービスと AWS がインフラストラクチャを保護する方法については、「AWS クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected Framework」の「インフラストラクチャ保護」を参照してください。

AWS の発行済み API コールを使用して、ネットワーク経由で Amazon Transcribe にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS) TLS 1.2 および TLS 1.3 をお勧めします。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートです。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon Transcribe の脆弱性分析と管理

構成および IT 管理は、AWS とお客様の間で共有される責任です。詳細については、AWS 責任共有モデルを参照してください。

Amazon Transcribe とインターフェース VPC エンドポイント (AWS PrivateLink)

VPC と Amazon Transcribe とのプライベート接続を確立するには、インターフェース VPC エンドポイントを作成します。インターフェイスエンドポイントは以下から給電されます。AWS PrivateLink、個人的にアクセスできるテクノロジーAmazon Transcribeインターネットゲートウェイ、NAT デバイス、VPN 接続、またはAWS Direct Connect接続。VPC のインスタンスは、パブリック IP アドレスがなくても Amazon Transcribe API と通信できます。VPC と Amazon Transcribe 間のトラフィックは、Amazon ネットワークを離れません。

各インターフェイスエンドポイントは、サブネット内の 1 つ、または複数の Elastic Network Interface によって表されます。

詳細については、Amazon VPC ユーザーガイドの「インターフェイス VPC エンドポイント (AWS PrivateLink) 」を参照してください。

Amazon Transcribe VPC エンドポイントに関する考慮事項

のインターフェイス VPC エンドポイントを設定する前にAmazon Transcribe、必ず確認してくださいインターフェイスエンドポイントのプロパティと制限事項にAmazon VPCユーザーガイド。

Amazon Transcribe は、VPC からのすべての API アクションの呼び出しをサポートしています。

Amazon Transcribe 用のインターフェイス VPC エンドポイントの作成

の VPC エンドポイントを作成できます。Amazon Transcribeを使用するサービスAmazon VPC AWS Management ConsoleまたはAWS CLI。詳細については、を参照してください。インターフェースエンドポイントの作成にAmazon VPCユーザーガイド。

でのバッチ転記用Amazon Transcribe、次のサービス名を使用して VPC エンドポイントを作成します。

- com.amazonaws。*####-2*. 文字起こし

でトランスクリプションをストリーミングする場合Amazon Transcribe、次のサービス名を使用して VPC エンドポイントを作成します。

- com.amazonaws。*####-2*. トランスクリプションストリーミング

エンドポイントのプライベート DNS を有効にすると、AWS リージョンのデフォルト DNS 名 (`transcribestreaming.us-east-2.amazonaws.com` など) を使用して、Amazon Transcribe に対する API リクエストを実行できます。

詳細については、を参照してください。インターフェースエンドポイントからサービスにアクセスするにAmazon VPCユーザーガイド。

Amazon Transcribe 用の VPC エンドポイントポリシーの作成

VPC エンドポイントには、以下のストリーミングサービスまたはバッチトランスクリプションサービスへのアクセスを制御するエンドポイントポリシーをアタッチできます。Amazon Transcribe。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。

- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、を参照してください。VPC エンドポイントによるサービスへのアクセスの制御にAmazon VPCユーザーガイド。

例:の VPC エンドポイントポリシーAmazon Transcribeバッチ転写アクション

以下は、でのバッチ転記に関するエンドポイントポリシーの例です。Amazon Transcribe。エンドポイントにアタッチされると、このポリシーは、すべてのリソースですべてのプリンシパルに、リストされている Amazon Transcribe アクションへのアクセス権を付与します。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartTranscriptionJob",
            "transcribe:ListTranscriptionJobs"
         ],
         "Resource":"*"
      }
   ]
}
```

例:の VPC エンドポイントポリシーAmazon Transcribeストリーミングトランスクリプションアクション

以下は、でのストリーミングトランスクリプションのエンドポイントポリシーの例です。Amazon Transcribe。エンドポイントにアタッチされると、このポリシーは、すべてのリソースですべてのプリンシパルに、リストされている Amazon Transcribe アクションへのアクセス権を付与します。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartStreamTranscription",
```

```
            "transcribe:StartStreamTranscriptionWebsocket"
        ],
        "Resource":"*"
      }
   ]
}
```

共有サブネット

共有されているサブネットの VPC エンドポイントを作成、説明、変更、削除することはできません。ただし、VPC エンドポイントを使用することはできます。VPC 共有の詳細については、を参照してください。VPC を他のアカウントと共有するにAmazon Virtual Private Cloudガイド。

Amazon Transcribe のセキュリティのベストプラクティス

以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションに相当するものではありません。これらのベストプラクティスはお客様の環境に適切ではないか、十分ではない場合があるため、絶対的な解決策ではなく、役に立つ情報としてお考えください。

- AWS KMS暗号化コンテキストなどのデータ暗号化を使用する

  AWS KMS 暗号化コンテキストは、プレーンテキスト、非シークレットキー: 値ペアのマップです。このマップは、暗号化コンテキストペアと呼ばれる追加の認証データを表し、データのセキュリティレイヤーを追加できます。

  詳細については、AWS KMS の暗号化コンテキスト を参照してください。
- できる限り一時的な認証情報を使用する

  長期クレデンシャル (など) の認証情報ではなく、一時的な認証情報をできるだけ使用します。IAM プログラムによるアクセスや長期的な認証情報を持つユーザーが必要なシナリオでは、アクセスキーをローテーションすることをお勧めします。長期的な認証情報を定期的にローテーションすることで、プロセスに慣れることができます。これを行うことで、従業員が退職するときなど、認証情報をローテーションする必要がある場合に役に立ちます。IAMアクセス最終使用者情報を利用して、アクセスキーを安全にローテーションして削除することをお勧めします。

  詳細については、の「アクセスキーのローテーション」と「セキュリティのベストプラクティス」を参照してくださいIAM。
- IAMAWSAmazon Transcribeアクセスを必要とするアプリケーションやサービスにはロールを使用する

IAMロールを使用して、Amazon Transcribeにアクセスする必要があるアプリケーションまたはサービスの一時的な認証情報を管理できます。ロールを使用する場合、Amazon EC2AWSインスタンスまたはのサービス (など) に長期の認証情報 (パスワードやアクセスキーなど) を配布する必要はありません。 IAMロールは、AWSアプリケーションがリソースにリクエストを行うときに使用できる一時的な認証を付与できます。

詳細については、「ロール」および「IAMロールの一般的なシナリオ:ユーザー、アプリケーション、サービス」を参照してください。

- タグベースのアクセスコントロールを使用する

  タグを使用して、内のアクセスを制御できますAWS アカウント。Amazon TranscribeIn. タグは、トランスクリプションジョブ、カスタムボキャブラリ、カスタムボキャブラリフィルター、カスタム言語モデルに追加できます。

  詳細については、タグベースのアクセスコントロール を参照してください。

- AWS監視ツールを使う

  モニタリングは、Amazon Transcribe および AWS ソリューションの信頼性、セキュリティ、可用性、パフォーマンスを維持する上で重要です。Amazon Transcribeを使用して監視できますCloudTrail。

  詳細については、によるモニタリング Amazon TranscribeAWS CloudTrail を参照してください。

- Enable AWS Config (Gems の有効化)

  AWS ConfigAWSリソースの設定を評価、監査、審査できます。を使用するとAWS Config、構成やAWSリソース間の関係の変更を確認することができます。また、詳細なリソース設定履歴を調べ、社内ガイドラインで指定された設定に対して、全体的なコンプライアンスを判断できます。これにより、コンプライアンス監査、セキュリティ分析、変更管理、運用上のトラブルシューティングを簡素化できます。

  詳細については、「What IsAWS Config?」を参照してください。

Amazon Transcribe 医療

Amazon TranscribeMedical は、医師が指示したメモ、医薬品安全モニタリング、遠隔医療の予定、医師と患者の会話など、医療関連の音声を文字起こししたい医療専門家向けに設計された自動音声認識 (ASR) サービスです。 Amazon TranscribeMedical は、リアルタイムストリーミング (マイク経由) またはアップロードされたファイルの文字起こし (バッチ) のいずれかで利用できます。

> **Important**
>
> Amazon Transcribe医療は専門家による医療の助言、診断、治療の代用品ではありません。ユースケースに適した信頼しきい値を特定し、高い精度を必要とする状況では高い信頼しきい値を使用してください。特定のユースケースでは、適切な訓練を受けたレビュー担当者によって人的に結果を見直し、検証する必要があります。 Amazon Transcribe医療専門家による正確さと健全な医療判断の確認後、患者ケアのシナリオでのみ使用してください。

Amazon TranscribeMedical は、責任共有モデルの下で運営されています。AWSはAmazon Transcribe Medical を稼働するインフラストラクチャを保護する責任があり、ユーザーがデータの管理の責任を負います。詳細については、「責任共有モデル」を参照してください。

Amazon TranscribeMedical は英語 (en-US) で利用できます。

最良の結果を得るには、PCM 16 ビットエンコーディングの FLAC や WAV などのロスレスオーディオ形式を使用してください。 Amazon Transcribe Medical16,000 Hz 以上のサンプルレートをサポートします。

成績証明書の分析にはAWS のサービス、次のような他の方法を使用できますAmazon Comprehend Medical。

サポート対象の専門分野

専門分野	サブ専門分野	オーディオ入力
心臓病学	none	ストリーミングのみ
神経学	none	ストリーミングのみ
オンコロジー	none	ストリーミングのみ
プライマリケア	家庭医療	バッチ、ストリーミング

専門分野	サブ専門分野	オーディオ入力
プライマリケア	内科	バッチ、ストリーミング
プライマリケア	産婦人科 (OB-GYN)	バッチ、ストリーミング
プライマリケア	小児科	バッチ、ストリーミング
放射線学	none	ストリーミングのみ
泌尿器科	none	ストリーミングのみ

リージョンとクォータ

AWS リージョンコール分析は以下でサポートされています。

[Region] (リージョン)	文字起こしタイプ
af-south-1 (ケープタウン)	バッチ
ap-east-1 (香港)	バッチ
ap-northeast-1 (東京)	バッチ、ストリーミング
ap-northeast-2 (ソウル)	バッチ、ストリーミング
ap-south-1 (ムンバイ)	バッチ
ap-southeast-1 (シンガポール)	バッチ
ap-southeast-2 (シドニー)	バッチ、ストリーミング
ca-central-1 (カナダ、central-1 (カナダ、central-1)	バッチ、ストリーミング
eu-central-1 (フランクフルト)	バッチ、ストリーミング
eu-north-1 (ストックホルム)	バッチ
eu-west-1 (アイルランド)	バッチ、ストリーミング

[Region] (リージョン)	文字起こしタイプ
eu-west-2 (ロンドン)	バッチ、ストリーミング
eu-west-3 (パリ)	バッチ
me-south-1 (バーレーン)	バッチ
sa-east-1 (サンパウロ)	バッチ、ストリーミング
us-east-1 (バージニア北部)	バッチ、ストリーミング
us-east-2 (オハイオ)	バッチ、ストリーミング
us-gov-east-1 (GovCloud、米国東部)	バッチ、ストリーミング
us-gov-west-1 (GovCloud、米国西部)	バッチ、ストリーミング
us-west-1 (サンフランシスコ)	バッチ
us-west-2 (オレゴン)	バッチ、ストリーミング

、、Amazon TranscribeおよびコールAmazon Transcribe Medical、サポートされる地域が異なることに注意してください。 。

サポートされている各リージョンのエンドポイントを取得するには、『AWS一般リファレンス』の「サービスエンドポイント」を参照してください。

文字起こしに関連するクォータのリストについては、AWS一般リファレンスのサービスクォータを参照してください。一部のクォータは、リクエストに応じて変更することができます。「調整可能」列に「はい」と表示されている場合は、増額をリクエストできます。これを行うには、提供されたリンクを選択します。

医療専門分野と用語

医療文字起こしジョブを作成するときは、ソースファイルの言語、医療専門分野、オーディオタイプを指定します。言語および PRIMARYCARE 医療専門分野として、米国英語 (en-US) を入力します。値として初期診療を入力すると、次の医療分野のソースオーディオから文字起こしを生成できます。

- 家庭医療

- 内科
- 産婦人科 (OB-GYN)
- 小児科

オーディオタイプに応じて、ディクテーションと会話のどちらかを選択できます。医師が患者の訪問または処置に関する報告を提出している音声ファイルのディクテーションを選択します。医師と患者間の会話、または医師間の会話を伴う音声ファイルの会話を選択します。

文字起こしジョブの出力を保存するには、すでに作成した Amazon S3 バケットを選択します。Amazon S3バケットの詳細については、「はじめに」を参照してくださいAmazon Simple Storage Service。

サンプル JSON に入力するリクエストパラメータの最小数を次に示します。

```
{
   "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 Transcribe医療を使用すると、代替文字起こしを生成できます。詳細については、「代替文字起こしの生成」を参照してください。

また、スピーカーの分割を有効にしたり、オーディオ内のチャンネルを特定することもできます。詳細については、「話者パーティショニングの有効化」および「マルチチャネルの音声文字起こし」を参照してください。

医療用語と測定値の文字起こし

Amazon Transcribe医学用語や測定値を文字起こしできます。 Amazon Transcribe医学会話用語の略語を出力します。たとえば、「血圧」は BP として文字起こしされます。このページの表には、Amazon Transcribe医療用語や測定値に使用する規則のリストがあります。[Spoken Term (音声用語)] 列は、ソースオーディオで話される用語を指します。[Output (出力)] 列は、文字起こし結果に表示される略語を示します。

ソースオーディオで話される用語が、ここで文字起こしの出力にどのように対応しているかを確認できます。

ソースオーディオで発音される用語	出力で使用される略語	出力例
摂氏	C	患者の体温は 37.4°C である。
摂氏	C	患者の体温は 37.4°C である。
華氏	F	患者の体温は 101F である。
グラム	g	100g の塊を患者から抽出した。
メートル	m	患者の身長は 1.8m である。
フィート	フィート	患者の身長は 6 フィートである。
キロ	kg	患者の体重は 80kg である。
キログラム	kg	患者の体重は 80kg である。
c c	cc	患者は 100cc の生理食塩水を摂取した。
立方センチメートル	cc	患者は 100cc の生理食塩水を摂取した。
ミリリットル	mL	患者は 100mL の尿を排泄した。
血圧	BP	患者の BP が上昇した。
b p	BP	患者の BP が上昇した。
X オーバー Y	X/Y	患者の BP は 120/80 であった。

ソースオーディオで発音される用語	出力で使用される略語	出力例
心拍数	BPM	患者は 160 BPM の心拍数の心房細動を発症した。
心拍数	BPM	患者は 160 BPM の心拍数の心房細動を発症した。
O 2	O2	患者の O2 飽和度は 98% であった。
CO2	CO2	患者は CO2 上昇のために呼吸支援を必要とした。
術後	術後	患者は、術後評価のために来院した。
術後	術後	患者は、術後評価のために来院した。
CAT スキャン	CT スキャン	脳出血の患者には CT スキャンの使用が必要である。
脈 80	P 80	患者のバイタルは、P 80、R 17、...
呼吸 17	R 17	患者のバイタルは、P 80、R 17、...
入出力	I/O	患者は I/O 洞調律であった
L 5	L5	L4 と L5 の間で腰椎穿刺を実施した

文字起こし番号

Amazon Transcribe医学会話変換ジョブ たとえば、「one thousand two hundred forty-two」と話すと、「1242」と文字起こしされます。

数値は、次のルールに従って文字起こしされます。

ルール	説明
10 を超える基数を数字に変換します。	• 「Fifty five」> 55 • 「a hundred」 > 100 • 「One thousand and thirty one」 > 1,031 • 「One hundred twenty-three million four hundred fifty six thousand seven hundred eight nine」 > 123,456,789
「million」または「billion」の後に数字が続かない場合、「million」または「billion」が後に続く基数を、単語が後に続く数字に変換します。	• 「one hundred million」 > 100 million • 「one billion」 > 1 billion • 「two point three million」 > 2.3 million
10 を超える序数を数字に変換します。	• 「Forty third」 > 43rd • 「twenty sixth avenue」 > 26th avenue
分数は数値形式に変換	• 「a quarter」> 1/4 • 「three sixteenths」 > 3/16 • 「a half」 > 1/2 • 「a hundredth」 > 1/100
10 より小さい数値は数字に変換 (行に複数ある場合)	• 「three four five」 > 345 • 「My phone number is four two five five five five one two one two」 > 4255551212
少数は「ドット」または「点」で示す	• 「three hundred and three dot five」 > 303.5 • 「three point twenty three」 > 3.23 • 「zero point four」 > 0.4 • 「point three」 > 0.3
数値の後の「percent」という単語をパーセント記号 (%) に変換します。	• 「twenty three percent」 > 23% • 「twenty three point four five percent」 > 23.45%

ルール	説明
数字の後の「dollar」、「US dollar」、「Australian dollar」、「AUD」、「USD」という単語を、数字の前のドル記号 ($) に変換します。	• 「one dollar and fifteen cents」 > $1.15 • 「twenty three USD」 > $23 • 「twenty three Australian dollars」 > $23
「pounds」または「milligrams」を「lbs」または「mg」に変換します。	• 「twenty three pounds」 > 23 lbs • 「forty-five milligrams」 > 45 mg
数字の後の「rupees」、「Indian rupees」、または「INR」という単語を、数字の前のルピー記号 (#) に変換します。	• 「twenty three rupees」 > #23 • 「fifty rupees thirty paise」 > #50.30
時刻は数字に変換	• 「seven a m eastern standard time」 > 7 a.m. eastern standard time • 「twelve thirty p m」 > 12:30 p.m.
2 桁で表した年は 4 桁に変換 20 世紀、21 世紀、および 22 世紀にのみ有効。	• 「nineteen sixty two」 > 1962 • 「the year is twenty twelve」 > the year is 2012 • 「twenty nineteen」 > 2019 • 「twenty one thirty」 > 2130
日付は数字に変換	• 「May fifth twenty twelve」 > May 5th 2012 • 「May five twenty twelve」 > May 5 2012 • 「five May twenty twelve」 > 5 May 2012
数値範囲は単語「to」に変換	• 「twenty three to thirty seven」 > 23 to 37

医療会話の文字起こし

Amazon TranscribeMedical を使用して、バッチ文字起こしジョブまたはリアルタイムストリーミングを使用して、臨床医と患者の間の医療会話を文字起こしすることができます。バッチ文字起こしジョブを使用すると、オーディオファイルを変換することができます。Amazon TranscribeMedical

が可能な限り最高の精度で文字起こし結果を生成するようにするには、文字起こしジョブまたはストリーミングで臨床医の専門医を指定する必要があります。

臨床医と患者の訪問は、以下の医療専門分野で文字起こしすることができます。

- 心臓病学: ストリーミングの文字起こしのみ利用可能
- 神経学: ストリーミング転写でのみ利用可能
- 腫瘍学: ストリーミング転写でのみ利用可能
- プライマリケア: 次のタイプの医療行為が含まれます。
  - 家庭医療
  - 内科
  - 産婦人科 (OB-GYN)
  - 小児科
- 泌尿器学: ストリーミング転写でのみ利用可能

医療用カスタムボキャブラリーを使用すると、文字起こしの精度を向上することができます。「医学用語のカスタム語彙」の詳細については、医療カスタムボキャブラリーによる転写精度の向上 を参照してください。

Amazon TranscribeMedical は、デフォルトでは、信頼度が最も高い文字起こしが返されます。代替文字起こしを返すように設定する場合は、「代替文字起こしの生成」を参照してください。

転写出力で数値と医学的測定値がどのように表示されるかについては、文字起こし番号 と 医療用語と測定値の文字起こし をご覧ください。

トピック

- 医療会話のオーディオファイルの文字起こし
- リアルタイムストリーミングで医療ディクテーションの文字起こし
- 話者パーティショニングの有効化
- マルチチャネルの音声文字起こし

医療会話のオーディオファイルの文字起こし

バッチ文字起こしジョブを使用して、医療会話のオーディオファイルの文字起こしを行います。これを使用して、臨床医と患者の対話を文字起こしすることができま

す。StartMedicalTranscriptionJobAPI またはでバッチトランスクリプションのジョブを開始できますAWS Management Console。

StartMedicalTranscriptionJob API で医療分野の文字起こしジョブを開始する場合、PRIMARYCARE を Specialty パラメータの値として指定します。

AWS Management Console

臨床医と患者の対話の文字起こし (AWS Management Console)

AWS Management Consoleを使用して臨床医と患者の対話を書き起こす場合、文字起こしジョブを作成し、オーディオ入力タイプの会話を選択します。

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[文字起こしジョブ] を選択します。
3. [Create job (ジョブの作成)] を選択します。
4. ジョブ詳細を指定 ページ内の ジョブ設定 で次の指定を行います。
   a. 名前: 文字起こしジョブの名前です。
   b. 音声入力タイプ:会話
5. 残りのフィールドには、Amazon S3オーディオファイルの場所と、文字起こしジョブの出力を保存する場所を指定します。
6. [Next (次へ)] を選択します。
7. [作成] を選択します。

API

バッチ文字起こしジョブ (API) を使用した医療会話の文字起こし

- StartMedicalTranscriptionJob API では、以下のものを指定します。
  a. MedicalTranscriptionJobName の場合、AWS アカウント で一意の名前を指定します。
  b. LanguageCode として、音声ファイルで話されている言語と語彙フィルターの言語に対応する言語コードを指定します。
  c. MediaFileUri オブジェクトの Media パラメータの場合、文字起こしを行うメディアファイルの名前を指定します。

d. Specialty の場合、音声ファイルで話す臨床医の専門分野を PRIMARYCARE に指定します。

e. Type には、CONVERSATION を指定します。

f. にはOutputBucketName、Amazon S3文字起こし結果を保存するバケットを指定します。

以下は、AWS SDK for Python (Boto3) を使用して、PRIMARYCARE を専門とする臨床医と患者の医療に関する会話を文字起こしするリクエストの例です。

```
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)
```

次のコード例は、臨床医と患者の会話の文字起こし結果を示しています。

```
{
    "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

バッチ文字起こしジョブ (AWS CLI) を使用した医療会話の文字起こし

- 以下のコードを実行します。

```
aws transcribe start-medical-transcription-job \
--region us-west-2 \
--cli-input-json file://example-start-command.json
```

以下のコードは、example-start-command.json の内容を示しています。

```
{
      "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"
  }
```

リアルタイムストリーミングで医療ディクテーションの文字起こし

医療関係の会話の音声ストリーミングは、HTTP/2 WebSocket またはプロトコルで文字起こしすることができます。 WebSocket プロトコルを使用してストリーミングを開始する方法については、を参照してください WebSocket ストリームのセットアップ。StartMedicalStreamTranscription API を使用して HTTP/2 ストリーミングを開始します。

次の専門分野でストリーミングを文字起こしを行うことができます。

- 心臓病学
- 神経学
- 腫瘍学

- プライマリケア
- 泌尿器科

各医療専門分野には、多くのタイプの処置と予定が含まれています。したがって、臨床医は、さまざまな種類のメモを指示します。以下の例を参考にして、 WebSocket リクエストのspecialty URI パラメータの値や StartMedicalStreamTranscriptionAPISpecialty のパラメータを指定してください。

- 電気生理学または心エコー検査の相談については、CARDIOLOGY を選択します。
- 医学腫瘍学、外科腫瘍学、または放射線腫瘍学の相談については、ONCOLOGY を選択します。
- 一過性脳虚血発作または脳血管発作のいずれかで脳卒中を起こした患者に診察を提供する医師の場合は、NEUROLOGY を選択します。
- 尿失禁に関する相談については、UROLOGY を選択します。
- 年次検診または緊急ケア訪問の場合は、PRIMARYCARE を選択します。
- ホスピタリストの訪問の場合は、PRIMARYCARE を選択します。
- 出産、卵管結紮法、IUD 挿入、または中絶に関する相談については、PRIMARYCARE を選択します。

AWS Management Console

ストリーミングの医療会話の文字起こし (AWS Management Console)

を使用して臨床医と患者の対話をリアルタイムストリーミングで文字起こしする場合、医学的会話の文字起こしのオプションを選択してストリーミングを開始後、マイクに向かって対話を開始します。AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[リアルタイム文字起こし] を選択します。
3. 会話 を選択します。
4. 医療専門分野 の場合、臨床医の専門分野を選択します。
5. [Start streaming] (ストリーミングの開始) を選択します。
6. マイクに向かって話してください。

HTTP/2 ストリーミングでの医療会話の文字起こし

HTTP/2 リクエストのパラメータのための構文を次に示します。

医療関係の会話の HTTP/2 ストリーミングを文字起こしする場合、StartMedicalStreamTranscription API を使用し、以下を指定します。

- LanguageCode - 言語コードです。有効値は en-US です。
- MediaEncoding - 音声入力に使用されるエンコーディングです。有効な値は、pcm、ogg-opus、flac です。
- Specialty - 医療専門家の専門分野です。
- Type – CONVERSATION

リアルタイムストリーミング内の特定の文字起こし精度を向上させるには、カスタムボキャブラリーを使用します。カスタムボキャブラリーを有効にするには、使用するカスタムボキャブラリーの名前に VocabularyName パラメータの値を設定します。詳細については、「医療カスタムボキャブラリーによる転写精度の向上」を参照してください。

別のスピーカーからの音声にラベルを付ける場合、ShowSpeakerLabel のパラメータ true を設定します。詳細については、「話者パーティショニングの有効化」を参照してください。

医療関係の会話を文字起こしするための HTTP/2 ストリーミングの設定の詳細については、HTTP/2 ストリームの設定 を参照してください。

WebSocket ストリーミングでの医療会話の文字起こし

WebSocket リクエストを使用して、医療会話の文字起こしができます。 WebSocket リクエストを行う場合、署名付き URI を作成します。この URI には、Amazon Transcribeアプリケーションと Medical 間の音声ストリーミングをセットアップするために必要な情報が含まれています。WebSocket リクエストの作成の詳細については、を参照してください WebSocket ストリームのセットアップ。

署名付き URI を作成するには、次のテンプレートを使用します。

```
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
```

リアルタイムストリーミング内の特定の文字起こし精度を向上させるには、カスタムボキャブラリーを使用します。カスタムボキャブラリーを有効にするには、使用するカスタムボキャブラリーの名前に vocabulary-name の値を設定します。詳細については、「医療カスタムボキャブラリーによる転写精度の向上」を参照してください。

別のスピーカーからの音声にラベルを付けるには、show-speaker-label へのパラメータ true を設定します。詳細については、「話者パーティショニングの有効化」を参照してください。

「署名付き URI の作成方法」の詳細については、を参照してください WebSocket ストリームのセットアップ。

話者パーティショニングの有効化

Amazon TranscribeMedical でスピーカーの分割を有効にするには、スピーカーのダイアライゼーションを使用します。これにより、患者が何を言ったのか、臨床医が文字起こし出力で何を言ったかを確認できます。

話者ダイアライゼーションを有効にすると、Amazon Transcribe Medical は各話者の発話に対して各話者の一意の識別子のラベルを付けます。発話 と発言の単位であり、通常は無音で他の発話と区切られます。バッチ文字起こしでは、臨床医からの発話は spk_0 のラベルを受け取ることができ、患者の発話は spk_1 のラベルを受け取ることができます。

ある話者からの発話が別の話者からの発話と重なる場合、Amazon Transcribe Medical は、開始時刻順で文字起こしを指示します。入力オーディオで発話が被っても文字起こし出力では被りません。

バッチ文字起こしジョブ、またはリアルタイムストリーミングを使用して音声ファイルを文字起こしする場合、話者ダイアライゼーションを有効にできます。

トピック

バッチ文字起こしでの話者分割の有効化

バッチ文字起こしジョブで話者パーティション化の有効有効を有効に有効するには、API またはを使用します。パーティション化の有効有効有効を有効に有効できます。有効化有効を有効にするには、StartMedicalTranscriptionJobAPI またはを使用しますAWS Management Console。これにより、臨床医と患者の会話でテキストをパーティション化し、文字起こし出力で誰が何を言ったかを判断できます。

AWS Management Console

AWS Management Console文字起こしジョブで話者ダイアライゼーションを有効にするには、オーディオ識別を有効にしてから話者パーティション化の有効有効を有効にします有効にします有効にします有効にします有効にします有効にします有効にします。

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[文字起こしジョブ] を選択します。
3. [Create job (ジョブの作成)] を選択します。
4. [ジョブの詳細を指定する] ページで、文字起こしジョブに関する情報を入力します。
5. [Next] (次へ) を選択します。
6. [オーディオ識別] を有効にします。
7. オーディオ識別タイプには、「スピーカーパーティショニング」を選択します。
8. 話者の最大数 では、オーディオで話していると思われる話者の最大数を指定します。
9. [作成] を選択します。

API

バッチ文字起こしジョブ (API) を使用して話者のパーティション化の有効有効有効有効有効有効を有効に有効にする有効有効有効を有効にするには有効

- StartMedicalTranscriptionJob API では、以下のものを指定します。

a. MedicalTranscriptionJobName の場合、AWS アカウント で一意の名前を指定します。

b. LanguageCode の場合、音声ファイル内で話されている言語に対応する言語コードです。

c. MediaFileUri オブジェクトの Media パラメータの場合、文字起こしを行うメディアファイルの名前を指定します。

d. Specialty の場合、音声ファイルで話す臨床医の専門分野を指定します。

e. Type には、CONVERSATION を指定します。

f. にはOutputBucketName、Amazon S3文字起こし結果を保存するバケットを指定します。

g. Settings オブジェクトとして、以下を指定します。

   i. ShowSpeakerLabels – true.

   ii. MaxSpeakerLabels - オーディオ内で話していると思われるスピーカーの数を示す 2 ~ 10 の整数です。

次のリクエストでは、AWS SDK for Python (Boto3)を使用して、話者パーティション化の有効化された有効化を有効にしたプライマリケアの臨床医の患者との対話のバッチ文字起こしジョブを開始します。

```
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)
```

次の例のコードは、スピーカーパーティション化の有効化の有効有効化されたパーティション化の有効有効を示しています。パーティション化の有効有効有効を有効にします有効化されたパーティション化の有効有効を示しています。パーティション化の有効有効を有効にします有効

```
{
    "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

プライマリケアを実践している臨床医と患者との間の会話の音声ファイルの文字起こし (AWS CLI)

- 以下のコードを実行します。

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

以下のコードは、example-start-command.json の内容を示しています。

```
{
    "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
        }
}
```

リアルタイムストリーミングで話者パーティショニングを有効にする

リアルタイムストリーミングで話者をパーティション化し、スピーチにラベルを付ける場合、AWS Management Consoleまたはストリーミングリクエストを使用します。話者パーティショニングは、ストリーミングでは 2～5 人の話者で最適に機能します。Amazon TranscribeMedical では、スト

リーミング内の 5 人以上のスピーカーをパーティション化のパーティション化の有効化できますが、その数を超えるとパーティション化の有効性が低下します。

HTTP/2 リクエストを開始する場合、StartMedicalStreamTranscription API を使用します。 WebSocket リクエストを開始する場合、「署名付き URI を使用します。URI には、Amazon Transcribeアプリケーションと Medical 間の双方向通信を設定するために必要な情報が含まれています。

マイクに向かって話されたオーディオで話者パーティショニングを有効にする (AWS Management Console)

を使用して、臨床医と患者の会話のリアルタイムストリーミングを開始したり、マイクにリアルタイムで話されるディクテーションを開始したりできます。AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[リアルタイム文字起こし] を選択します。
3. オーディオ入力タイプ の場合、文字起こしする医療音声の種類を選択します。
4. [その他の設定] で、[スピーカーパーティショニング] を選択します。
5. ストリーミングを開始 を選択して、リアルタイム音声の文字起こしを開始します。
6. マイクに向かって話してください。

HTTP/2 ストリーミングで話者パーティショニングを有効にする

医療関係の会話の HTTP/2 ストリーミング内のパーティション化の有効化有効を有効にするには、StartMedicalStreamTranscriptionAPI を選択し、以下を指定します。

- LanguageCode の場合、ストリーム内の言語に対応する言語コードです。有効値は en-US です。
- MediaSampleHertz の場合、音声のサンプルレートを指定します。
- Specialty の場合、提供者の専門分野を指定します。
- ShowSpeakerLabel – true

医療関係の会話を文字起こしするための HTTP/2 ストリーミングの設定の詳細については、HTTP/2 ストリームの設定 を参照してください。

WebSocket リクエストでスピーカーパーティショニングを有効にする

API WebSocket を使用してストリーミング内の話者を分割する場合、 WebSocket 次の形式を使用してリクエストをスタートするための署名付き URI を作成し、show-speaker-labeltrueと特定します。

```
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
```

次のコードは、ストリーミングリクエストの切り捨てられたレスポンス例を示しています。

```
{
  "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 は、話し手の交代や音声の一時停止など、自然な音声セグメントに基づいて着信音声ストリーミングを中断します。セグメント全体の文字起こしが行われるまで、各レスポンスにさらに多くの文字起こしスピーチが含まれるように、文字起こしは徐々にアプリケーションに返されます。前のコードは、完全に書き起こされたスピーチセグメントの切り捨てられた例です。スピーカーのラベル付けは、完全に書き起こされたセグメントに対してのみ表示されます。

次のリストは、文字起こしのストリーミング の出力におけるオブジェクトとパラメータの組織を示しています。

Transcript

各音声セグメントには、それぞれ独自の Transcript オブジェクトがあります。

Results

各 Transcript オブジェクトには独自の Results オブジェクトがあります。このオブジェクトには isPartial フィールドが含まれます。その値が false の場合、でてくる結果はスピーチセグメント全体に対するものです。

Alternatives

各 Results オブジェクトには Alternatives オブジェクトがあります。

Items

各 Alternatives オブジェクトには独自の Items オブジェクトがあり、それには文字起こし出力の各単語および句読点に関する情報が含まれます。話者のパーティション化の有効化有効化を有効にすると、各単語は完全に文字起こされたスピーチセグメントにSpeakerラベル付けされます。 Amazon TranscribeMedical はこのラベル付けを使用して、ストリーミング内の各スピーカーに一意の整数を割り当てます。Type の値を持つ speaker-change パラメータは、ある人が話すのを停止し、別の人が始めようとしていることを示します。

Transcript

各項目の オブジェクトには、文字起こしされた音声セグメントが Transcript フィールドの値として含まれます。

WebSocket リクエストの詳細については、を参照してください WebSocket ストリームのセットアップ。

マルチチャネルの音声文字起こし

複数のチャネルを持つ音声ファイルまたはストリーミングがある場合は、チャネル識別を使用して、それらの各チャンネルの音声を文字起こしすることができます。 Amazon TranscribeMedical は各チャネルから別々に音声を文字起こします。各チャンネルの個別のトランプスクリプトを単一の文字起こし出力に結合します。

チャネル識別を使用して、音声内の個別のチャネルを特定し、各チャネルの音声の文字起こしを行います。発信者およびエージェントのシナリオなどの状況でこれを有効にします。これを使用して、薬物安全モニタリングを実行するコンタクトセンターからの録音またはストリーミング内のエージェントと発信者を区別します。

バッチ処理とリアルタイムストリーミングの両方でチャネル識別を有効にできます。次のリストは、メソッドごとに有効にする方法を説明しています。

- Batch トランスクリプション —AWS Management Console および StartMedicalTranscriptionJobAPI
- ストリーミング文字起こし — WebSocket ストリーミングと StartMedicalStreamTranscriptionAPI

マルチチャネルの音声ファイルの文字起こし

音声ファイルの文字起こしを行うと、Amazon Transcribe Medical は各チャネルごとにアイテムのリストを返します。アイテムは、文字起こしされた単語または句読点です。各単語には、開始時刻と終了時刻があります。あるチャンネルの人が別のチャンネルで話しかけると、各チャンネルのアイテムの開始時間と終了時刻が重なり、個人が互いに話しかけ合っています。

デフォルトでは、2 つのチャネルで音声ファイルを文字起こしできます。2 つ以上のチャネルを持つファイルを文字起こしする必要がある場合は、クォータの引き上げをリクエストできます。クォータ引き上げのリクエストの詳細については、を参照してくださいAWS のサービス。

バッチ文字起こしジョブでマルチチャネル音声を文字起こしする場合、AWS Management Console または StartMedicalTranscriptionJobAPI を使用します。

AWS Management Console

AWS Management Consoleバッチ文字起こしジョブでチャネル識別を有効にするには、オーディオ識別を有効にしてからチャネル識別を有効にします。チャネル識別は、AWS Management Console の音声識別のサブセットです。

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[文字起こしジョブ] を選択します。
3. [Create job (ジョブの作成)] を選択します。
4. [ジョブの詳細を指定する] ページで、文字起こしジョブに関する情報を入力します。
5. [Next] (次へ) を選択します。
6. [オーディオ識別] を有効にします。
7. オーディオ識別タイプ では、[チャネルの識別] を選択します。
8. [作成] を選択します。

API

マルチチャネルの音声ファイルの文字起こし (API)

- StartMedicalTranscriptionJob API では、以下のものを指定します。

  a. TranscriptionJobName として、AWS アカウント で一意の名前を指定します。

  b. LanguageCode の場合、音声ファイル内で話されている言語に対応する言語コードです。有効値は en-US です。

  c. MediaFileUri オブジェクトの Media パラメータの場合、文字起こしを行うメディアファイルの名前を指定します。

  d. Settings オブジェクトの場合、ChannelIdentification を true にセットします。

以下は、AWS SDK for Python (Boto3) を使ったリクエストの例です。

```
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

バッチ変換ジョブを使用してマルチチャネル音声ファイルの文字起こしをするには (AWS CLI)

- 以下の コードを実行します。

```
aws transcribe start-medical-transcription-job \
--region us-west-2 \
--cli-input-json file://example-start-command.json
```

以下は example-start-command.json のコードです。

```
{
      "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
        }
}
```

次のコードは、2 つのチャネルで会話がある音声ファイルの文字起こし出力を示しています。

```
{
  "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"
            },
            ...
        ]
```

```
    }
}
```

マルチチャネルの音声ストリーミングの文字起こし

HTTP/2 WebSocket またはストリーミングで、別々のチャネルから音声を文字起こしするには、StartMedicalStreamTranscriptionAPIを使用します。

デフォルトでは、2 つのチャネルでストリーミングを文字起こしできます。2 つ以上のチャネルを持つストリーミングを文字起こしする必要がある場合、クォータの引き上げをリクエストできます。クォータ増加の要求の詳細については、「AWS のサービスクォータ」を参照してください。

HTTP/2 ストリーミングでのマルチチャネル音声の文字起こし

HTTP/2 ストリーミング内のマルチチャネル音声を文字起こしする場合、StartMedicalStreamTranscriptionAPI を選択し、以下を指定します。

- `LanguageCode` - 音声の言語コードです。有効値は `en-US` です。
- `MediaEncoding` - 音声のエンコーディングです。有効値は、`ogg-opus`、`flac`、`pcm` です。
- `EnableChannelIdentification` – `true`
- `NumberOfChannels` - ストリーミング音声のチャネル数です。

医療関係の会話を文字起こしするための HTTP/2 ストリーミングの設定の詳細については、HTTP/2 ストリームの設定 を参照してください。

WebSocket ストリーミング内のマルチチャネル音声を文字起こしする文字起こしする文字起こし

WebSocket ストリーミング内のパーティション化のパーティション化の有効化を行う場合、次の形式を使用して署名付き URI を作成し、 WebSocket リクエストを開始します。`enable-channel-identification` を `true` に、`number-of-channels` にストリーミングのチャンネル数を指定します。「署名付き URI には、Amazon TranscribeアプリケーションとMedical 間の双方向通信を設定するために必要な情報が含まれています。

```
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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

WebSocket リクエストの詳細については、を参照してください WebSocket ストリームのセットアップ。

マルチチャネルストリーミング出力

ストリーミングトランスクリプションの出力は、HTTP/2 WebSocket とリクエストと同じです。以下に出力例を示します。

```
{
    "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"
}
```

各音声セグメントには、音声が属するチャネルを示す channelId フラグがあります。

メディカルディクテーションの文字起こし

Amazon TranscribeMedical を使用して、バッチ文字起こしジョブまたはリアルタイムストリームを使用して、臨床医が指示する医療メモを文字起こしジョブまたはリアルタイムストリームで文字起こしすることができます。バッチ文字起こしジョブを使用すると、オーディオファイルを変換することができます。Medical が可能な限り高い精度で文字起こし結果を生成するようにするには、Amazon Transcribe文字起こしジョブまたはストリームで臨床医の専門分野を指定します。

次の専門分野でメディカルディクテーションを文字起こしすることができます。

- 心臓病学 : ストリーミングの文字起こしのみ利用可能
- 神経学: ストリーミング転写でのみ利用可能
- 腫瘍学: ストリーミング転写でのみ利用可能
- プライマリケア: 次のタイプの医療行為が含まれます。
  - 家庭医療
  - 内科
  - 産婦人科 (OB-GYN)
  - 小児科
- 放射線医学 : ストリーミング転写でのみ利用可能
- 泌尿器学 : ストリーミング転写でのみ利用可能

カスタムボキャブラリーを使用すると、文字起こしの精度を向上することができます。「医学用語のカスタム語彙」の詳細については、医療カスタムボキャブラリーによる転写精度の向上 を参照してください。

Amazon TranscribeMedical は、デフォルトでは、信頼度の最も高い文字起こしが返されます。代替文字起こしを返すように設定する場合は、「代替文字起こしの生成」を参照してください。

転写出力で数値と医学的測定値がどのように表示されるかについては、文字起こし番号 と 医療用語と測定値の文字起こし をご覧ください。

トピック

- メディカルディクテーションのオーディオファイルの文字起こし
- リアルタイムストリームでメディカルディクテーションの書き起こし

メディカルディクテーションのオーディオファイルの文字起こし

バッチ文字起こしジョブを使用して、医療会話のオーディオファイルを文字起こしします。これを使用して、臨床医と患者の対話を文字起こしすることができます。StartMedicalTranscriptionJobAPI またはでバッチ文字起こしジョブを開始できますAWS Management Console。

StartMedicalTranscriptionJob API で医療分野の文字起こしジョブを開始する場合、PRIMARYCARE を Specialty パラメータの値として指定します。

AWS Management Console

臨床医と患者の対話の文字起こし (AWS Management Console) の文字起こし

AWS Management Consoleを使用して臨床医と患者の対話を書き起こす場合、文字起こしジョブを作成し、オーディオ入力タイプの会話を選択します。

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[文字起こしジョブ] を選択します。
3. [Create job (ジョブの作成)] を選択します。
4. ジョブ詳細を指定 ページ内の ジョブ設定 で次の指定を行います。
    a. 名前: 文字起こしジョブの名前です。
    b. オーディオ入力タイプ: ディクテーション
5. 残りのフィールドには、Amazon S3オーディオファイルの場所と、文字起こしジョブの出力を保存する場所を指定します。
6. [Next (次へ)] を選択します。
7. [作成] を選択します。

API

バッチ文字起こしジョブ (API) を使用した医療会話の文字起こし

- StartMedicalTranscriptionJob API では、以下のものを指定します。

  a. MedicalTranscriptionJobName の場合、AWS アカウント で一意の名前を指定します。

  b. LanguageCode として、音声ファイルで話されている言語と語彙フィルターの言語に対応する言語コードを指定します。

  c. MediaFileUri オブジェクトの Media パラメータに、文字起こしを行うメディアファイルの名前を指定します。

  d. Specialty の場合、音声ファイルで話す臨床医の専門分野を指定します。

  e. Type には、DICTATION を指定します。

  f. にはOutputBucketName、Amazon S3文字起こし結果を保存するバケットを指定します。

  以下は、AWS SDK for Python (Boto3) を使用して、PRIMARYCARE を専門とする臨床医のメディカルディクテーションを文字起こしするリクエストの例です。

```
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)
```

次のコード例は、メディカルディクテーションの書き起こし結果を示しています。

```
{
    "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

バッチ文字起こしジョブ (でででででで会話の文字起こしジョブAWS CLI) でスピーカーの分割を有効にするには

- 以下のコードを実行します。

```
aws transcribe start-medical-transcription-job \
--region us-west-2 \
--cli-input-json file://example-start-command.json
```

以下のコードは、example-start-command.json の内容を示しています。

```
{
      "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"
}
```

リアルタイムストリームでメディカルディクテーションの書き起こし

WebSocket ストリーミングを使用して、メディカルディクテーションを音声ストリームとして文字起こしします。を使用して、自分または他のユーザーが直接話す音声をマイクに文字起こしすることもできます。AWS Management Console

HTTP/2 WebSocket ストリームまたはストリームでは、次の医療分野のオーディオを文字起こしすることができます。

- 心臓病学
- オンコロジー
- 神経学
- プライマリケア
- 放射線学
- 泌尿器科

各医療専門分野には、多くのタイプの処置と予定が含まれています。したがって、臨床医は、さまざまな種類のメモを指示します。以下の例を参考にして、 WebSocket リクエストのspecialty URI パラメータの値や StartMedicalStreamTranscriptionAPISpecialty のパラメータを指定してください。

- 電気生理学または心エコー検査後のディクテーションについては、CARDIOLOGY を選択します。
- 外科腫瘍学または放射線腫瘍学の処置後のディクテーションについては、ONCOLOGY を選択します。
- 脳炎の診断を示すメモを指示する医師の場合は、NEUROLOGY を選択します。
- 膀胱結石を壊す手順ノートの口述については、UROLOGY を選択します。
- 内科相談後の臨床医ノートの口述については、PRIMARYCARE を選択します。
- CT スキャン、PET スキャン、MRI、または X 線写真の発見を伝える医師の口述については、RADIOLOGY を選択します。
- 婦人科相談後の医師のメモの口述については、PRIMARYCARE を選択します。

リアルタイムストリーム内の特定の文字起こし精度を向上させるには、カスタムボキャブラリーを使用します。カスタムボキャブラリーを有効にするには、使用するカスタムボキャブラリーの名前に vocabulary-name の値を設定します。

マイクに向かって話されたディクテーションを文字起こしするにはAWS Management Console

AWS Management Consoleメディカルディクテーションの音声ストリームを書き起こす場合、メディカルディクテーションを書き起こしを選択してストリーミングを開始後、マイクに向かって会話を開始します。

メディカルディクテーションの音声ストリームの書き起こし (AWS Management Console) の文字起こし

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[リアルタイム文字起こし] を選択します。
3. ディクテーションを選択します。
4. 医療専門分野で、ストリームで話す臨床医の専門分野を選択します。
5. [Start streaming] (ストリーミングの開始) を選択します。
6. マイクに向かって話してください。

HTTP/2 ストリームでの ディクテーションの文字起こし

医療ディクテーションの HTTP/2 ストリームを書き起こすには、`StartMedicalStreamTranscription` API を使用し、以下を指定します。

- `LanguageCode`: 言語コードです。有効値は `en-US` です。
- `MediaEncoding` - 音声入力に使用されるエンコーディングです。有効な値は、`pcm`、`ogg-opus`、`flac` です。
- `Specialty`: 医療専門家の専門分野です。
- `Type` – `DICTATION`

医療ディクテーションを文字起こしするための HTTP/2 ストリームの設定の詳細については、HTTP/2 ストリームの設定 を参照してください。

WebSocketストリーミングリクエストを使用したメディカルディクテーションの文字起こし

WebSocketリクエストを使用してリアルタイムストリームでメディカルディクテーションを書き起こすには、署名付き URI を作成します。この URI には、Amazon Transcribeアプリケーションと Medical の間の音声ストリームをセットアップするために必要な情報が含まれています。WebSocket リクエストの作成の詳細については、を参照してください WebSocket ストリームのセットアップ。

署名付き URI を作成するには、次のテンプレートを使用します。

```
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
```

「署名付き URI の作成方法」の詳細については、を参照してください WebSocket ストリームのセットアップ。

医療カスタムボキャブラリーによる転写精度の向上

Amazon TranscribeMedical の文字起こしの精度を向上させるために、1 つ以上のカスタムボキャブラリーを作成、使用します。カスタムボキャブラリー は、ドメイン固有の単語またはフレーズのコレクションです。このコレクションは、Amazon Transcribe Medical のそれらの単語やフレーズを文字起こしで文字起こしを選択して文字起こしを向上させるのに役立ちます。

Amazon TranscribeMedical を使用する場合、お客様はご自身のデータの完全性について責任を負うものとします。カスタム語彙には、機密情報、個人情報 (PII)、または保護対象の医療情報 (PHI) を入力しないでください。

個別の小さなカスタムボキャブラリーを作成し、それぞれが特定の音声録音を文字起こしする場合、最良の結果が得られます。すべての録音で使用する大きなカスタムボキャブラリーを 1 つ作成した場合よりも、文字起こしの精度が向上します。

デフォルトでは、自分の AWS アカウント で最大 100 個のカスタム語彙を使用できます。カスタムボキャブラリーのサイズは 50 KB を超えることはできません。で使用できるカスタムボキャブラリー数の増加をリクエストする方法についてはAWS アカウント、AWSサービスクォータを参照してください。

カスタムボキャブラリーは、アメリカ英語 (en-US) で利用できます。

トピック

医療カスタムボキャブラリー用のテキストファイルを作成する

カスタムボキャブラリーを作成する場合、UTF-8 形式のテキストファイルを作成します。このファイルでは、4 列のテーブルを作成し、各列がフィールドを指定します。各欄は、ドメイン固有の用語の発音方法またはこれらの用語を文字起こしで表示する方法をAmazon Transcribe Medical に指示します。これらのフィールドを含むテキストファイルは、Amazon S3バケットに保存します。

テキストファイルのフォーマット方法を理解する

医療カスタムボキャブラリーを作成するには、列名をヘッダー行として入力します。ヘッダー行の下にある各列の値を入力します。

表の 4 つの列の名前を以下に示します。

- `Phrase`: 列、値は必要です。
- `IPA`: 列は必須です。値はオプションでもかまいません。
- `SoundsLike`: 列は必須です。値はオプションでもかまいません。
- `DisplayAs`: 列は必須です。値はオプションでもかまいません。

カスタム語彙を作成するときは、次のことを必ず実行してください。

- 各列は 1 つのタブ文字で区切ります。 Amazon Transcribe列をスペースまたは複数のタブ文字で区切ろうとするとエラーメッセージが表示されます。
- 列内の各値の後に末尾にスペースや空白がないことを確認してください。

各列に入力する値が以下であることを確認します。

- 256 文字未満 (ハイフンを含む)

- 文字セットの文字を使用する場合のみ、「Amazon Transcribeメディカル用文字セット」を参照してください。

テーブルの列の値を入力する

次の情報は、テーブルの 4 つの列の値を指定する方法を示しています。

- **Phrase**: 認識する必要がある語句。この列には値を入力する必要があります。

  エントリが句の場合、単語はハイフン (-) で区切ります。たとえば、**cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy** を **cerebral-autosomal-dominant-arteriopathy-with-subcortical-infarcts-and-leukoencephalopathy** として入力します。

  頭字語、または文字が単一の文字とそれに続くドットとして個別に発音される必要があるその他の単語 (例: **D.N.A.** や **S.T.E.M.I.**) を入力します。「STEMIs」などの頭字語の複数形を入力するには、頭字語と「s」をハイフンで区切ります (**S.T.E.M.I-s**)。頭字語には大文字または小文字を使用できます。

  Phrase 列は必須です。入力言語として許可されている文字はいずれも使用できます。使用できる文字については、「Amazon Transcribeメディカル用文字セット」を参照してください。DisplayAs列を指定しない場合、Amazon Transcribe MedicalPhrase は出力ファイル内の列の内容を使用します。

- **IPA** (列は必須、値はオプション) : 単語または句の発音を指定するには、国際音声記号 (IPA) の文字をこの列に使用することができます。IPA 列には、先頭または末尾にスペースを含めることはできません。また、入力の phoneme を区切るには、1 つのスペースを使用する必要があります。たとえば、英語で **acute-respiratory-distress-syndrome** を **ə k j u t # # s p # # ə t # # i d # s t # # s s # n d # o# m** と入力したとします。**A.L.L.** には **e# # l # l** と入力します。

  IPA 列の内容を指定しない場合でも、空白の IPA 列を含める必要があります。IPA 列に値を含めた場合、SoundsLike 列に値を指定することはできません。

  特定の言語で使用できる IPA 文字の一覧については、「Amazon Transcribeメディカル用文字セット」を参照してください。Amazon TranscribeMedical で利用可能な唯一の言語は、アメリカ英語です。

- **SoundsLike** (列は必須、値はオプション): 単語や句を小さい断片に分割し、言語の標準的な正書法を使用して各断片の発音を指定することで、単語の発音方法を模倣することができます。たと

えば、**cerebral-autosomal-dominant-arteriopathy-with-subcortical-infarcts-and-leukoencephalopathy** 句の発音ヒントは **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** のように指定することができます。句 **atrioventricular-nodal-reentrant-tachycardia** のヒントは、**ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia** のようになります。ヒントの各部分はハイフン (-) を使って区切ります。

SoundsLike 列の値を指定しない場合でも、空白の SoundsLike 列を含める必要があります。SoundsLike 列に値を含めた場合、IPA 列に値を指定することはできません。

入力言語として許可されている文字はいずれも使用できます。許可された文字の一覧については、「Amazon Transcribeメディカル用文字セット」を参照してください。

- **DisplayAs** (列は必須、値はオプション) : 出力時の単語または句の外観を定義します。たとえば、単語または句が **cerebral-autosomal-dominant-arteriopathy-with-subcortical-infarcts-and-leukoencephalopathy** の場合は、ハイフンが表示されないように、cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy という形式で表示されるよう指定することができます。また、出力に用語全体ではなく頭字語を表示する場合、DisplayAs を CADASIL として指定することもできます。

  DisplayAs列を指定しない場合、Amazon Transcribe MedicalPhrase は出力の入力ファイル内の列を使用します。

  UTF-8 文字はいずれも、DisplayAs 列で使用することができます。

IPA および DisplayAs 列の値にのみスペースを含むことができます。

カスタム語彙のテキストファイルを作成するには、各単語または各語彙を個別の行のテキストファイルに配置します。列はタブ文字で区切ります。IPA および DisplayAs 列の値にのみスペースを含めます。Amazon TranscribeMedical.txtAmazon S3AWS リージョン を使用してカスタムボキャブラリーを作成する場所と同じバケットに拡張子付きのファイルを保存します。

Windows でテキストファイルを編集する場合、ファイルが CRLF 形式ではなく LF 形式であることを確認してください。そうしないと、カスタム語彙を作成できなくなります。一部のテキスト工ディタでは、検索コマンドと置換コマンドで書式を変更できます。

次の例は、カスタム語彙の作成に使用できるテキストを示しています。これらの例からカスタム語彙を作成するには、例をテキストエディタにコピーし、[TAB] を Tab 文字に置き換えて、保存したテキストファイルを Amazon S3 にアップロードします。

```
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]
```

列は任意の順序で入力できます。次の例は、カスタム語彙入力ファイル の他の有効な構造を 示しています。

```
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
```

読みやすくするために、次の表は、上記の例をより明確に html 形式で示しています。これらは、例の説明のみが目的です。

フレーズ	IPA	SoundsLike	DisplayAs
acute-respiratory-distress-syndrome			acute respiratory distress syndrome
A.L.L。	eɪ ɛ l ɛ l		すべて

フレーズ	IPA	SoundsLike	DisplayAs
atrioventricular-nodal-reentrant-tachycardia		ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia	

フレーズ	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	すべて

DisplayAs	SoundsLike	IPA	フレーズ
acute respiratory distress syndrome			acute-respiratory-distress-syndrome
すべて		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

テキストファイルを使用して医療カスタムボキャブラリーを作成する

カスタムボキャブラリーを作成するには、コレクションに単語やフレーズを含むテキストファイルを用意しておく必要があります。 Amazon TranscribeMedical では、このテキストファイルを使用してカスタムボキャブラリーを作成します。`CreateMedicalVocabulary`APIAmazon Transcribe またはMedical コンソールを使用してカスタムボキャブラリーを作成できます。

AWS Management Console

AWS Management Consoleを使用してカスタムボキャブラリーを作成するには、単語またはフレーズを含むテキストファイルのAmazon S3 URI を指定します。

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[カスタムボキャブラリー] を選択します。
3. 名前 を使用する場合、語彙の設定で、カスタムボキャブラリーの名前を選択します。
4. Amazon S3 でオーディオファイルまたはビデオファイルの場所を指定します。
   - S3のボキャブラリー設定のボキャブラリー入力ファイルの場所で、カスタムボキャブラリーの作成に使用するテキストファイルを識別するAmazon S3 URI を指定します。
   - S3のボキャブラリー入力ファイルの場所については、S3の参照を選択してテキストファイルを参照し、それを選択します。
5. [語彙の作成] を選択します。

カスタムボキャブラリーの処理ステータスがで確認できますAWS Management Console。

API

医学用語のカスタム語彙の作成 (API)

- `StartTranscriptionJob` API では、以下のものを指定します。
  a. `LanguageCode` には、`en-US` を指定します。
  b. の場合`VocabularyFileUri`、Amazon S3カスタムボキャブラリーを定義するために使用するテキストファイルの場所を指定します。
  c. `VocabularyName` の場合、カスタムボキャブラリーの名前を指定します。指定する名前は、AWS アカウント 内で一意でなければなりません。

カスタムボキャブラリーの処理状況を表示する場合、`GetMedicalVocabulary` API を使用します。

以下は、カスタム語彙を作成するために AWS SDK for Python (Boto3) を使用したリクエスト例です。

```
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

バッチ文字起こしジョブ (AWS CLI) で話者の分割を有効にするには

- 以下の コードを実行します。

```
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
```

医療用カスタムボキャブラリーを使用したオーディオファイルの文字起こし

StartMedicalTranscriptionJobまたはを使用して、カスタムボキャブラリーで文字起こしの精度を向上して文字起こしジョブを開始します。AWS Management Console

AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[文字起こしジョブ] を選択します。
3. [Create job (ジョブの作成)] を選択します。
4. [ジョブの詳細を指定する] ページで、文字起こしジョブに関する情報を入力します。
5. [Next] (次へ) を選択します。
6. カスタマイズ で、カスタムボキャブラリー を有効にします。
7. ボキャブラリー選択 で、カスタムボキャブラリーを選択します。
8. [作成] を選択します。

API

バッチ文字起こしジョブ (API) を使用してオーディオファイル内のスピーカーの分割を有効にするには

- StartMedicalTranscriptionJob API では、以下のものを指定します。
  a. MedicalTranscriptionJobName の場合、AWS アカウント で一意の名前を指定します。
  b. LanguageCode として、音声ファイルで話されている言語と語彙フィルターの言語に対応する言語コードを指定します。
  c. MediaFileUri オブジェクトの Media パラメータの場合、文字起こしを行うメディアファイルの名前を指定します。
  d. Specialty の場合、音声ファイルで話す臨床医の専門分野を指定します。
  e. Type の場合、音声ファイルが会話かディクテーションかを指定します。
  f. にはOutputBucketName、Amazon S3文字起こし結果を保存するバケットを指定します。
  g. Settings オブジェクトとして、以下を指定します。
    - VocabularyName: カスタムボキャブラリーの名前です。

次のリクエストでは、AWS SDK for Python (Boto3) を使用して、カスタムボキャブラリーでバッチ文字起こしジョブを開始します。

```
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)
```

カスタムボキャブラリーを使用してリアルタイムストリームを文字起こし

リアルタイムストリームでの文字起こしの精度を向上させるために、HTTP/2 WebSocket またはストリーミングを使用したカスタムボキャブラリーを使用できます。HTTP/2 リクエストを開始する場合、StartMedicalStreamTranscription API を使用します。カスタムボキャブラリーはAWS Management Console、StartMedicalStreamTranscriptionAPI を使用するか、 WebSocket プロトコルを使用してリアルタイムで使用できます。

マイクに話されているディクテーションの書き起こし (AWS Management Console)

を使用してメディカルディクテーションの音声ストリームを書き起こしを選択してストリーミングを開始後、マイクに向かって会話を開始します。AWS Management Console

メディカルディクテーションの音声ストリームの書き起こし (AWS Management Console)

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[リアルタイム文字起こし] を選択します。
3. 医療専門分野 で、ストリームで話す臨床医の専門分野を選択します。
4. 音声入力タイプ の場合、会話 または ディクテーション のいずれかを選択します。
5. 追加設定 の場合、カスタムボキャブラリー を選択します。
   - ボキャブラリー選択 で、カスタムボキャブラリーを選択します。
6. [Start streaming] (ストリーミングの開始) を選択します。
7. マイクに向かって話してください。

HTTP/2 ストリーミングでスピーカーのパーティショニングを有効にする

HTTP/2 リクエストのパラメータのための構文を次に示します。

```
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
```

パラメータの説明のは次のとおりです。

- host:AWS リージョン (前の例の 'us-west-2')AWS リージョン を呼び出しているもので更新します。有効なリストについてはAWS リージョン、「」AWS リージョンと「エンドポイント」を参照してください。
- 権限:これは生成されたフィールドです。署名の作成の詳細については、「署名バージョン 4 で AWS リクエストに署名する」を参照してください。
- x-amz-target: このフィールドは変更しないでください。前の例に示した内容を使用してください。
- x-amz-content-sha256: これは生成されたフィールドです。署名の計算の詳細については、「署名バージョン 4 でAWSリクエストに署名する」を参照してください。
- x-amz-date: 署名が作成された日付と時刻。形式は YYYYMMDDTHHMSSZ です。ここで、YYYYY=年、MM=月、DD=日、HH=時間、MM=分、SS=秒、「T」と「Z」は固定文字です。詳細については、「署名バージョン 4 における日付の処理」を参照してください。
- x-amzn-transcribe-session-id: ストリーミングセッションの名前。
- x-amzn-transcribe-language-code: 入力音声に使用されるエンコード。サポートされている言語および言語固有の機能有効な値のリストについては、`StartMedicalStreamTranscription`またはを参照してください。
- x-amzn-transcribe-media-encoding: 入力音声に使用されるエンコード。有効な値は、`pcm`、`ogg-opus`、`flac` です。
- x-amzn-transcribe-sample-rate: 入力音声のサンプルレート (Hz 単位)。 Amazon Transcribe8,000 Hz ~ 48,000 Hz の範囲をサポートします。電話の音声などの低品質のオーディオは、通常約 8,000 Hz です。高品質のオーディオは、通常 16,000 Hz から 48,000 Hz の範囲です。指定するサンプルレートは、オーディオのサンプルレートと一致する必要があることに注意してください。
- x-amzn-transcribe-vocabulary-name: 文字起こしで使いたい語彙の名前。
- x-amzn-transcribe-specialty: 転記中の医療専門分野。
- x-amzn-transcribe-type: これがディクテーションか会話かを選択します。
- x-amzn-transcribe-show-speaker-label: ダイアライゼーションを有効にするには、この値をにする必要がありま`true`す。
- content-type: このフィールドは変更しないでください。前の例で示した内容を使用してください。

WebSocketリクエストでスピーカーパーティショニングを有効にする

API WebSocket を使用してストリーム内の話者を分割する場合、 WebSocket `vocabulary-name`次の形式を使用してリクエストをスタートするための署名付き URI を作成し、

```
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メディカル用文字セット

Amazon TranscribeMedical のカスタムボキャブラリーを使用する場合、次の文字セットを使用します。

英語の文字セット

英語のカスタム語彙の場合、Phrase 列および SoundsLike 列に次の文字を使用できます。

- a ~ z
- A ~ Z
- ' (apostrophe)
- - (ハイフン)
- . (ピリオド)

語彙入力ファイルの IPA 列には、国際音声記号 (IPA) 文字を使用できます。

[Character] (文字)	コード	[Character] (文字)	コード
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
l	006C	ɡ	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

[Character] (文字)	コード	[Character] (文字)	コード
u	0075	ʧ	02A7
v	0076	θ	03B8

トランスクリプションにおける個人の健康情報 (PHI) の特定

個人の健康情報の識別を使用して、トランスクリプションの結果に個人の健康情報 (PHI) にラベル付けします。ラベル付けを確認することで、患者の識別に使用できる PHI を見つけることができます。

リアルタイムストリーミングまたはバッチトランスクリプションジョブを使用して PHI を識別できます。

独自の後処理を使用して、トランスクリプション出力で識別された PHI を編集できます。

個人の健康情報の識別を使用して、次のタイプの PHI を識別します。

- 個人の PHI:
  - 名前 — 氏名または姓とイニシャル
  - [Gender] (性別)
  - 年齢
  - [電話番号]
  - 患者に直接関係する日付 (年を含まない)
  - E メールアドレス
- 地理的 PHI:
  - 物理アドレス
  - ZIP コード
  - 医療センターまたは診療所の名前
- PHI アカウント :
  - ファックス番号
  - 社会保障番号 (SSN)
  - 健康保険受取人番号
  - 口座番号

  - 証明書/免許証番号
- 車両 PHI:
  - 車両識別番号 (VIN)
  - ナンバープレート番号
- その他の PHI:
  - ウェブユニフォームリソースの場所 (URL)
  - インターネットプロトコル (IP) アドレス番号

Amazon TranscribeMedical は、1996 年のHealth 保険の相互運用性と説明責任に関する法令 (HIPAA) の対象となるサービスです。詳細については、「Amazon Transcribe 医療」を参照してください。オーディオファイル内の PHI の識別については、「オーディオファイル内の PHI の識別」を参照してください。ストリーミング内の PHI の識別については、「リアルタイムストリーミングでの PHI の識別 」を参照してください。

トピック

- オーディオファイル内の PHI の識別
- リアルタイムストリーミングでの PHI の識別

オーディオファイル内の PHI の識別

バッチトランスクリプションジョブを使用して、オーディオファイルを書き起こし、その中 の個人の健康情報 (PHI) を特定します。個人のHealth 情報 (PHI を識別する) を有効にすると、Amazon Transcribe Medical はトランスクリプション結果で識別した PHI ににラベル付けします。Amazon TranscribeMedical が識別できる PHI の詳細については、「」を参照してくださいトランスクリプションにおける個人の健康情報 (PHI) の特定。

`StartMedicalTranscriptionJob`API またはのいずれかを使用して、バッチ文字起こしジョブを開始することができますAWS Management Console。

AWS Management Console

AWS Management Consoleを使用して臨床医と患者の対話を書き起こす場合、文字起こしジョブを作成し、オーディオ入力タイプの会話を選択します。

オーディオファイルを書き起こし、そのAWS Management Console PHI を識別するには、

1. AWS Management Consoleにサインインします。

2. ナビゲーションペインの [Amazon Transcribe医療] で、[文字起こしジョブ] を選択します。
3. [Create job (ジョブの作成)] を選択します。
4. [ジョブ詳細を指定] ページ内の [ジョブ設定] で次の指定を行います。
   a. 名前 - お客様の AWS アカウント に固有のトランスクリプションジョブの名前。
   b. オーディオ入力タイプ – [会話] または [ディクテーション]。
5. 残りのフィールドには、Amazon S3オーディオファイルの場所と、文字起こしジョブの出力を保存する場所を指定します。
6. [Next] (次へ) を選択します。
7. [オーディオ設定]で、[PHI 識別] を選択します。
8. [作成] を選択します。

API

バッチトランスクリプションジョブ (API) を使用してオーディオファイルを書き起こし、その PHI を識別するには、

- StartMedicalTranscriptionJob API では、以下のものを指定します。
   a. にはMedicalTranscriptionJobName、自分だけの名前を指定してくださいAWS アカウント。
   b. LanguageCode の場合、オーディオファイルで話されている言語に対応する言語コードを指定します。
   c. MediaFileUri パラメータがあるMedia オブジェクトの場合、文字起こしを行うオーディオファイルの名前を指定します。
   d. Specialty の場合、音声ファイルで話す臨床医の専門分野を PRIMARYCARE として指定します。
   e. Typeを使用する場合で、CONVERSATIONと DICTATIONのいずれかを指定します。
   f. の場合OutputBucketName、Amazon S3トランスクリプション結果を保存するバケットを指定します。

   以下は、リクエストの例です。オーディオファイルを書き起こし、患者の PHI を識別するために AWS SDK for Python (Boto3) を使用します。

```
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)
```

次のコード例は、患者 PHI を識別した場合のトランスクリプションの結果を示しています。

```
{
    "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

バッチトランスクリプションジョブ (AWS CLI) を使用してオーディオファイルを書き起こし、その PHI を識別するには

- 以下の コードを実行します。

```
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
```

リアルタイムストリーミングでの PHI の識別

HTTP/2 WebSocket またはストリーミングで個人のHealth 情報 (PHI を識別できます) を識別できます。PHI 識別を有効にすると、Amazon Transcribe Medical はトランスクリプション結果で識別した PHI を識別します。Amazon TranscribeMedical が識別できる PHI の詳細については、「」を参照してくださいトランスクリプションにおける個人の健康情報 (PHI) の特定。

マイクで話されるディクテーションで PHI を識別する

を使用して、マイクで拾った音声を書き起こし、PHIAWS Management Console を識別するには、オーディオ入力タイプとして [ディクテーション] を選択し、ストリーミングを開始し、コンピュータのマイクで話し始めます。

ディクテーション内の PHI を識別するには、AWS Management Console

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインで、[リアルタイム文字起こし] を選択します。
3. [オーディオ入力タイプ] で、[ディクテーション]を選択します。
4. [その他の設定] で [PHI ID] を選択します。
5. [ストリーミングの開始] を選択し、マイクに向かって話してください。
6. [ストリーミングの停止] を選択すると、ディクテーションが終了します。

HTTP/2 ストリーミング内の PHI の識別

PHI 識別をアクティブにして HTTP/2 ストリーミングを開始するには、StartMedicalStreamTranscription API を使用し、以下を指定してください。

- LanguageCode の場合、ストリーミング内の言語に対応する言語コードを指定してください。米国英語の場合は、[en-US] を指定してください。
- MediaSampleHertz の場合、オーディオのサンプルレートを指定します。
- content-identification-type には、PHI を指定します。

WebSocket ストリーミング内の PHI を識別する

PHI WebSocket 識別をアクティブにしてストリーミングを開始するには、次の形式を使用して署名付き URL を作成します。

```
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
```

パラメータの定義は API リファレンスにあります。すべてのAWS API オペレーションに共通するパラメータは、「共通パラメータ」セクションに記載されています。

代替文字起こしの生成

Amazon TranscribeMedical を使用すると、最も信頼性の高い文字起こしを取得できます。また、より低い信頼性で追加の文字起こしを返すようにAmazon Transcribe Medical を設定することもできます。

代替文字起こしを使用して、変換されたオーディオのさまざまな解釈を確認します。たとえば、ユーザーが書き起こしをレビューできるアプリケーションでは、選択できる代替文字起こしを提示できます。

AWS Management Consoleまたは `StartMedicalTranscriptionJob`API を使用して代替トランスクリプションを生成できます。

AWS Management Console

AWS Management Consoleを使用して代替文字起こしを生成するには、ジョブを設定する際、代替結果を有効にします。

1. AWS Management Consoleにサインインします。
2. ナビゲーションペインのAmazon Transcribe Medical で、[文字起こしジョブ] を選択します。
3. [Create job (ジョブの作成)] を選択します。
4. [ジョブの詳細を指定する] ページで、文字起こしジョブに関する情報を入力します。
5. [Next] (次へ) を選択します。
6. [代替結果] を有効にする。
7. [代替の最大数] には、2 から 10 までの整数値を入力して、出力に必要な代替文字起こしの最大数を指定します。

8. [作成] を選択します。

API

バッチ文字起こしジョブ (API) を使用してオーディオファイル内のテキストを分割するには

- StartMedicalTranscriptionJob API では、以下のものを指定します。

  a. MedicalTranscriptionJobName の場合、AWS アカウント で一意の名前を指定します。

  b. LanguageCode として、音声ファイルで話されている言語と語彙フィルターの言語に対応する言語コードを指定します。

  c. MediaFileUriMediaオブジェクトのパラメータに、文字起こしを行うメディアファイルの場所を指定します。

  d. Specialty の場合、音声ファイルで話す臨床医の専門分野を指定します。

  e. Type の場合、医療会話を文字起こしするか、口述を筆記するかを指定します。

  f. にはOutputBucketName、Amazon S3文字起こし結果を保存するバケットを指定します。

  g. Settings オブジェクトとして、以下を指定します。

     i. ShowAlternatives – true.

     ii. MaxAlternatives - 2 から 10 までの整数値で、文字起こし出力に必要な代替文字起こしの数を示します。

次のリクエストでは、AWS SDK for Python (Boto3) を使用して、最大 2 つの代替文字起こしを生成する文字起こしジョブを開始します。

```
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

プライマリケア臨床医と患者との間の会話の音声ファイルを転記するには、AWS CLI

- 以下のコードを実行します。

```
aws transcribe start-transcription-job \
--cli-input-json file://filepath/example-start-command.json
```

以下のコードは、example-start-command.json の内容を示しています。

```
{
      "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医療およびインターフェイス VPC エンドポイント (AWS PrivateLink)

VPC との間にプライベート接続を確立できますAmazon Transcribe作成による医療インターフェイス VPC エンドポイント。インターフェイスエンドポイントは以下から給電されます。AWS PrivateLink、プライベートアクセスを可能にするテクノロジーAmazon Transcribeインターネットゲートウェイ、NAT デバイス、VPN 接続、またはAWS Direct Connect接続。VPC 内のインスタンスとの通信にパブリック IP アドレスは必要ありませんAmazon Transcribe医療API。VPC と VPC 間のトラフィックAmazon Transcribe医療はAmazonのネットワークを離れません。

各インターフェースエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。

詳細については、を参照してください。インターフェイス VPC エンドポイント (AWS PrivateLink)にAmazon VPCユーザーガイド。

に関する考慮事項Amazon Transcribe医療用 VPC エンドポイント

のインターフェイス VPC エンドポイントを設定する前にAmazon Transcribe医療、必ず確認してくださいインターフェイスエンドポイントのプロパティと制限事項にAmazon VPCユーザーガイド。

Amazon TranscribeMedical では、VPC からすべての API アクションを呼び出すことができます。

のインターフェイス VPC エンドポイントの作成Amazon Transcribe医療

の VPC エンドポイントを作成できます。Amazon Transcribeどちらかを利用した医療サービスAWS Management ConsoleまたはAWS CLI。詳細については、を参照してください。インターフェースエンドポイントの作成にAmazon VPCユーザーガイド。

でのバッチ転記用Amazon Transcribe医療関連の方は、次のサービス名を使用して VPC エンドポイントを作成してください。

- com.amazonaws。*####-2*. 文字起こし

でのストリーミングトランスクリプション用Amazon Transcribe医療関連の方は、次のサービス名を使用して VPC エンドポイントを作成してください。

- com.amazonaws。*####-2*. トランスクリプションストリーミング

エンドポイントのプライベート DNS を有効にすると、以下の API リクエストを行うことができますAmazon Transcribeメディカルはデフォルトの DNS 名をAWS リージョン、例えば、`transcribestreaming.us-east-2.amazonaws.com`。

詳細については、を参照してください。インターフェースエンドポイントからサービスにアクセスするにAmazon VPCユーザーガイド。

の VPC エンドポイントポリシーの作成Amazon Transcribeメディカルストリーミング

アクセスを制御するエンドポイントポリシーを VPC エンドポイントにアタッチできますAmazon Transcribe医療。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、「Amazon VPC ユーザーガイドの「VPC エンドポイントでサービスへのアクセスを制御する」を参照してください。

例:の VPC エンドポイントポリシーAmazon Transcribe医療ストリーミングの文字起こしアクション

以下は、でのストリーミングトランスクリプションのエンドポイントポリシーの例ですAmazon Transcribe医療。エンドポイントにアタッチすると、このポリシーによりリストされたエンドポイントへのアクセスが許可されますAmazon Transcribeすべてのリソースですべての校長に医療処置を行います。

```
{
```

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartMedicalStreamTranscription",
         ],
         "Resource":"*"
      }
   ]
}
```

例:の VPC エンドポイントポリシーAmazon Transcribe医療バッチ転写アクション

以下は、でのバッチ転記に関するエンドポイントポリシーの例ですAmazon Transcribe医療。エンドポイントにアタッチすると、このポリシーによりリストされたエンドポイントへのアクセスが許可されますAmazon Transcribeすべてのリソースですべての校長に医療処置を行います。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartMedicalTranscriptionJob"
         ],
         "Resource":"*"
      }
   ]
}
```

共有サブネット

共有されているサブネットの VPC エンドポイントを作成、説明、変更、削除することはできません。ただし、VPC エンドポイントを使用することはできます。VPC 共有の詳細については、を参照してください。VPC を他のアカウントと共有するにAmazon Virtual Private Cloudガイド。

AWS HealthScribe

AWS HealthScribe は HIPAA に対応した新しい機械学習 (ML) 機能です。音声認識と生成系 AI を組み合わせて、患者と臨床医との会話を文字起こしし、レビューしやすい臨床メモを作成します。AWSHealthScribe は、医療ソフトウェアベンダーが文書作成する負担を軽減し、診察エクスペリエンスを向上させる臨床アプリケーションを構築できるよう支援します。このサービスは、豊富な会話トランスクリプトの提供、スピーカーの役割の特定、会話の分類、医学用語の抽出、予備的な臨床メモの作成を自動的に行います。AWSHealthScribe はこれらの機能を組み合わせることで、個別の AI サービスを統合して最適化する必要がなくなり、導入を迅速に行うことができます。

一般的なユースケース

- 文書作成する時間の短縮 — 臨床医は、AI が生成した臨床メモを使用して臨床文書を迅速に完成させることができます。臨床メモはアプリケーション内で簡単に表示、調整、確定できます。
- 医療従事者が効率よく仕事できる — AI が生成したトランスクリプトや臨床メモ、診察音声を医療従事者に提供することで、文書作成までの時間を短縮できます。
- 患者の診察を効率的に要約する — ユーザーがアプリケーション内で会話の要点をすばやく思い出せるようなエクスペリエンスを作ります。

> **Important**
>
> AWS HealthScribe が生成する結果は確率的であり、音質、バックグラウンドノイズ、話者の明瞭さ、医学用語の複雑さ、状況に応じた言語のニュアンス、機械学習や生成 AI の性質など、さまざまな要因により常に正確であるとは限りません。AWSHealthScribe は、臨床医や医療従事者の補助的な役割を果たすように設計されています。AWSHealthScribe の出力は、訓練を受けた医療専門家による正確性の確認と健全な医療判断の適用を受けた後で、電子カルテなどの一部として、患者の医療シナリオでのみ使用する必要があります。AWSHealthScribe の出力は、専門的な医療アドバイス、診断や治療に代わるものではなく、疾患や健康状態の治療、軽減、予防、診断を目的としたものではありません。

AWS HealthScribe は、責任共有モデルの下で運営されています。AWS は AWS HealthScribe を実行するインフラストラクチャを保護する責任があり、お客様がデータの管理の責任を負います。詳細については、「責任共有モデル」を参照してください。

AWS HealthScribe は米国東部 (バージニア北部) リージョンで利用できます。

サービスは米国英語 (en-US) で利用できます。最良の結果を得るには、FLAC または PCM 16 ビットエンコーディング の WAV などの可逆音声形式を使用します。AWSHealthScribe は 16,000 Hz 以上のサンプリングレートをサポートしています。

AWS HealthScribe は現在、一般医療と整形外科の専門分野をサポートしています。

AWS HealthScribe ジョブでは、医療の相談内容を分析して 2 つの JSON 出力ファイル (トランスクリプトファイルと臨床ドキュメント) を生成します。

トランスクリプトファイルでは、単語レベルのタイムスタンプ付きの標準的なターンバイターンの文字起こし出力に加えて、AWS HealthScribe は次の機能を提供します。

- 参加者ロールの検出により、会話トランスクリプトで患者と臨床医を区別できます。
- トランスクリプトセクショニングは、トランスクリプトの会話を、雑談、主観、客観などの臨床的な関連性に基づいて分類します。これを使用して、トランスクリプトの特定の部分を表示できます。
- 臨床エンティティには、会話の中で言及された薬、病状、治療法などの構造化された情報が含まれます。

臨床ドキュメントファイルにより、AWS HealthScribe は以下を提供します。

- 概要。これには、主訴、現在の病歴、体組織の確認、既往歴、評価、計画など、臨床ドキュメントの主要なセクションに関する要約されたメモが含まれます。
- エビデンスリンク。これにより、AI が生成した概要に使用されているすべての文章が、元の診察トランスクリプトにリンクされるため、ユーザーはアプリケーションで要約の正確性を検証しやすくなります。

AWS HealthScribe 固有の API オペレーション

- StartMedicalScribeJob
- ListMedicalScribeJobs
- GetMedicalScribeJob
- DeleteMedicalScribeJob

AWS HealthScribe リクエストの例を確認するには、「AWSHealthScribe ジョブの開始」を参照してください。

トランスクリプトファイル

トランスクリプトファイルには、会話の内容がターンバイターン形式で表示されます。

さらに、会話のターンごとに以下のインサイトが得られます。

- 参加者ロール — 各参加者には、臨床医または患者のどちらかのラベルが付けられます。会話の各カテゴリに複数の参加者がいる場合、各参加者には番号が割り当てられま 例えば、CLINICIAN_1、CLINICIAN_2、および PATIENT_1、PATIENT_2 です。
- セクション — ダイアログの各ターンは、特定された内容に基づいて 4 つのセクションのうちの 1 つに割り当てられます。
  - 主観的 — 患者が自身の健康上の懸念について提供する情報。
  - 客観的 — 臨床医が身体検査、臨床検査、画像検査、または診断検査を通じて観察した情報。
  - 評価と計画 — 医師の評価と治療計画に関する情報。
  - フロー管理を表示 — 世間話や推移に関する情報。
- インサイト — 会話に含まれる臨床関連のエンティティ (ClinicalEntity) を抽出します。AWSHealthScribe は Amazon Comprehend Medical がサポートするすべての臨床エンティティを検出します。

出力情報の詳細については、「トランスクリプト出力の例」を参照してください。

臨床ドキュメントファイル

ドキュメントのインサイトファイルは、臨床ドキュメントの以下の主要セクションに関する要約を示します。

セクション	説明
主訴	患者の来院理由の簡単な説明。
現在の病歴	重症度、発症、発症時期、現在の治療法、患部など、患者の病気に関する情報を記載したメモ。
体組織の確認	さまざまな体組織について患者が報告した症状の評価。

セクション	説明
既往歴	患者の以前の病状、手術、治療の詳細。
評価	臨床医による患者の健康評価に関する情報を記載したメモ。
計画	治療、生活習慣の調整、予約などを記載したメモ。

Summary に記載されているすべての文章には、元の診察トランスクリプトへの参照が含まれています。これにより、ユーザーはアプリケーションの要約が正確であることを簡単に確認できます。AI が生成したインサイトにトレーサビリティと透明性を持たせることは、責任ある AI 原則 (説明可能性など) に合致しています。これらの参考資料を要約メモとともに臨床医や医療関係者に提供することで、信頼を育み、臨床現場での AI の安全な使用を促すことができます。

Summary のすべての文章には EvidenceLinks が付属しており、要約されたトランスクリプト内の関連する会話を SegmentId に提供します。

出力情報の詳細については、「臨床ドキュメント出力の例」を参照してください。

AWS HealthScribe ジョブを始める

AWS HealthScribe ジョブは、AWS CLI または AWS SDK を使用して開始できます。例については以下を参照してください。

AWS CLI

この例では start-medical-scribe-job コマンドを使用しています。詳細については、「StartMedicalScribeJob」を参照してください。

```
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
```

start-medical-scribe-job コマンド、および追加設定を含むリクエストボディを使用した別の例になります。

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

ファイル my-first-medical-scribe-job.json には、次のリクエストボディが含まれています。

```
{
  "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)

次の例では、AWS SDK for Python (Boto3) を使用して start_medical_scribe_job リクエストを行います。詳細については、「StartMedicalScribeJob」を参照してください。

```
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**
>
> 現在、AWS マネジメントコンソールは AWS HealthScribe ジョブをサポートしていません。

出力例

StartMedicalScribeJob リクエストでは、トランスクリプトに加えて、別の臨床ドキュメントが生成されます。どちらのファイルも JSON 形式で、リクエストで指定した出力場所に保存されます。各出力タイプの例を以下に示します。

トランスクリプトの出力例

AWS HealthScribe のトランスクリプトファイル (StartMedicalScribeJob リクエストによる) の形式は次のとおりです。


```
{
  "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"
      }
    ]
  }
}
```

start-medical-scribe-job コマンド、および追加設定を含むリクエストボディを使用した別の例になります。

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

ファイル my-first-medical-scribe-job.json には、次のリクエストボディが含まれています。

```
{
  "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"
    }
  ]
}
```

臨床ドキュメント出力の例

ドキュメントのインサイトファイル (StartMedicalScribeJob リクエストによる) の形式は次のとおりです。

```
{
  "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 の保管中のデータ暗号化

AWS HealthScribe は、デフォルトで暗号化を提供し、Amazon S3 マネージドキーを使用して保管中のお客様の機密データを保護します。

- Amazon S3 マネージドキー (SSE-S3) — AWS HealthScribe はデフォルトで Amazon S3 マネージドキーを使用し、中間ファイルを自動的に暗号化します。Amazon S3 マネージドキーは表示、管理、使用することはできず、その使用を監査することもできません。ただし、データを暗号化するキーを保護するためにアクションを実施したり、プログラムを変更したりする必要はありません。詳細については、「SSE-S3」を参照してください。

保管中のデータをデフォルトで暗号化することで、機密データの保護におけるオーバーヘッドと複雑な作業を減らすのに役立ちます。同時に、セキュリティを重視したアプリケーションを構築して、暗号化のコンプライアンスと規制の厳格な要件を満たすことができます。

この暗号化レイヤーを無効にしたり、別の暗号化タイプを選択したりすることはできませんが、AWS HealthScribe でジョブを作成する際にカスタマーマネージドキーを選択することで、既存の Amazon S3 マネージドキーに重ねて 2 番目の暗号化レイヤーを追加できます。

- カスタマーマネージドキー - AWS HealthScribe では、ユーザーが作成、所有、管理する対称カスタマーマネージドキーを使用して、AWS が所有する既存の暗号化に重ねて別の暗号化レイヤーを追加できます。この暗号化層はユーザーが完全に制御できるため、次のようなタスクを実行できます。
  - キーポリシーの策定と維持
  - IAM ポリシーとグラントの策定と維持
  - キーポリシーの有効化と無効化
  - 暗号化素材のローテーション
  - タグの追加
  - キーエイリアスの作成
  - キー削除のスケジュール設定

詳細については、「AWS Key Management Service デベロッパーガイド」の「カスタマーマネージドキー」を参照してください。

> **Note**
>
> AWS HealthScribe では、個人を特定できるデータを無料で保護するために、AWS が所有するキーを使用して保管中の暗号化を自動的に有効にします。ただし、カスタマーマネージドキーの使用には AWS KMS 料金が適用されます。料金の詳細については、「AWS Key Management Service の料金」を参照してください。
> AWS KMS の詳細については、「AWS Key Management Service とは」を参照してください。

カスタマーマネージド キーを作成する

対称カスタマーマネージドキーを作成するには、AWS Management Console または AWS KMS API を使用します。対称カスタマーマネージドキーを作成するには、「AWS Key Management Service デベロッパーガイド」の「対称カスタマーマネージドキーの作成」の手順に従います。

キーポリシーは、カスタマーマネージドキーへのアクセスを制御します。すべてのカスタマーマネージドキーには、キーポリシーが 1 つだけ必要です。このポリシーには、そのキーを使用できるユーザーとその使用方法を決定するステートメントが含まれています。カスタマーマネージドキーを作成する際に、キーポリシーを指定することができます。詳細については、「AWS Key Management Service デベロッパーガイド」の「カスタマーマネージドキーへのアクセスの管理」を参照してください。

StartMedicalScribeJob リクエストで DataAccessRoleArn として指定した IAM ロールと同じアカウントのキーを使用している場合は、キーポリシーを更新する必要はありません。別のアカウントのカスタマーマネージドキーを DataAccessRole として使用するには、キーポリシーの DataAccessRoleArn を信頼して以下のアクションを実行する必要があります。

- kms:Encrypt — カスタマーマネージドキーを使用した暗号化を許可します。
- kms:Decrypt — カスタマーマネージドキーを使用した復号を許可します。
- kms:DescribeKey — カスタマーマネージドキーの詳細を提供し、AWS HealthScribe でキーを検証できるようにします。

次に示すのは、カスタマーマネージドキーを使用するためのクロスアカウント許可を IAM ロールに付与するために追加できるポリシーステートメントの例です。

```
"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" : "*"
  }
]
```

カスタマーマネージドキーと DataAccessRole が同じアカウントにあるか異なるアカウントにあるかにかかわらず、DataAccessRole にはカスタマーマネージドキーを使用して上記のアクションを実行するアクセス許可が必要です。次に示すのは、DataAccessRole に追加できるポリシーステートメントの例です。

```
"Statement" : [
  {
    "Sid": "Allow role to perform AWS KMS actions for customer managed key",
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }
]
```

ポリシーでのアクセス許可の指定に関する詳細については、「AWS Key Management Service デベロッパーガイド」を参照してください。キーアクセスのトラブルシューティングに関する詳細については、「AWS Key Management Service デベロッパーガイド」を参照してください。

AWS HealthScribe でのカスタマーマネージドキーの指定

カスタマーマネージドキーを StartMedicalScribeJob リクエストの別の暗号化レイヤーとして指定できます。StartMedicalScribeJob リクエストの作成時に、リクエストに OutputEncryptionKMSKeyId フィールドを含めることで、カスタマーマネージドキーを指定できます。

AWS KMS の暗号化コンテキスト

AWS KMS 暗号化コンテキストは、プレーンテキスト、非シークレットキー: 値ペアのマップです。このマップは、暗号化コンテキストペアと呼ばれる追加の認証データを表し、データのセキュリティレイヤーを追加できます。AWSHealthScribe では、お客様指定の Amazon S3 バケットへの AWS HealthScribe 出力を暗号化するための対称暗号化キーが必要です。詳細については、「AWS KMS の非対称キー」を参照してください。

暗号化コンテキストペアを作成するときは、機密情報を含めないでください。暗号化コンテキストはシークレットではなく、CloudTrail ログ内でプレーンテキストで見ることができます (したがって、暗号化操作を特定し分類するために使用できます)。暗号化コンテキストのキーと値には、アンダースコア (_)、ダッシュ (-)、スラッシュ (/、\) 、コロン (:) などの特殊文字を含めることができます。

> **Tip**
>
> 暗号化コンテキストペアの値を暗号化されるデータに関連付けると便利です。必須ではありませんが、ファイル名、ヘッダー値、暗号化されていないデータベースフィールドなど、暗号化されたコンテンツに関連する機密性のないメタデータを使用することをお勧めします。
> API で出力暗号化を使用するには、StartMedicalScribeJob オペレーションで KMSEncryptionContext パラメータを設定します。出力暗号化オペレーションに暗号化コンテキストを提供するには、OutputEncryptionKMSKeyId パラメータで対称 AWS KMS キー ID を参照する必要があります。
> IAM ポリシーで AWS KMS 条件キーを使用すると、暗号化オペレーションのリクエストで使用された暗号化コンテキストに基づいて、対称暗号化 AWS KMS キーへのアクセスを制御できます。暗号化コンテキストポリシーの例については、「AWS KMS 暗号化コンテキストポリシー」を参照してください。
> 暗号化コンテキストの使用は任意ですが、推奨されています。詳細については、「暗号化コンテキスト」を参照してください。

のドキュメント履歴 Amazon Transcribe

- ドキュメントの最新更新日: 2023 年 11 月 13 日

次の表に、 の各リリースにおける重要な変更点を示します Amazon Transcribe。このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

変更	説明	日付
機能更新	ダイアライゼーションの最大発話者数を 10 ではなく 30 に更新します。	2024 年 5 月 10 日
セクション更新	生成通話の要約を更新し、エラー出力の詳細を追加しました。	2024 年 4 月 30 日
セクション更新	カスタム語彙列の更新 - IPA および SoundsLike。	2024 年 4 月 30 日
機能更新	Amazon Transcribe Call Analytics で、通話の生成要約がサポートされるようになりました。	2023 年 11 月 29 日
セクション更新	PII のマスキングと言語識別の新しい出力形式を更新しました。	2023 年 11 月 13 日
機能更新	ダイアライゼーションをチャネル識別と組み合わせることができるようになりました。	2023 年 3 月 6 日
機能更新	チャネル識別をダイアライゼーションと組み合わせることができるようになりました。	2023 年 3 月 6 日

セクション更新	IAM ベストプラクティスが更新されました。	2023 年 2 月 13 日
新しい言語	Amazon Transcribe がベトナム語とスウェーデン語をサポートするようになりました。	2022 年 12 月 6 日
新機能	Amazon Transcribe がリアルタイムコール分析をサポートするようになりました。	2022 年 11 月 28 日
機能更新	ストリーミングリダクションと識別がヒンディー語とタイ語で利用できるようになりました。	2022 年 11 月 11 日
セクション更新	新しい PII カテゴリがストリーミングリダクションおよび識別できるようになりました。	2022 年 9 月 14 日
セクション更新	カスタム言語モデルのセクションが改訂されました。	2022 年 6 月 18 日
セクション更新	バッチ言語識別で、音声ファイルごとに複数の言語を識別できるようになりました。	2022 年 5 月 31 日
ガイドの更新	Amazon Transcribe API リファレンスはスタンドアロンガイドになりました。	2022 年 4 月 1 日
新しい章	Amazon Transcribe、 Amazon Transcribe Medical、および Amazon Transcribe Call Analytics の新しい比較テーブルが含まれています。	2022 年 3 月 21 日

新しい章	新しい SDK コード例の章が含まれています。	2022 年 3 月 21 日
機能更新	コール分析では、コールサマリーが提供されるようになりました。	2022 年 3 月 21 日
章の更新	入門章では、 Amazon Transcribe ユースケースを紹介するようになりました。	2022 年 3 月 21 日
章の更新	「はじめに」の章がメソッド固有のものになるように更新されました。	2022 年 3 月 21 日
章の更新	ストリーミングの章が更新され、再構成されました。	2022 年 3 月 21 日
機能更新	言語識別では、ストリーミング文字起こしによるカスタム語彙とカスタム語彙フィルターがサポートされるようになりました。	2022 年 3 月 11 日
新しいイベント	新しいイベントタイプがあります。語彙イベントです。	2022 年 2 月 7 日
セクション更新	カスタム語彙のセクションが更新されました。	2022 年 1 月 20 日
新機能	ストリーミング文字起こしによる言語識別が可能になりました。	2021 年 11 月 23 日
新機能	言語識別は、カスタム言語モデル、カスタム語彙、語彙フィルタリング、コンテンツリダクションで使用できるようになりました。	2021 年 10 月 29 日

新機能	Amazon Transcribe は、ストリーミング文字起こしによるカスタム言語モデルをサポートするようになりました。	2021 年 10 月 20 日
新機能	Amazon Transcribe でビデオファイルの字幕を生成できるようになりました。	2021 年 9 月 16 日
新機能	Amazon Transcribe は、ストリーミングの PII リダクションと識別をサポートするようになりました。	2021 年 9 月 14 日
新機能	Amazon Transcribe では、AWS アカウント リソースのセキュリティレベルを強化するための AWS KMS 暗号化コンテキストがサポートされるようになりました。	2021 年 9 月 10 日
新しい言語	Amazon Transcribe では、アフリカーンス語、デンマーク語、中国語 (繁体字) 、タイ語、ニュージーランド英語、南アフリカ英語がサポートされるようになりました。	2021 年 8 月 26 日
新機能	Amazon Transcribe でリソースのタグ付けがサポートされるようになりました。	2021 年 8 月 24 日
新機能	Amazon Transcribe でバッチ文字起こしジョブのコール分析がサポートされるようになりました。	2021 年 8 月 4 日

新機能	Amazon Transcribe は、バッチカスタム言語モデルでのカスタム語彙の使用をサポートするようになりました。	2021 年 5 月 12 日
新機能	Amazon Transcribe は、ストリーミング文字起こしの部分結果安定化をサポートするようになりました。	2021 年 5 月 11 日
新機能	Amazon Transcribe は、カスタム言語モデルでオーストラリア英語、英国英語、ヒンディー語、米国スペイン語をサポートするようになりました。	2021 年 3 月 19 日
新機能	Amazon Transcribe では、オーディオ文字起こしのストリーミング用に OGG/OPUS および FLAC コーデックがサポートされるようになりました。	2020 年 11 月 24 日
新しい言語	Amazon Transcribe では、音声文字起こしのストリーミングにイタリア語とドイツ語のサポートが追加されました。	2020 年 11 月 4 日
AWS リージョン 拡張	Amazon Transcribe がフランクフルト (eu-central-1) とロンドン (eu-west-2) で利用可能になりました。	2020 年 11 月 4 日

新機能	Amazon Transcribe では、バッチ文字起こしでインターフェイス VPC エンドポイントのサポートが追加されました。	2020 年 10 月 9 日
新機能	Amazon Transcribe では、ストリーミングでのチャネル識別のサポートが追加されました。	2020 年 9 月 17 日
新機能	Amazon Transcribe では、バッチ文字起こしで自動言語識別のサポートが追加されました。	2020 年 9 月 15 日
新機能	Amazon Transcribe では、ストリーミングでのスピーカーパーティショニングのサポートが追加されました。	2020 年 8 月 19 日
新機能	Amazon Transcribe では、カスタム言語モデルのサポートが追加されました。	2020 年 8 月 5 日
新機能	Amazon Transcribe では、ストリーミングでのインターフェイス VPC エンドポイントのサポートが追加されました。	2020 年 6 月 26 日
新機能	Amazon Transcribe では、ストリーミングでの語彙フィルタリングのサポートが追加されました。	2020 年 5 月 20 日

新機能	Amazon Transcribe では、個人を特定できる情報を自動的に編集するためのサポートが追加されました。	2020 年 2 月 26 日
新機能	Amazon Transcribe では、文字起こしからフィルタリングする単語のカスタム語彙を作成するためのサポートが追加されました。	2019 年 12 月 20 日
新機能	Amazon Transcribe では、文字起こしジョブのキューイングのサポートが追加されました。	2019 年 12 月 19 日
新しい言語	Amazon Transcribe では、アラビア語、ヘブライ語、日本語、マレー語、スイスドイツ語、テルグ語、トルコ語のサポートが追加されました。	2019 年 11 月 21 日
AWS リージョン 拡張	Amazon Transcribe がアジアパシフィック (東京) (ap-northeast-1) で利用可能になりました。	2019 年 11 月 21 日
新機能	Amazon Transcribe は代替文字起こしのサポートを追加します。	2019 年 11 月 20 日
新しい言語	Amazon Transcribe では、オランダ語、ペルシア語、インドネシア語、アイルランド英語、ポルトガル語、スコットランド英語、タミル語、ウェールズ英語のサポートが追加されました。	2019 年 11 月 12 日

新しい言語	Amazon Transcribe は、オーストラリア英語 (en-AU) のストリーミング文字起こしをサポートするようになりました。	2019 年 10 月 25 日
AWS リージョン 拡張	Amazon Transcribe が、中国 (北京) (cn-north-1) および中国 (寧夏) (cn-northwest-1) で利用可能になりました。	2019 年 10 月 9 日
新機能	Amazon Transcribe では、文字起こし出力ファイルを暗号化 KMS key するために独自の を指定できます。詳細については、 StartStreamTranscription API の OutputEncryptionKMSKeyId パラメータを参照してください。	2019 年 9 月 24 日
新しい言語	Amazon Transcribe では、中国語 (北京語) 、簡体字、中国本土、ロシア語のサポートが追加されました。	2019 年 8 月 23 日
新機能	Amazon Transcribe は、WebSocket プロトコルを使用した音声文字起こしのストリーミングのサポートを追加します。	2019 年 7 月 19 日
新機能	AWS CloudTrail は StartStreamTranscription API のイベントを記録するようになりました。	2019 年 7 月 19 日

AWS リージョン 拡張	Amazon Transcribe が米国西部 (北カリフォルニア) (米国西部-1) で利用可能になりました。	2019 年 6 月 27 日
新しい言語	Amazon Transcribe は、モダンスタンダードアラビア語のサポートを追加します。	2019 年 5 月 28 日
新機能	Amazon Transcribe は、数字の単語を米国英語の数字に文字起こしするようになりました。たとえば、「forty-two」は「42」に書き起こされます。	2019 年 5 月 23 日
新しい言語	Amazon Transcribe では、ヒンディー語とインド英語のサポートが追加されました。	2019 年 5 月 15 日
新しい SDK	AWS SDK for C++ が をサポートするようになりました Amazon Transcribe。	2019 年 5 月 8 日
新しい言語	Amazon Transcribe はスペイン語のサポートを追加します。	2019 年 4 月 19 日
AWS リージョン 拡張	Amazon Transcribe が欧州 (フランクフルト) (eu-central-1) およびアジアパシフィック (ソウル) (ap-northeast-2) で利用可能になりました。	2019 年 4 月 18 日

新しい言語	Amazon Transcribe では、英国英語、フランス語、カナダフランス語のストリーミング文字起こしのサポートが追加されました。	2019 年 4 月 5 日
新機能	AWS SDK for Ruby V3 が をサポートするようになりました Amazon Transcribe	2019 年 3 月 25 日
新機能	Amazon Transcribe では、音声入力で認識 Amazon Transcribe する特定の単語のリストであるカスタム語彙を使用できます。	2019 年 3 月 25 日
新しい言語	Amazon Transcribe では、ドイツ語と韓国語のサポートが追加されました。	2019 年 3 月 22 日
新しい言語	Amazon Transcribe は、米国スペイン語 (es-US) のストリーミング文字起こしをサポートするようになりました。	2019 年 2 月 7 日
AWS リージョン 拡張	Amazon Transcribe が南米 (サンパウロ) (sa-east-1) で利用可能になりました。	2019 年 2 月 7 日
AWS リージョン 拡張	Amazon Transcribe が、アジアパシフィック (ムンバイ) (ap-south-1)、アジアパシフィック (シンガポール) (ap-southeast-1)、欧州 (ロンドン) (eu-west-2)、欧州 (パリ) (eu-west3) で利用可能になりました。	2019 年 1 月 24 日

新しい言語	Amazon Transcribe では、フランス語、イタリア語、ブラジルポルトガル語のサポートが追加されました。	2018 年 12 月 20 日
新機能	Amazon Transcribe でオーディオストリームの文字起こしがサポートされるようになりました。	2018 年 11 月 19 日
新しい言語	Amazon Transcribe では、オーストラリア英語、英国英語、カナダフランス語のサポートが追加されました。	2018 年 11 月 15 日
AWS リージョン 拡張	Amazon Transcribe が、カナダ (中部) (ca-central-1) およびアジアパシフィック (シドニー) (ap-southeast-2) で利用可能になりました。	2018 年 7 月 17 日
新機能	文字起こしジョブからの出力を保存する独自の場所を指定できるようになりました。	2018 年 7 月 11 日
新機能	AWS CloudTrail と Amazon CloudWatch Events の統合が追加されました。	2018 年 6 月 28 日
新機能	Amazon Transcribe は、カスタム語彙のサポートを追加します。	2018 年 4 月 4 日
新しいガイド	これは Amazon Transcribe デベロッパーガイドの初回リリースです。	2017 年 11 月 29 日

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。