aws

デベロッパーガイド

Amazon Transcribe

Amazon Transcribe: デベロッパーガイド

Table of Contents

とは Amazon Transcribe

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。をスタンドアロンの文字起こしサービス Amazon Transcribe として使用したり、任意のアプリケーションにspeech-to-text機能を追加したりできます。

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オーディエンスに適した言語を確保したり、マルチチャ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パーティション化したりできます。

リアルタイムでメディアを文字起こし (ストリーミング) することも、 Amazon S3 バケット (バッチ) にあるメディアファイルを文字起こしすることもできます。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を確認するには、サポートされている言語および言語固有の機能 の表を参照してください。

トピック

- Amazon Transcribe および HIPAA の適格性
- 料金
- 利用可能なリージョンとクォータ

「 とは」を参照してください Amazon Transcribe。 このサービスの短いビデオツアーについては、「」を参照してください。

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

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

Amazon Transcribe および HIPAA の適格性

Amazon Transcribe は AWS HIPAA 資格および BAA の対象です。BAA のお客様は、保管中および転送中のすべての PHI を使用中に暗号化する必要があります。PHI の自動識別は、 Amazon

Transcribe が動作するすべてのリージョンで追加料金なしで利用できます。詳細については、「HIPAA の適格性と BAA」をご覧ください。

料金

Amazon Transcribe はpay-as-you-goのサービスです。料金は文字起こし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。

使用料は 1 秒ごとに課金され、1 リクエストあたりの最低料金は 15 秒です。PII コンテンツのリダクションやカスタム言語モデルなどの機能には追加料金が適用されることに注意してください。

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

利用可能なリージョンとクォータ

Amazon Transcribe は、以下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
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
ap-southeast-5 (マレーシア)	ストリーミング
ap-southeast-7 (タイ)	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

リージョン	文字起こしタイプ
eu-central-1 (フランクフルト)	バッチ、ストリーミング
eu-central-2 (チューリッヒ)	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
eu-west-3 (パリ)	バッチ
me-south-1 (バーレーン)	バッチ
mx-central-1 (メキシコ)	ストリーミング
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

リージョンのサポートは Amazon Transcribe、、Amazon Transcribe Medical、コール分析で異なります。

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 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。

Amazon Transcribe features

ユースケースに最適な Amazon Transcribe ソリューションを決定するために、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 は米国英語でのみ使用できます。
[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の文字起こしを指し、 はストリーミング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、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
アルバニア語	sq-AL	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
アムハラ語	am-ET	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
アラビア語、(ガルフ)	ar-AE	バッチ、ストリーミング	バッチ	なし	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*
アラビア語、(モダンスタンダード)	ar-SA	バッチ、ストリーミング	なし	いいえ	いいえ	いいえ	なし
Armenian	hy-AM	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
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
Azerbaijani	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、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
ベラルーシ語	be-BY	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
ベンガル語	bn-IN	バッチ、ストリーミング*	バッチ	バッチ、ストリーミング*	なし	いいえ	なし
ボスニア語	bs-BA	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

[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*
Burmese	my-MM	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
ブルガリア語	bg-BG	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
カタロニア語	ca-ES	バッチ、ストリーミング	ストリーミング	バッチ、ストリーミング	なし	いいえ	なし
中部クルド語 (イラン)	ckb-IR	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
中部クルド語 (イラク)	ckb-IQ	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
中国語、広東語	zh-HK (yue-HK)	バッチ、ストリーミング	なし	いいえ	いいえ	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
中国語 (繁体字)	zh-TW	バッチ、ストリーミング	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*
クロアチア語	hr-HR	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
チェコ語	cs-CZ	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
Danish	da-DK	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	なし	いいえ	なし
Dutch	nl-NL	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	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、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	ストリーミング	post-call
英語 英語(アイルランド)	en-IE	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	ストリーミング	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*
英語 英語 (ニュージーランド)	en-NZ	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	ストリーミング	なし
英語 英語 (スコットランド)	en-AB	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	ストリーミング	post-call
英語 英語 (南アフリカ)	en-ZA	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	ストリーミング	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、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	ストリーミング	post-call
エストニア語	et-EE	バッチ	なし	バッチ	なし	いいえ	なし
エストニア語	et-ET	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
ペルシア語	fa-IR	バッチ、ストリーミング	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*
ペルシア語、アフガニスタン	fa-AF	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
Finnish	fi-FI	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
フランス語	fr-FR	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	バッチ、ストリーミング	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	バッチ、ストリーミング	バッチ、ストリーミング	なし	バッチ、ストリーミング	post-call, real-time
ガリシア語	gl-ES	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
グルジア語	ka-GE	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
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	バッチ、ストリーミング	post-call, real-time
ドイツ語 ドイツ語 (スイス)	de-CH	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	バッチ、ストリーミング	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*
Greek	el-GR	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
グジャラート語	gu-IN	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
ハイチクレオール語	ht-HT	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
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、ストリーミング	バッチ	なし	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	バッチ	バッチ、ストリーミング	バッチ	なし	post-call
Hungarian	hu-HU	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
アイスランド語	is-IS	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
Indonesian	id-ID	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	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*
イタリア語	it-IT	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	バッチ、ストリーミング	post-call, real-time
Japanese	ja-JP	バッチ、ストリーミング	バッチ、ストリーミング	なし	バッチ、ストリーミング	なし	post-call
ジャワ語	jv-ID	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
カビル語	kab-DZ	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
カンナダ語	kn-IN	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
カザフ語	kk-KZ	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
クメール語	km-KH	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
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
Korean	ko-KR	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

[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*
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、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
リトアニア語	lt-LT	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
ルガンダ語	lg-IN	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
マケドニア語	mk-MK	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
マレー語	ms-MY	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
マラヤーラム語	ml-IN	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
Maltese	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、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
牧地マリ語	mhr-RU	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*
モンゴル語	mn-MN	バッチ	なし	バッチ	なし	いいえ	なし
ネパール語	ne-NP	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
ノルウェーブークモール語	no-NO	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	なし	いいえ	なし
オディア/オリヤ語	or-IN	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
パシュト語	ps-AF	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
Polish	pl-PL	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	なし	いいえ	なし
ポルトガル語	pt-PT	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	バッチ、ストリーミング	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	バッチ、ストリーミング	バッチ、ストリーミング	なし	バッチ、ストリーミング	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*
パンジャブ語	pa-IN	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
Romanian	ro-RO	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	なし	いいえ	なし
Russian	ru-RU	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	なし	いいえ	なし
セルビア語	sr-RS	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
シンハラ語	si-LK	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
スロバキア語	sk-SK	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
スロベニア語	sl-SI	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
ソマリ語	so-SO	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	なし	いいえ	なし
スペイン語	es-ES	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	ストリーミング	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、メキシコ語	es-MX	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
スペイン語、アメリカ	es-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
スンダ語	su-ID	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
スワヒリ語 (ケニア)	sw-KE	バッチ、ストリーミング*	バッチ	バッチ、ストリーミング*	なし	いいえ	なし
スワヒリ語 (ブルンジ)	sw-BI	バッチ、ストリーミング*	バッチ	バッチ、ストリーミング*	なし	いいえ	なし
スワヒリ語 (ルワンダ)	sw-RW	バッチ、ストリーミング*	バッチ	バッチ、ストリーミング*	なし	いいえ	なし
スワヒリ語 (タンザニア)	sw-TZ	バッチ、ストリーミング*	バッチ	バッチ、ストリーミング*	なし	いいえ	なし
スワヒリ語 (ウガンダ)	sw-UG	バッチ、ストリーミング*	バッチ	バッチ、ストリーミング*	なし	いいえ	なし
Swedish	sv-SE	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	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*
タガログ語/フィリピン語	tl-PH	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	なし	いいえ	なし
タミル語	ta-IN	バッチ、ストリーミング*	バッチ	ストリーミング*	なし	いいえ	なし
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、ストリーミング*	なし	ストリーミング*	なし	いいえ	なし
Thai	th-TH	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	なし	いいえ	なし
Turkish	tr-TR	バッチ、ストリーミング*	バッチ	バッチ、ストリーミング*	なし	いいえ	なし
Ukrainian	uk-UA	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	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、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

[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*
Vietnamese	vi-VN	バッチ、ストリーミング	バッチ	バッチ、ストリーミング	なし	いいえ	なし
ウェールズ語	cy-WL	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
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、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

*これらの言語のストリーミングは、 AWS リージョン af-south-1 (ケープタウン ）、ap-northeast-1 (東京 ）、ap-southeast-5 (マレーシア ）、ap-southeast-7 (タイ ）、cn-northwest-1 (寧夏) では使用できません。

*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 (Transcribe Medical と HealthScribe はサポートされていません)

バッチ文字起こし	ストリーミング文字起こし
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	SDK はストリーミングではサポートされていません。
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。 Amazon Transcribe

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

> **Tip**
>
> また、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、認識 Amazon Transcribe 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

- 中国語、北京語 (台湾)、繁体字の文字セット
- 中国語、広東語 (香港)、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	コード	文字	コード
а	0430	лъ	0459
б	0431	нь	045A
в	0432	ћ	045B
г	0433	ќ	045C
д	0434	#	045D
е	0435	ў	045E
ж	0436	џ	045F

文字	コード	文字	コード
з	0437	ҁ	0491
и	0438	҂	0493
й	0439	҇	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
ш	0448	һ	04BB
щ	0449	ҽ	04BD
ъ	044A	ҿ	04BF

文字	コード	文字	コード
ы	044B	#	04CA
ь	044C	ă	04D1
э	044D	ä	04D3
ю	044E	ĕ	04D7
я	044F	ə	04D9
#	0450	ʒ	04E1
ё	0451	ӣ	04E3
ђ	0452	ö	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
s	0455	ӱ	04F1
i	0456	ӳ	04F3
ї	0457	#	04F7
j	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	コード	文字	コード
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	コード	文字	コード
ء	0621	س	0633
آ	0622	ش	0634
أ	0623	ص	0635
ؤ	0624	ض	0636
إ	0625	ط	0637
ئ	0626	ظ	0638
ا	0627	ع	0639
ب	0628	غ	063A

文字	コード	文字	コード
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	コード	文字	コード
á	00E1	ñ	00F1
é	00E9	ó	00F3

文字	コード	文字	コード
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	コード	文字	コード
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	コード	文字	コード
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	ք	0580
ծ	056E	օ	0581
կ	056F	ֆ	0582
հ	0570	և	0583
ձ	0571	f	0584
ղ	0572	o	0585
ճ	0573	֊	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	コード	文字	コード
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	ҡ	049F
н	043D	ҡ	04A1

文字	コード	文字	コード
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	コード	文字	コード
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

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	コード	文字	コード
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	コード	文字	コード
а	0430	с	0441
б	0431	т	0442
в	0432	у	0443
г	0433	ф	0444
д	0434	х	0445
е	0435	ц	0446
ж	0436	ч	0447
з	0437	ш	0448
й	0439	ы	044B
к	043A	ь	044C
л	043B	э	044D
м	043C	ю	044E
н	043D	я	044F
о	043E	ё	0451

文字	コード	文字	コード
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	コード	文字	コード
ঁ	0981	দ	09A6
ং	0982	ধ	09A7
ঃ	0983	ন	09A8
অ	0985	প	09AA
আ	0986	ফ	09AB
ই	0987	ব	09AC
ঈ	0988	ভ	09AD
উ	0989	ম	09AE
ঊ	098A	য	09AF
ঋ	098B	র	09B0
এ	098F	ল	09B2

文字	コード	文字	コード
ঐ	0990	শ	09B6
ও	0993	ষ	09B7
ঔ	0994	স	09B8
ক	0995	হ	09B9
খ	0996	়	09BC
গ	0997	#	09BD
ঘ	0998	া	09BE
ঙ	0999	ি	09BF
চ	099A	ী	09C0
ছ	099B	ু	09C1
জ	099C	ূ	09C2
ঝ	099D	ৃ	09C3
ঞ	099E	ৄ	09C4
ট	099F	ে	09C7
ঠ	09A0	ৈ	09C8
ড	09A1	ো	09CB
ঢ	09A2	ৌ	09CC
ণ	09A3	্	09CD
ত	09A4	#	09CE
থ	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	コード	文字	コード
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	コード	文字	コード
а	0430	п	043F
б	0431	р	0440
в	0432	с	0441
г	0433	т	0442

文字	コード	文字	コード
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	コード	文字	コード
à	00E0	ï	00EF
ç	00E7	ò	00F2

文字	コード	文字	コード
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	コード	文字	コード
ئ	0626	م	0645
ا	0627	ن	0646
ب	0628	و	0648
ت	062A	پ	067E
ج	062C	چ	0686
ح	062D	ڕ	0695
خ	062E	ژ	0698
د	062F	ڤ	06A4
ر	0631	ک	06A9

文字	コード	文字	コード
ز	0632	گ	06AF
س	0633	ڵ	06B5
ش	0634	ھ	06BE
ع	0639	ۆ	06C6
غ	063A	ۇ	06C7
ف	0641	ی	06CC
ق	0642	ێ	06CE
ل	0644	ە	06D5

中国語、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、数字を使用した中国語 (簡体字) の 4 つのトーンを表します。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

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

> **Note**
>
> 5 番目 (ニュートラル) 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、数字を使用した中国語 (繁体字) の 4 つのトーンを表します。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
#####     #ˋ-###ˋ-##ˊ-##ˇ-###ˋ            ###
```

中国語、広東語 (香港)、繁体字の文字セット

中国語 (広東語) 香港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-hk-character-set

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	コード	文字	コード
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	コード	文字	コード
á	00E1	ď	010F
é	00E9	ě	011B
í	00ED	ň	0148
ó	00F3	ř	0159

文字	コード	文字	コード
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	コード	文字	コード
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	コード	文字	コード
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	コード	文字	コード
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	コード	文字	コード
ء	0621	ظ	0638
آ	0622	ع	0639
أ	0623	غ	063A
ؤ	0624	ف	0641
ئ	0626	ق	0642
ا	0627	ل	0644

文字	コード	文字	コード
ب	0628	م	0645
ت	062A	ن	0646
ث	062B	ه	0647
ج	062C	و	0648
ح	062D	ً	064E
خ	062E	ٌ	064F
د	062F	ٍ	0650
ذ	0630	ّ	0651
ر	0631	پ	067E
ز	0632	چ	0686
س	0633	ژ	0698
ش	0634	ک	06A9
ص	0635	گ	06AF
ض	0636	ی	06CC
ط	0637		

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	コード	文字	コード
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	コード	文字	コード
À	00C0	à	00E0
Â	00C2	â	00E2
Ç	00C7	ç	00E7
È	00C8	è	00E8
É	00C9	é	00E9
Ê	00CA	ê	00EA
Ë	00CB	ë	00EB

文字	コード	文字	コード
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	コード	文字	コード
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	コード	文字	コード
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	コード	文字	コード
ʒ	10DE	ɮ	10EE
ʤ	10DF	ʧ	10EF
ʓ	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	コード	文字	コード
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	コード	文字	コード
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	コード	文字	コード
ઁ	0A81	૬	0AA6
ં	0A82	૧	0AA7
ઃ	0A83	૨	0AA8
અ	0A85	૫	0AAA
આ	0A86	૬	0AAB
ઇ	0A87	૭	0AAC
ઈ	0A88	૮	0AAD
ઉ	0A89	૯	0AAE
ઊ	0A8A	૟	0AAF
ઋ	0A8B	ૠ	0AB0
ઍ	0A8D	ૡ	0AB2
એ	0A8F	ૢ	0AB3
ઐ	0A90	૤	0AB5
ઑ	0A91	૥	0AB6

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

Amazon Transcribe は、次の文字をマッピングします。

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

文字	コード	文字	コード
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

韓国語のカスタム語彙の場合、 Phraseフィールドで以下を使用できます。

- a ~ z
- A ~ Z
- - (ハイフン)
- . (ピリオド)

Phrase フィールドは、次のリンクにリストされている任意の Hangul 音節を使用できます。

- Wikipedia の Hangul Syllables。

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

文字	コード	文字	コード
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7
ш	0448	һ	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
ѕ	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	#	04F7

文字	コード	文字	コード
j	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	コード	文字	コード
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	コード	文字	コード
ą	0105	į	012F
č	010D	š	0161
ę	0119	ų	0173
ė	0117	ū	016B
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	コード	文字	コード
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	コード	文字	コード
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
о	043E	ң	04A3
п	043F	ҥ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB
т	0442	ҭ	04AD

文字	コード	文字	コード
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7
ш	0448	һ	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
і	0456	ӳ	04F3

文字	コード	文字	コード
ï	0457	#	04F7
j	0458	ы	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	コード	文字	コード
ം	0D02	ന	0D28
ഃ	0D03	പ	0D2A
അ	0D05	ഫ	0D2B
ആ	0D06	ബ	0D2C

文字	コード	文字	コード
ഇ	0D07	ഭ	0D2D
ഈ	0D08	മ	0D2E
ഉ	0D09	യ	0D2F
ഊ	0D0A	ര	0D30
ഋ	0D0B	റ	0D31
ഌ	0D0E	ല	0D32
ഏ	0D0F	ള	0D33
ഐ	0D10	ഴ	0D34
ഒ	0D12	വ	0D35
ഓ	0D13	ശ	0D36
ഔ	0D14	ഷ	0D37
ക	0D15	സ	0D38
ഖ	0D16	ഹ	0D39
ഗ	0D17	ാ	0D3E
ഘ	0D18	ി	0D3F
ങ	0D19	ീ	0D40
ച	0D1A	ു	0D41
ഛ	0D1B	ൂ	0D42
ജ	0D1C	ൃ	0D43
ഝ	0D1D	െ	0D46

文字	コード	文字	コード
ഞ	0D1E	േ	0D47
ട	0D1F	ൈ	0D48
ഠ	0D20	ൊ	0D4A
ഡ	0D21	ോ	0D4B
ഢ	0D22	ൗ	0D4C
ണ	0D23	്	0D4D
ത	0D24	#	0D7A
ഥ	0D25	#	0D7B
ദ	0D26	#	0D7C
ധ	0D27	#	0D7D

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	コード	文字	コード
à	00E0	ù	00F9
è	00E8	ċ	010B
ì	00EC	ġ	0121

文字	コード	文字	コード
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	コード	文字	コード
ँ	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

文字	コード	文字	コード
ए	090F	ल	0932
ऐ	0910	ळ	0933
ऑ	0911	व	0935
ओ	0913	श	0936
औ	0914	ष	0937
क	0915	स	0938
ख	0916	ह	0939
ग	0917	़	093C
घ	0918	ा	093E
ङ	0919	ि	093F
च	091A	ी	0940
छ	091B	ु	0941
ज	091C	ू	0942
झ	091D	ृ	0943
ञ	091E	ॅ	0945
ट	091F	ॉ	0947
ठ	0920	ॊ	0948
ड	0921	ॅ	0949
ढ	0922	ो	094B
ण	0923	ौ	094C

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

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

文字	コード	文字	コード
я	044F	ә	04D9
#	0450	ӡ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
ѕ	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	#	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	コード	文字	コード
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B

文字	コード	文字	コード
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	ҝ	0499
л	043B	қ	049B
м	043C	ҟ	049F
н	043D	ҡ	04A1
о	043E	ң	04A3
п	043F	ҥ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB
т	0442	ҭ	04AD
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5

文字	コード	文字	コード
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
і	0456	ӳ	04F3
ї	0457	#	04F7
ј	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	コード	文字	コード
å	00E5	æ	00E6
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	コード	文字	コード
◌ଁ	0B01	ଦ	0B26
ଂ	0B02	ଧ	0B27
ଃ	0B03	ନ	0B28

文字	コード	文字	コード
ଅ	0B05	ପ	0B2A
ଆ	0B06	ଫ	0B2B
ଇ	0B07	ବ	0B2C
ଈ	0B08	ଭ	0B2D
ଉ	0B09	ମ	0B2E
ଊ	0B0A	ଯ	0B2F
ଋ	0B0B	ର	0B30
ଏ	0B0F	ଲ	0B32
ଐ	0B10	ଳ	0B33
ଓ	0B13	ଗ	0B36
ଔ	0B14	ଷ	0B37
କ	0B15	ସ	0B38
ଖ	0B16	ହ	0B39
ଗ	0B17	଼	0B3C
ଘ	0B18	ା	0B3E
ଙ	0B19	ି	0B3F
ଚ	0B1A	ୀ	0B40
ଛ	0B1B	ୁ	0B41
ଜ	0B1C	ୂ	0B42
ଝ	0B1D	ୃ	0B43

文字	コード	文字	コード
ଞ	0B1E	୆	0B47
ଟ	0B1F	େ	0B48
ଠ	0B20	େ	0B4B
ଡ	0B21	ୈ	0B4C
ଢ	0B22	୍	0B4D
ଣ	0B23	ୖ	0B56
ତ	0B24	ୟ	0B5F
ଥ	0B25	ୠ	0B60
#	0B71		

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	コード	文字	コード
آ	0622	و	0648
أ	0623	ي	064A
ؤ	0624	ّ	064B
ئ	0626	ٔ	064C

文字	コード	文字	コード
ا	0627	ٍ	064D
ب	0628	َ	064E
ت	062A	ُ	064F
ث	062B	ِ	0650
ج	062C	ّ	0651
ح	062D	ْ	0652
خ	062E	#	0654
د	062F	ٰ	0670
ذ	0630	ټ	067C
ر	0631	پ	067E
ز	0632	ځ	0681
س	0633	څ	0685
ش	0634	چ	0686
ص	0635	ډ	0689
ض	0636	ړ	0693
ط	0637	ږ	0696
ظ	0638	ژ	0698
ع	0639	ښ	069A
غ	063A	ک	06A9
ف	0641	ګ	06AB

文字	コード	文字	コード
ق	0642	گ	06AF
ل	0644	ڽ	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	コード	文字	コード
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	コード	文字	コード
À	00C0	à	00E0
Á	00C1	á	00E1
Â	00C2	â	00E2
Ã	00C3	ã	00E3
Ä	00C4	ä	00E4
Ç	00C7	ç	00E7
È	00C8	è	00E8
É	00C9	é	00E9
Ê	00CA	ê	00EA
Ë	00CB	ë	00EB
Í	00CD	í	00ED
Ñ	00D1	ñ	00F1
Ó	00D3	ó	00F3
Ô	00D4	ô	00F4
Õ	00D5	õ	00F5

文字	コード	文字	コード
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	コード	文字	コード
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

文字	コード	文字	コード
ਕ	0A15	ਵ	0A35
ਖ	0A16	ਸ	0A38
ਗ	0A17	ਹ	0A39
ਘ	0A18	਼	0A3C
ਙ	0A19	ਾ	0A3E
ਚ	0A1A	ਿ	0A3F
ਛ	0A1B	ੀ	0A40
ਜ	0A1C	ੁ	0A41
ਝ	0A1D	ੂ	0A42
ਞ	0A1E	ੇ	0A47
ਟ	0A1F	ੈ	0A48
ਠ	0A20	ੋ	0A4B
ਡ	0A21	ੌ	0A4C
ਢ	0A22	੍	0A4D
ਣ	0A23	ੜ	0A5C
ਤ	0A24	ੰ	0A70
ਥ	0A25	ੱ	0A71
ਦ	0A26	ੲ	0A72
ਉ	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	コード	文字	コード
ă	0103	#	0219
â	00E2	#	021B
î	00EE	ș	015F
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	コード	文字	コード
'	0027	п	043F
-	002D	р	0440
.	002E	с	0441
а	0430	т	0442
б	0431	у	0443
в	0432	ф	0444

文字	コード	文字	コード
г	0433	х	0445
д	0434	ц	0446
е	0435	ч	0447
ж	0436	ш	0448
з	0437	щ	0449
и	0438	ъ	044A
й	0439	ы	044B
к	043A	ь	044C
л	043B	э	044D
м	043C	ю	044E
н	043D	я	044F
о	043E	ё	0451

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	コード	文字	コード
ć	0107	i	0456

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

文字	コード	文字	コード
р	0440	ҭ	04AD
с	0441	ү	04AF
т	0442	ұ	04B1
у	0443	ҳ	04B3
ф	0444	ҵ	04B5
х	0445	ҷ	04B7
ц	0446	h	04BB
ч	0447	ҽ	04BD
ш	0448	ҿ	04BF
щ	0449	#	04CA
ъ	044A	ӑ	04D1
ы	044B	ӓ	04D3
ь	044C	ӗ	04D7
э	044D	ә	04D9
ю	044E	ӟ	04E1
я	044F	ӣ	04E3
#	0450	ӧ	04E7
ё	0451	ө	04E9
ђ	0452	ӯ	04EF
ѓ	0453	ӱ	04F1

文字	コード	文字	コード
є	0454	ӯ	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	コード	文字	コード
#	0D82	#	0DAF
#	0D83	#	0DB0
#	0D85	#	0DB1
#	0D86	#	0DB3
#	0D87	#	0DB4
#	0D88	#	0DB5
#	0D89	#	0DB6
#	0D8A	#	0DB7
#	0D8B	#	0DB8
#	0D8C	#	0DB9

文字	コード	文字	コード
#	0D8D	#	0DBA
#	0D91	#	0DBB
#	0D92	#	0DBD
#	0D93	#	0DC0
#	0D94	#	0DC1
#	0D95	#	0DC2
#	0D96	#	0DC3
#	0D9A	#	0DC4
#	0D9B	#	0DC5
#	0D9C	#	0DC6
#	0D9D	#	0DCA
#	0D9E	#	0DCF
#	0D9F	#	0DD0
#	0DA0	#	0DD1
#	0DA1	#	0DD2
#	0DA2	#	0DD3
#	0DA3	#	0DD4
#	0DA4	#	0DD6
#	0DA5	#	0DD8
#	0DA7	#	0DD9

文字	コード	文字	コード
#	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	コード	文字	コード
á	00E1	ň	0148
ä	00E4	ó	00F3
č	010D	ô	00F4
ď	010F	ŕ	0155
é	00E9	š	0161
í	00ED	ť	0165

文字	コード	文字	コード
Í	013A	ú	00FA
Ǐ	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	コード	文字	コード
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	コード	文字	コード
s	0073	d	0064
t	0074	ある	0061
ある	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	コード	文字	コード
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	コード	文字	コード
s	0073	d	0064
t	0074	ある	0061
ある	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	コード	文字	コード
s	0073	d	0064
t	0074	ある	0061
ある	0061	r	0072

文字	コード	文字	コード
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	コード	文字	コード
Ä	00C4	ä	00E4
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	コード		
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	コード	文字	コード
அ	0B85	ர	0BB0
ஆ	0B86	ல	0BB2
இ	0B87	வ	0BB5
ஈ	0B88	ழ	0BB4
உ	0B89	ள	0BB3
ஊ	0B8A	ற	0BB1
எ	0B8E	ன	0BA9
ஏ	0B8F	ஜ	0B9C
ஐ	0B90	#	0BB6
ஒ	0B92	ஷ	0BB7
ஓ	0B93	ஸ	0BB8
ஔ	0B94	ஹ	0BB9
ஃ	0B83	்	0BCD
க	0B95	ா	0BBE
ங	0B99	ி	0BBF

文字	コード	文字	コード
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	コード	文字	コード
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C

文字	コード	文字	コード
д	0434	ҕ	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
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	コード	文字	コード
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

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

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

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

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

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

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

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	コード	文字	コード
#	0611	و	0648
#	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

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

文字	コード	文字	コード
ۏ	0647	ڔ	0693
ڕ	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	コード	文字	コード
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

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

ベトナム語の文字セット

Amazon Transcribe は、数字を使用したベトナム語の 6 つのトーンを表します。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	コード	文字	コード
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

文字	コード	文字	コード
ú	00FA	Ú	00DA
ý	00FD	Ý	00DD
ă	0103	Ă	0102
đ	0111	Đ	0110
ĩ	0129	Ĩ	0128
ũ	0169	Ũ	0168
ơ	01A1	Ơ	01A0
ư	01B0	Ư	01AF
ạ	1EA1	Ạ	1EA0
ả	1EA3	Ả	1EA2
ấ	1EA5	Ấ	1EA4
ầ	1EA7	Ầ	1EA6
ẩ	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

文字	コード	文字	コード
ẹ	1EB9	Ẹ	1EB8
ẻ	1EBB	Ẻ	1EBA
ẽ	1EBD	Ẽ	1EBC
ế	1EBF	Ế	1EBE
ề	1EC1	Ề	1EC0
ể	1EC3	Ể	1EC2
ễ	1EC5	Ễ	1EC4
ệ	1EC7	Ệ	1EC6
ỉ	1EC9	Ỉ	1EC8
ị	1ECB	Ị	1ECA
ọ	1ECD	Ọ	1ECC
ỏ	1ECF	Ỏ	1ECE
ố	1ED1	Ố	1ED0
ồ	1ED3	Ồ	1ED2
ổ	1ED5	Ổ	1ED4
ỗ	1ED7	Ỗ	1ED6
ộ	1ED9	Ộ	1ED8
ớ	1EDB	Ớ	1EDA
ờ	1EDD	Ờ	1EDC
ở	1EDF	Ở	1EDE

文字	コード	文字	コード
ỡ	1EE1	Ỡ	1EE0
ợ	1EE3	Ợ	1EE2
ụ	1EE5	Ụ	1EE4
ủ	1EE7	Ủ	1EE6
ứ	1EE9	Ứ	1EE8
ừ	1EEB	Ừ	1EEA
ử	1EED	Ử	1EEC
ữ	1EEF	Ữ	1EEE
ự	1EF1	Ự	1EF0
ỳ	1EF3	Ỳ	1EF2
ỵ	1EF5	Ỵ	1EF4
ỷ	1EF7	Ỷ	1EF6
ỹ	1EF9	Ỹ	1EF8

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	コード	文字	コード
à	00E0	ò	00F2
á	00E1	ó	00F3
â	00E2	ô	00F4
ä	00E4	ö	00F6
è	00E8	ù	00F9
é	00E9	ú	00FA
ê	00EA	û	00FB
ë	00EB	ü	00FC
ì	00EC	ý	00FD
í	00ED	ÿ	00FF
î	00EE	ŵ	0175
ï	00EF	ŷ	0177
ỳ	1EF3		

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	コード	文字	コード
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	コード	文字	コード
s	0073	d	0064
t	0074	ある	0061
ある	0061	r	0072
n	006E	d	0064

の Amazon Transcribe 仕組み

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します。

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、各単語や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完全なリストについては、「機能の概要」を参照してください。

文字起こしの方法は、次の 2 つの主要なカテゴリに分類できます。

- バッチ文字起こし: 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文字起こしします。AWS CLI、AWS マネジメントコンソール、およびさまざまな AWS SDK を使用してバッチ文字起こしを行うことができます。
- ストリーミング文字起こし: 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。ストリーミング文字起こしにはAWS マネジメントコンソール、HTTP/2、WebSocket、およびさまざまな AWS SDK を使用できます。

機能と言語のサポートは、バッチ文字起こしとストリーミング文字起こしで異なることに注意してください。さらなる詳細については、「Amazon Transcribe features」と「サポートされている言語」を参照してください。

トピック

- データの入力および出力
- 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
>
> バッチ: `StartTranscriptionJob`
> ストリーミング: `StartStreamTranscription`、startStreamTranscriptionWebSocket

データの入力および出力

Amazon Transcribe は、音声データを Amazon S3 バケットまたはメディアストリームのメディアファイルとして受け取り、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します。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していることになります。この 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、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、ジョブキューイング を使用します。これにより、 Amazon Transcribe は文字起こしジョブを追跡し、スロットが使用可能になったときに処理できます。

> **Note**
>
> Amazon Transcribe は、分析モデルの品質を継続的に向上させるために、コンテンツを一時的に保存することがあります。詳細については、「Amazon Transcribe よくある質問」を参照してください。によって保存されている可能性のあるコンテンツの削除をリクエストするには Amazon Transcribe、 でケースを開きますサポート。

トピック

- メディア形式
- 音声チャネル
- サンプルレート
- Output

メディア形式

サポートされるメディアタイプは、バッチ文字起こしとストリーミング文字起こしでは異なりますが、どちらにも可逆形式が推奨されます。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	• AMR • FLAC • M4A • MP3 • MP4 • Ogg	• FLAC • Ogg Opus • PCM エンコーディング

	バッチ	ストリーミング
	• WebM • WAV	
推奨形式	• FLAC • PCM 16 ビットエンコーディング の WAV	• FLAC • PCM 16 ビット符号付き リトルエンディアンの音声 (WAV は含まない)

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 などの可逆形式を使用します。

> **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。

音声チャネル

Amazon Transcribe は、シングルチャネルとデュアルチャネルのメディアをサポートしています。2 チャネルを超えるメディアは現在サポートされていません。

音声の 1 つのチャネルに複数のスピーカーが含まれていて、文字起こし出力で各スピーカーを分割してラベル付けしたい場合は、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。

音声に 2 つの異なるチャンルの音声が含まれている場合は、チャネル識別を使用して、トランスクリプト内の各チャ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どちらも 1 つのトランスクリプトファイルを作成します。

> **Note**
>
> スピーカーパーティショニングまたはチャ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

バッチ文字起こしジョブでは、サンプルレートを選択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、指定する値が音声の実際のサンプルレートと一致することを確認します。音声と一致しないサンプルレートを指定すると、ジョブが失敗することがあります。

ストリーミング文字起こしでは、リクエストにサンプルレートを含める必要があります。バッチ文字起こしジョブと同様に、指定する値が音声の実際のサンプルレートと一致していることを確認します。

電話録音などの低音質音声のサンプルレートは、通常 8,000 Hz を使用します。忠実度の高いオーディオの場合、 は 16,000 Hz ~ 48,000 Hz の値 Amazon Transcribe をサポートします。

Output

文字起こしの出力は JSON 形式です。トランスクリプトの最初の部分には、トランスクリプトそのものが段落形式で含まれ、その後に単語と句読点ごとに追加データが続きます。提供されるデータは、リクエストに含めた機能によって異なります。少なくとも、トランスクリプトにはすべての単語の開始時刻、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を含まない基本的な文字起こしリクエストの出力例を示しています。

すべてのバッチトランスクリプトは Amazon S3 バケットに保存されます。トランスクリプトを独自の Amazon S3 バケットに保存するか、安全なデフォルトバケット Amazon Transcribe を使用するかを選択できます。 Amazon S3 バケットの作成と使用について詳しくは、「バケットの使用」を参照してください。

自分の所有する Amazon S3 バケットにトランスクリプトを保存する場合は、文字起こし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、必ずこのバケットの Amazon Transcribe 書き込みアクセス許可を付与してください。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、 は安全なサービスマネージドバケット Amazon Transcribe を使用し、トランスクリプトのダウンロードに使用できる一時的な URI を提供します。一時的な 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 URI の使用中に `AccessDenied` 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 URI を取得する `GetTranscriptionJob` リクエストを行ってください。

デフォルトバケットを選択した場合、ジョブの有効期限 (90 日) になると、トランスクリプトは削除されます。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 出力を単語形式のターンバイターンのトランスクリプトに変換したい場合は、この「GitHub の例 (Python3 用)」を参照してください。このスクリプトは、通話後分析文字起こしや、ダイアライゼーションが有効になっている標準のバッチ文字起こしで動作します。

出力の例

トランスクリプトでは、段落形式で完全な文字起こしが得られ、その後に単語ごとの内訳が記載され、すべての単語と句読点のデータが示されます。これには、開始時間、終了時間、信頼スコア、タイプ (pronunciation または punctuation) が含まれます。

次の例は、追加機能を含まないシンプルなバッチ文字起こしジョブのものです。文字起こしリクエストに追加機能を適用するたびに、文字起こし出力ファイルに追加のデータが追加されます。

基本的なバッチ文字起こしには主に以下の 2 つのセクションがあります。

1. `transcripts`: 1 つのテキストブロックにトランスクリプト全体が含まれます。
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ます。
3. `audio_segments`: 音声セグメントとは、オーディオ録音の中で最小限の一時停止や中断のみを含む、途切れない音声言語の特定の部分を指します。このセグメントは自然な音声の流れをキャプチャし、開始時刻と終了時刻と共に `audio_segments` にキャプチャされます。音声セグメント内の `items` 要素は、セグメント内の各項目に対応する一連の識別子です。

文字起こしリクエストに追加機能を含めるたびに、トランスクリプトに追加情報が生成されます。

```
{
    "jobName": "my-first-transcription-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "Welcome to Amazon Transcribe."
            }
        ],
        "items": [
            {
```

```
                "id": 0,
                "start_time": "0.64",
                "end_time": "1.09",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Welcome"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 1,
                "start_time": "1.09",
                "end_time": "1.21",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "to"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 2,
                "start_time": "1.21",
                "end_time": "1.74",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Amazon"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 3,
                "start_time": "1.74",
                "end_time": "2.56",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Transcribe"
                    }
```

```
                ],
                "type": "pronunciation"
            },
            {
                "id": 4,
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ],
        "audio_segments": [
            {
                "id": 0,
                "transcript": "Welcome to Amazon Transcribe.",
                "start_time": "0.64",
                "end_time": "2.56",
                "items": [
                    0,
                    1,
                    2,
                    3,
                    4
                ]
            }
        ]
    },
    "status": "COMPLETED"
}
```


数字と句読点の文字起こし

Amazon Transcribe は、サポートされているすべての言語に句読点を自動的に追加し、記述システムで大文字と小文字を区別する言語に単語を適切に大文字にします。

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、数字の文字起こしをサポートしている言語では、 Amazon Transcribe は数字の使用状況に応じて異なる処理を行います。

たとえば、話者が「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」と言うと、次のように文字起こしされます。

- 数字の文字起こしをサポートしている言語: 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

数字の文字起こしをサポートしている言語を確認するには、「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。

の開始方法 Amazon Transcribe

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。

- AWS アカウントにサインアップする
- AWS CLI と SDKs をインストールする (文字起こし AWS マネジメントコンソール に を使用している場合は、このステップをスキップできます)
- IAM 認証情報の設定
- Amazon S3 バケットをセットアップする
- IAM ポリシーを作成する

これらの前提条件を満たしたら、文字起こしの準備は完了です。開始するには、以下のリストから希望の文字起こし方法を選択します。

- AWS CLI
- AWS マネジメントコンソール
- AWS SDK
- HTTP
- WebSocket

> **Tip**
>
> を初めて使用する場合 Amazon Transcribe や、機能を試したい場合は、 を使用することをお勧めしますAWS マネジメントコンソール。また、コンピュータのマイクを使ってストリームを開始したい場合も、これが最も簡単なオプションです。

HTTP/2 と WebSocket を使ったストリーミングは他の文字起こし方法よりも複雑なので、これらの方法を使い始める前に ストリーミング文字起こしの設定 セクションを確認することをおすすめします。ストリーミング文字起こしには SDK を使用することを強くおすすめします。

にサインアップする AWS アカウント

の使用を開始するには AWS、 が必要です AWS アカウント。の作成の詳細については AWS アカウント、「 AWS アカウント管理 リファレンスガイド」の「 の開始方法 AWS アカウント」を参照してください。

AWS CLI および SDKs のインストール

Amazon Transcribe API を使用するには、まず をインストールする必要があります AWS CLI。現在の AWS CLI はバージョン 2 です。Linux、Mac、Windows、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 用に設定する必要があります AWS リージョン。

SDK Amazon Transcribe で を使用する場合は、インストール手順に使用する言語を選択します。

- .NET
- C++
- Go
- Java V2
- JavaScript
- PHP v3
- AWS SDK for Python (Boto3) (バッチ文字起こし)
- Python (ストリーミング文字起こし)
- Ruby V3
- Rust (バッチ文字起こし)
- Rust (ストリーミング文字起こし)

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。 はファイル (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 Amazon S3 保存します。

バッチ文字起こしを実行するには、まずメディアファイルを Amazon S3 バケットにアップロードする必要があります。文字起こし出力に Amazon S3 バケットを指定しない場合、 Amazon Transcribe

はトランスクリプトを一時 AWS管理 Amazon S3 バケットに配置します。管理 AWS対象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を説明します。

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、またはロール) または AWS リソースにアタッチする必要があります。ポリシーは、それがアタッチされているエンティティのアクセス許可を定義します。たとえば、ロールは、アクセスを許可するポリシーをそのロールにアタッチしている場合にのみ、 Amazon S3 バケットにあるメディアファイルにアクセスできます。そのロールをさらに制限する場合は、代わりにそのアクセスを Amazon S3 バケット内の特定のファイルに制限できます。

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。

- のポリシーとアクセス許可 IAM
- IAM ポリシーの作成
- が IAM と Amazon Transcribe 連携する方法

で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のポリシーの例。カスタムポリシーを生成する場合は、AWS Policy Generator の使用を検討してください。

、 AWS マネジメントコンソール AWS 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、IAM 「ID アクセス許可の追加と削除」を参照してください。

ポリシーには形式があります。

JSON

```
{
    "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 (ARNs、 Amazon S3 バケットなどのすべての AWS リソースを一意に識別します。ポリシーで ARN を使用して、特定のアクションに対して特定のリソースを使用するアクセス許可を付与できます。たとえば、 Amazon S3 バケットとそのサブフォルダへの読み取りアクセスを許可する場合は、信頼ポリシーStatement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:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
}
```

Amazon S3 バケット、、amzn-s3-demo-bucketおよびそのサブフォルダに Amazon Transcribe 読み取り (GetObject、ListBucket) および書き込み (PutObject) アクセス許可を付与するポリシーの例を次に示します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
             "Effect": "Allow",
             "Action": [
                 "s3:PutObject"
             ],
             "Resource": [
                 "arn:aws:s3:::amzn-s3-demo-bucket",
                 "arn:aws:s3:::amzn-s3-demo-bucket/*"
             ]
        }
    ]
}
```

を使用した文字起こし AWS マネジメントコンソール

AWS コンソールは、バッチ文字起こしとストリーミング文字起こしに使用できます。 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。

バッチ文字起こしを開始する前に、まずメディアファイルを Amazon S3 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場合は AWS マネジメントコンソール、コンピュータのマイクを使用する必要があります。

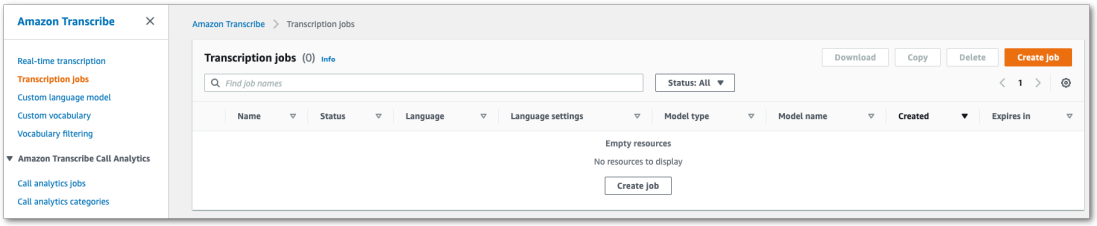
サポートされているメディア形式とその他のメディア要件と制約については、「データの入力および出力」を参照してください。

以下のセクションを展開して、文字起こしのそれぞれの方法の簡単なチュートリアルをご覧ください。

バッチ文字起こし

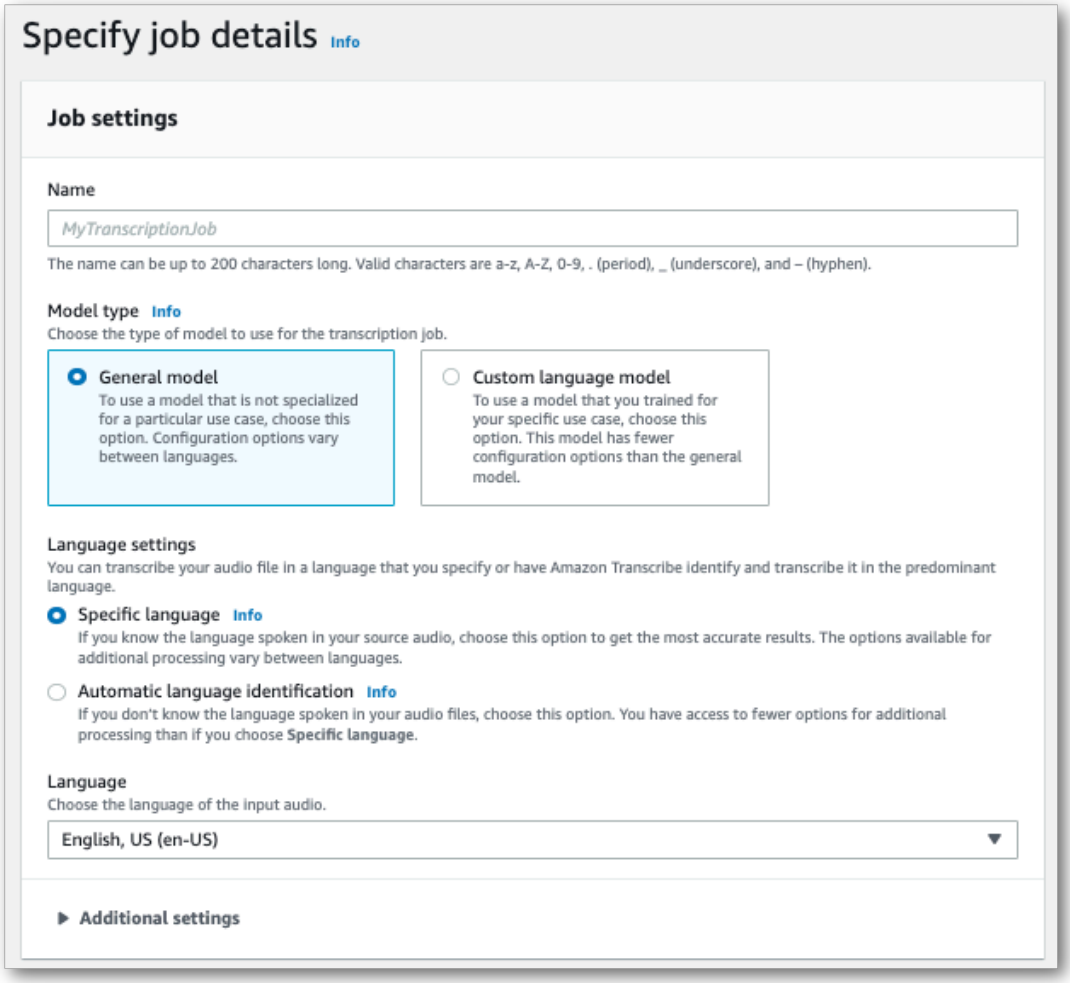
まず、文字起こししたいメディアファイルを Amazon S3 バケットにアップロードしたことを確認します。この方法がわからない場合は、「Amazon S3 ユーザーガイド」:「バケットにオブジェクトをアップロードする」を参照してください。

1. AWS マネジメントコンソール から、左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、文字起こしジョブのリストが表示されます。



[ジョブの作成] を選択します。

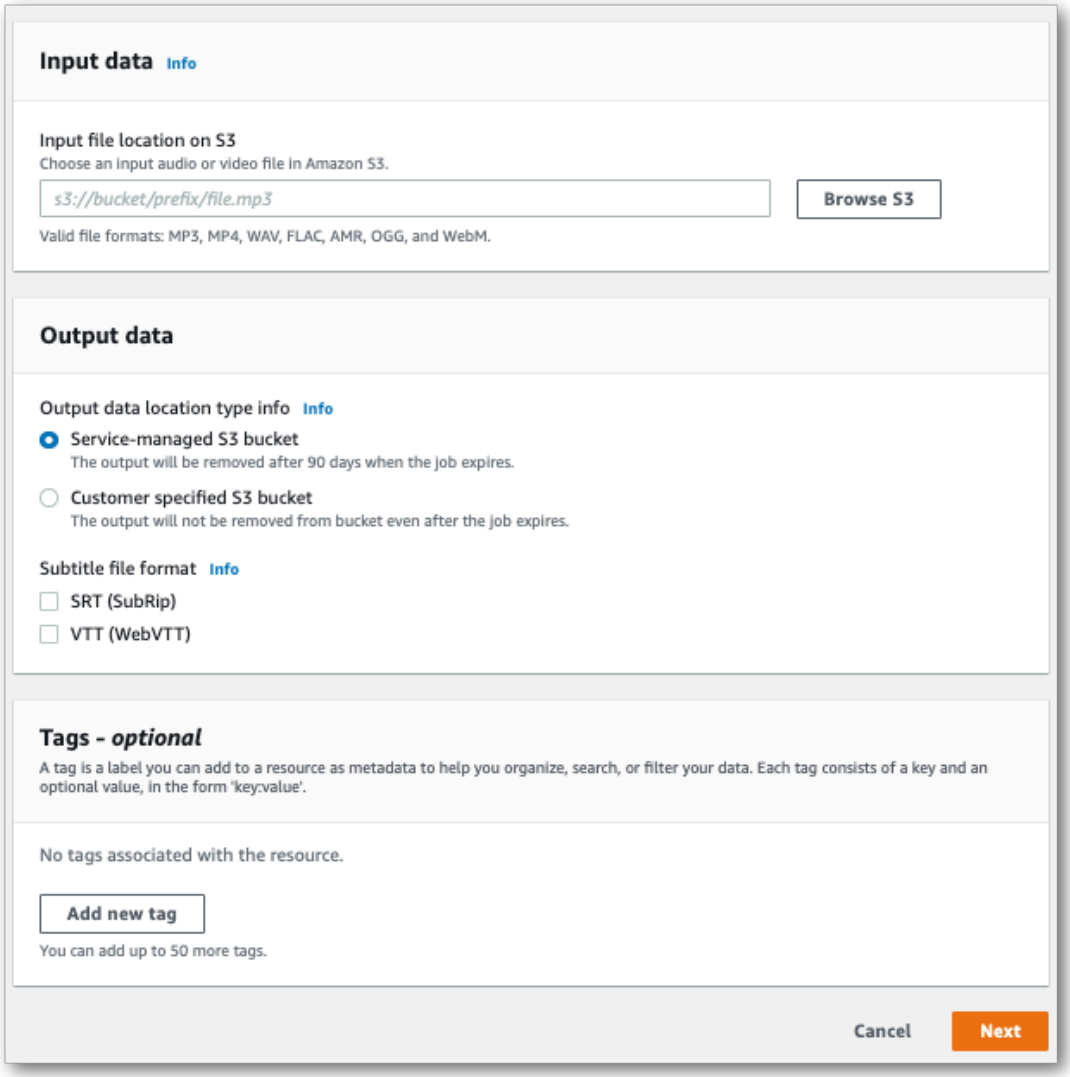
2. 「ジョブの詳細を指定」ページのフィールドに入力します。



入力場所は、 Amazon S3 バケット内のオブジェクトである必要があります。出力場所として、安全な Amazon S3 サービスマネージドバケットを選択するか、独自の Amazon S3 バケットを指定できます。

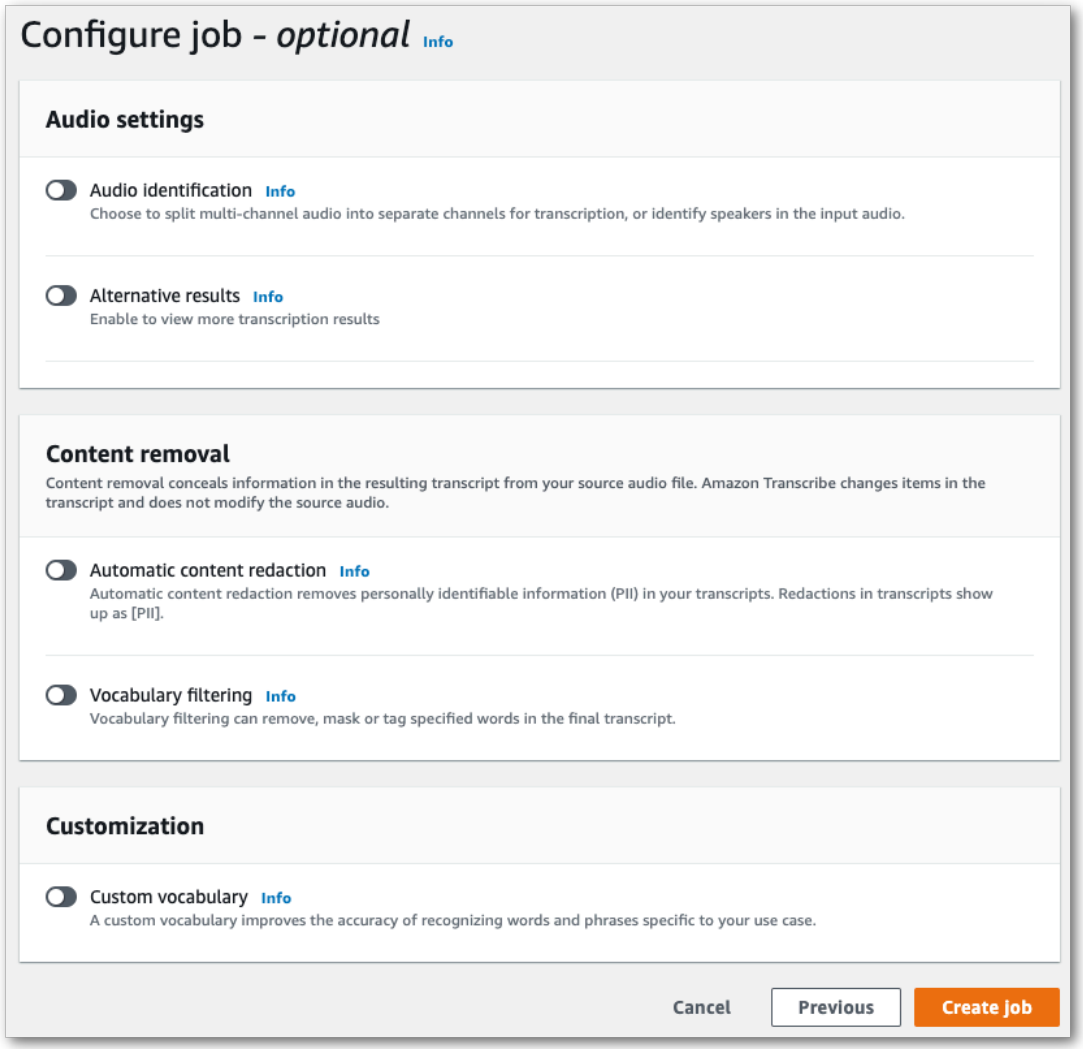
サービスマネージドバケットを選択した場合、 AWS マネジメントコンソール でトランスクリプトのプレビューを表示したり、ジョブの詳細ページ (以下を参照) からトランスクリプトをダウンロードしたりできます。

独自の Amazon S3 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 AWS マネジメントコンソール 、 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。



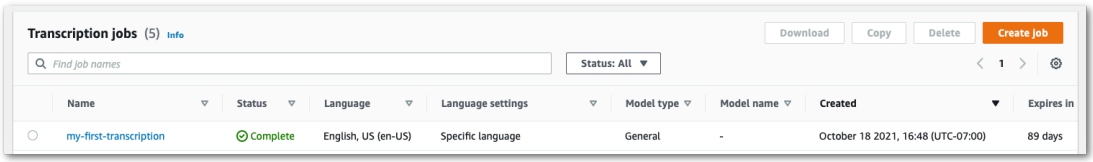
[次へ] を選択します。

3. 「ジョブを設定」ページで必要なオプションを選択します。文字起こしで カスタム語彙 または カスタム言語モデル を使用する場合は、文字起こしジョブを開始する前に、これらを作成する必要があります。



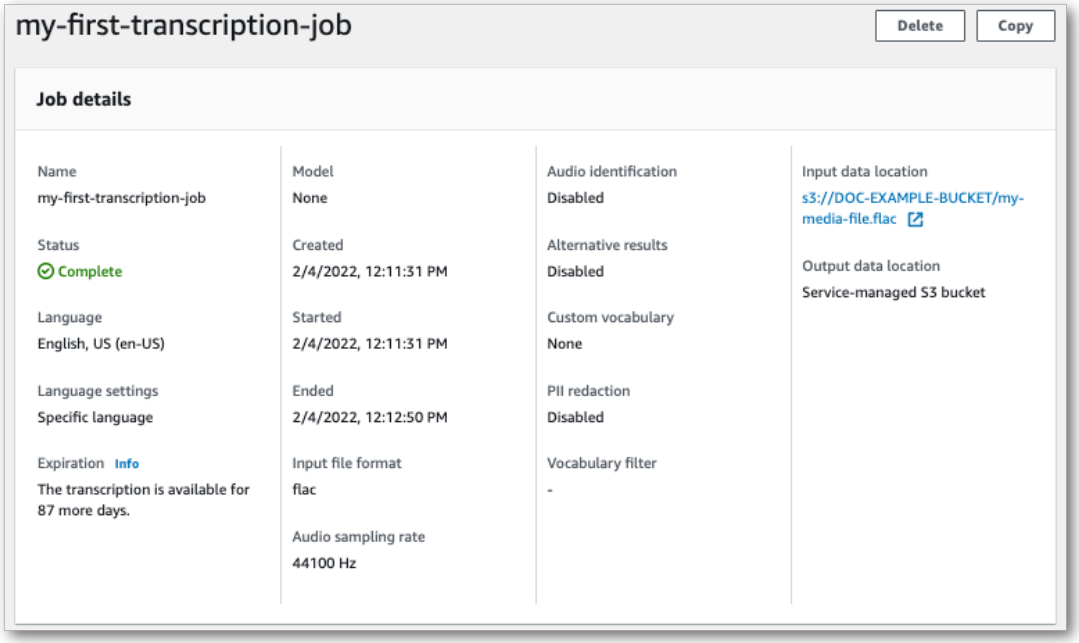
[ジョブの作成] を選択します。

4. これで、「文字起こしジョブ」ページに移動しました。ここでは、文字起こしジョブのステータスを確認できます。完了したら、[文字起こし] を選択します。



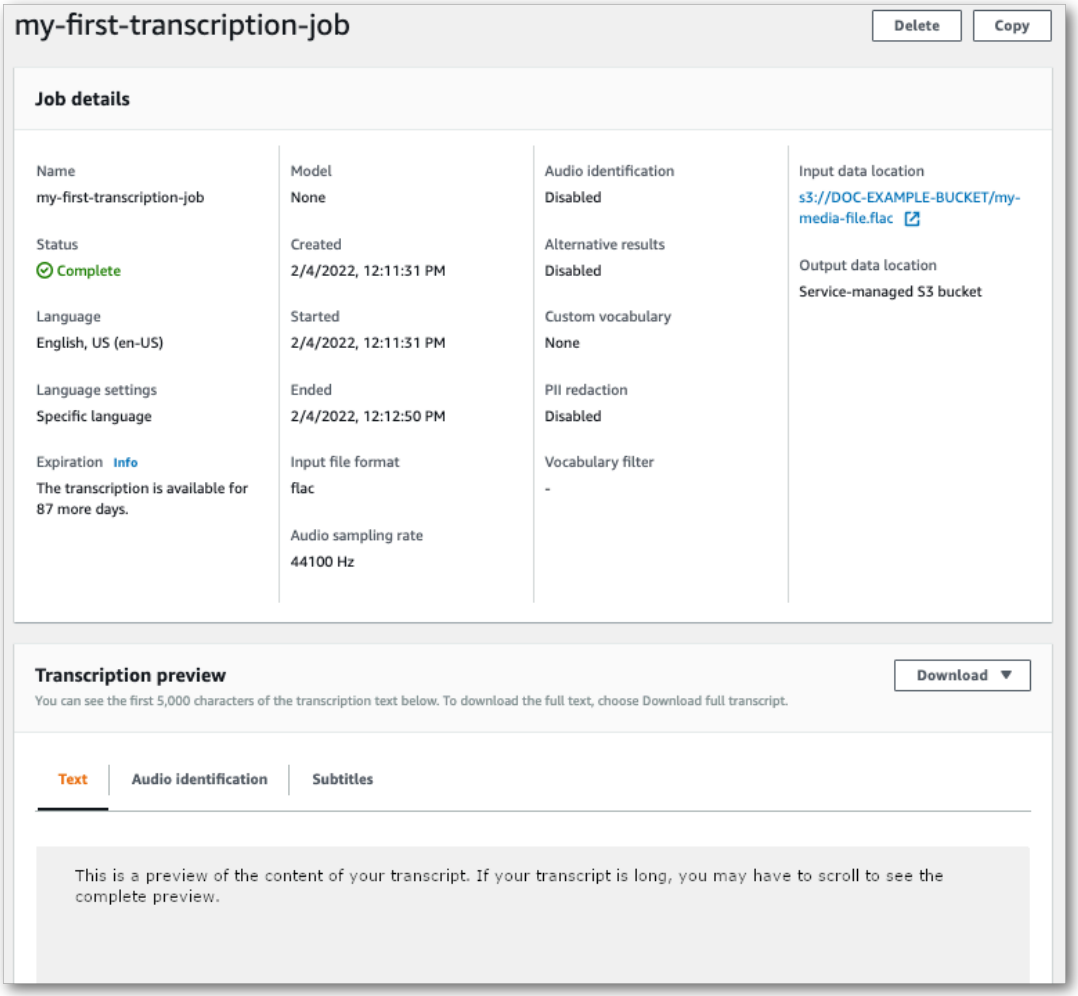
5. これで、文字起こしの「ジョブ詳細」ページが表示されます。ここには、文字起こしジョブの設定時に指定したすべてのオプションが表示されます。

   トランスクリプトを表示するには、出力データの場所の下の右側列にあるリンクされたファイルパスを選択します。これにより、指定した Amazon S3 出力フォルダに移動します。出力ファイルを選択します。このファイルには.json 拡張子が付いています。



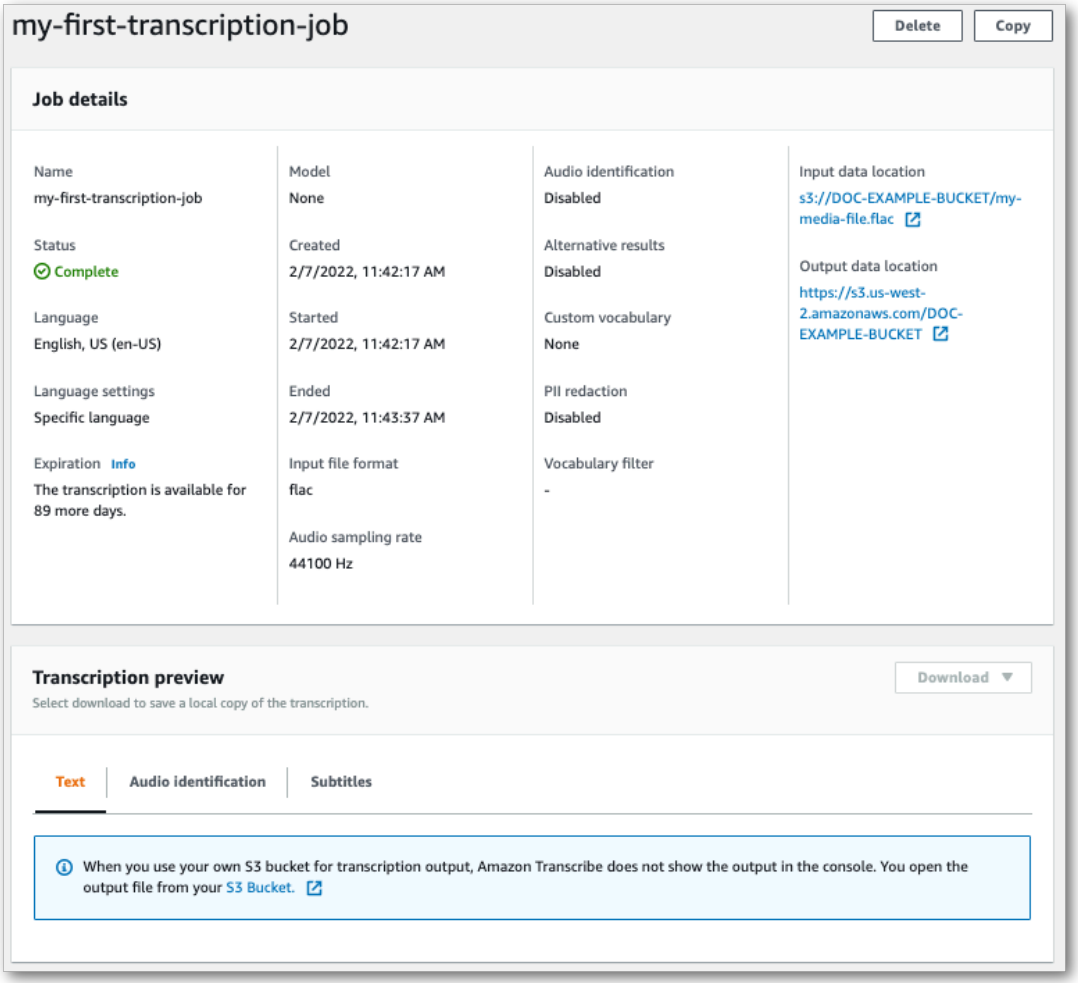
6. トランスクリプトのダウンロード方法は、サービスマネージド Amazon S3 バケットと独自の Amazon S3 バケットのどちらを選択するかによって異なります。

   a. サービスマネージド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に「文字起こしプレビュー」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。

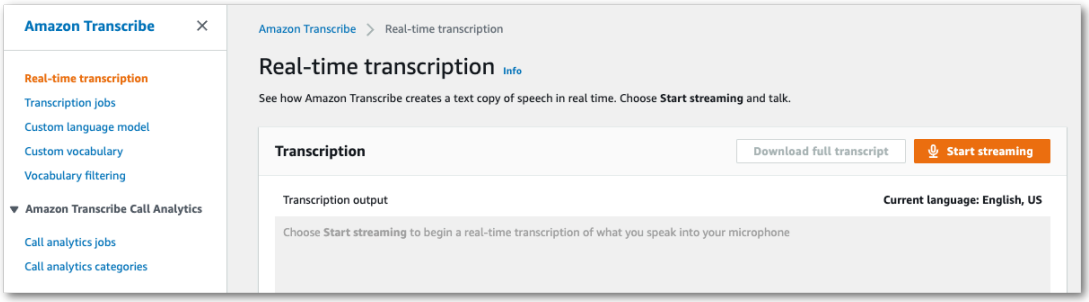
b. 独自の Amazon S3 バケットを選択した場合、文字起こしジョブの情報ページの文字起こしプレビューペインにテキストは表示されません。代わりに、選択した Amazon S3 バケットへのリンクを含む青い情報ボックスが表示されます。



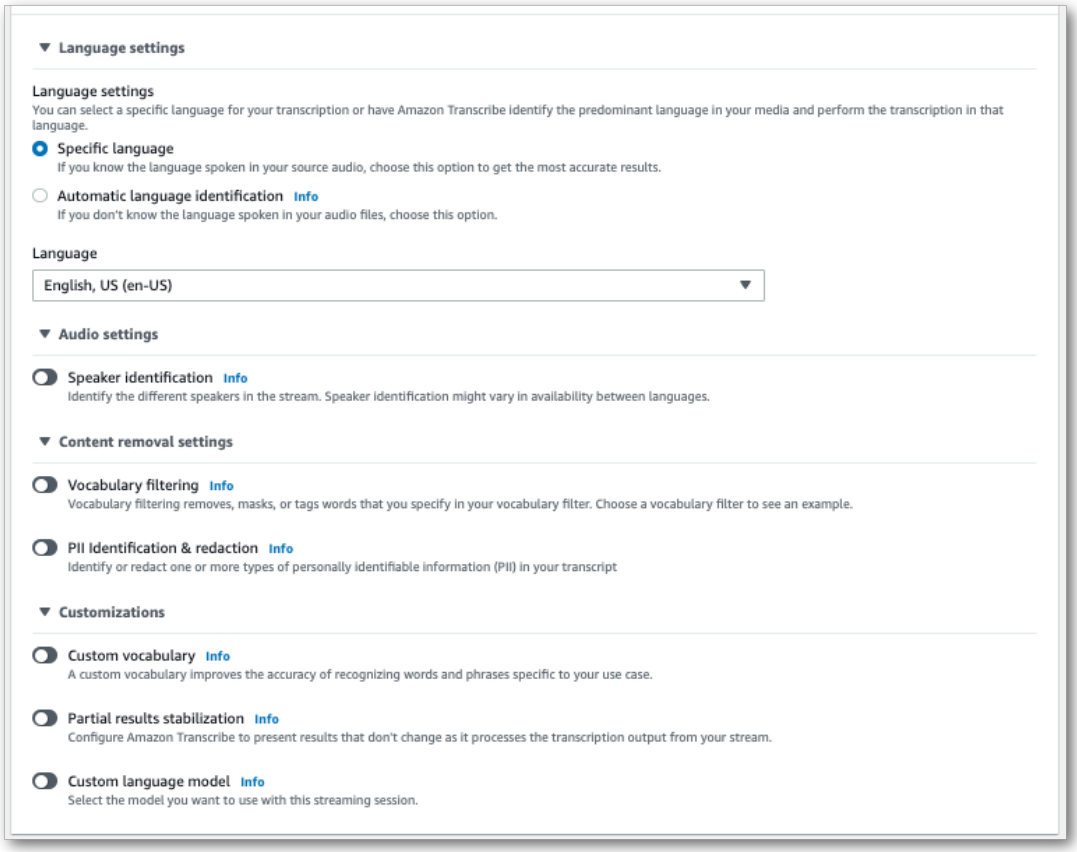
トランスクリプトにアクセスするには、ジョブの詳細ペインの出力データの場所にあるリンク、またはトランスクリプトプレビューペインの青い情報ボックス内の S3 バケットリンク Amazon S3 を使用して、指定されたバケットに移動します。

ストリーミング文字起こし

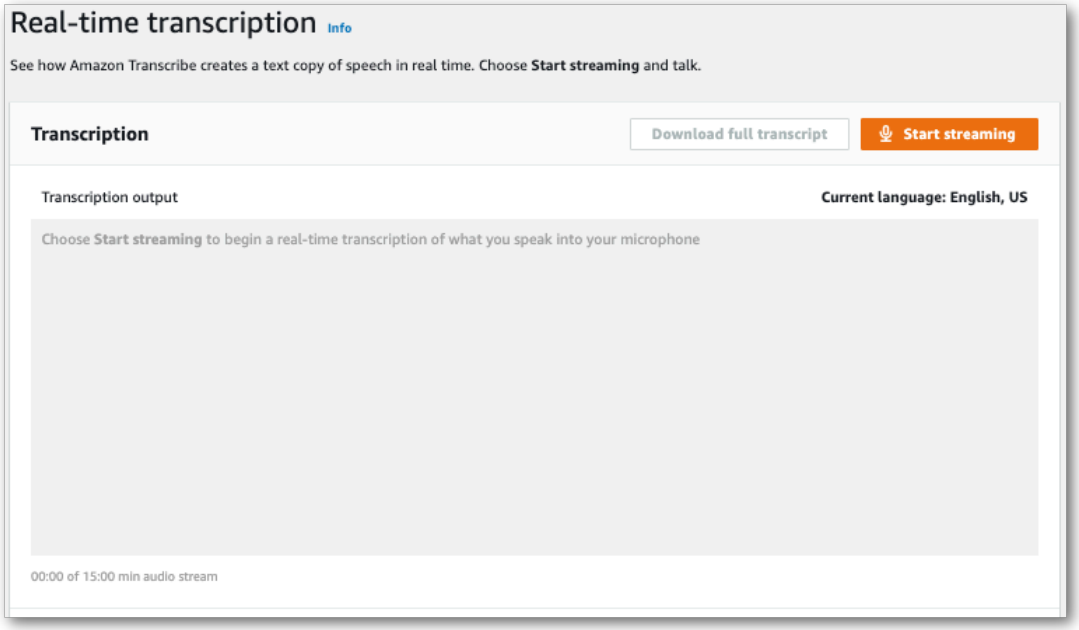
1. AWS マネジメントコンソール から、左のナビゲーションペインで [リアルタイム文字起こし] を選択します。これにより、メインのストリーミングページに移動し、ストリームを開始する前に、オプションを選択できます。



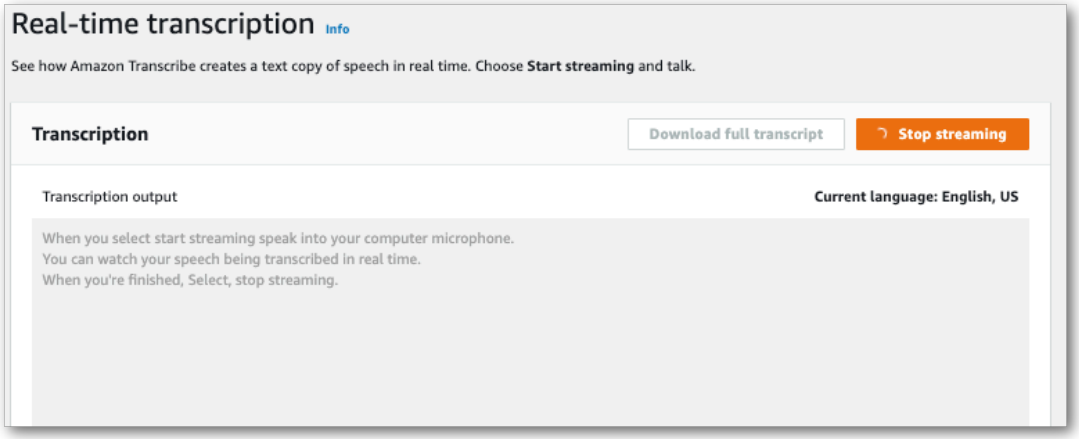
2. 文字起こし出力ボックスの下には、さまざまな言語と音声の設定を選択するオプションがあります。



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、コンピュータのマイクに向かって話し始めます。音声の文字起こしをリアルタイムで確認できます。



4. 終了したら、[ストリーミングを停止する] を選択します。



[すべてのトランスクリプトをダウンロード] を選択すると、トランスクリプトをダウンロードできるようになりました。

での文字起こし AWS CLI

を使用して文字起こ AWS CLI しを開始する場合、すべてのコマンドを CLI レベルで実行できます。または、使用したいコマンドを実行し、その後に AWS リージョン とリクエストボディを含む JSON ファイルの場所を指定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ています。

AWS CLI はストリーミング文字起こしをサポートしていません。

続行する前に、以下の点を確認してください。

- メディアファイルを Amazon S3 バケットにアップロードしました。 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不明な場合は、「最初の Amazon 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。
- AWS CLI をインストールします。

のすべての AWS CLI コマンドは、 Amazon Transcribe AWS CLI コマンドリファレンスで確認できます。

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. ターミナルウィンドウで、次のように入力します。

   ```
   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。

   「\」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 または language-code か identify-language のいずれかを含める必要があります。

   出力場所を指定する場合は、リクエストに output-bucket-name を含めます。指定した出力バケットのサブフォルダを指定する場合は、output-key も含めます。

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

output-bucket-name を使用して出力バケットを指定しない場合、 Amazon Transcribe は文字起こし出力はサービスマネージドバケットに配置されます。サービスマネージド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 は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://amzn-s3-demo-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"
    }
}
```

TranscriptionJobStatus が IN_PROGRESS から COMPLETED に変更されれば、文字起こしジョブは成功です。更新された TranscriptionJobStatus を確認するには、次のセクションで説明するように get-transcription-job または list-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取得。

文字起こしジョブに関する情報を取得するには、get-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、ジョブ AWS リージョン がある 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 は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://amzn-s3-demo-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
        }
    }
}
```

文字起こし出力に独自の Amazon S3 バケットを選択した場合、このバケットは と表示されますTranscriptFileUri。サービスマネージドバケットを選択した場合、一時的な URI が表示されます。この URI を使用してトランスクリプトをダウンロードします。

> **Note**
>
> サービスマネージド Amazon S3 バケットの一時的な URIs は 15 分間のみ有効です。URI の使用中に AccesDenied エラーが発生した場合は、get-transcription-job リクエストをもう一度実行して新しい一時的な URI を取得してください。

文字起こしジョブの一覧表示

特定の 内のすべての文字起こしジョブを一覧表示するには 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 AWS リージョン が配置されている のみです。

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

Amazon Transcribe は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、ジョブ AWS リージョン がある 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。ジョブがリストに表示されなくなります。

AWS SDKs

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ではサポートされてい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、「データの入力および出力」を参照してください。

使用可能なすべての AWS SDKs「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 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。
> また、SDK コードサンプルは、次の GitHub リポジトリにもあります。
>
> - 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不明な場合は、「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://amzn-s3-demo-bucket/my-input-files/my-media-
file.flac")
                .build();

        String outputS3BucketName = "s3://amzn-s3-demo-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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  OutputBucketName: "amzn-s3-demo-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://amzn-s3-demo-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 でのこのサービスの使用

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なる 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	AWS Tools for PowerShell コード例
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] リンクから、コードの例をリクエストしてください。

HTTP または WebSocket による文字起こし

Amazon Transcribe は、バッチ (HTTP/1.1) 文字起こしとストリーミング (HTTP/2) 文字起こしの両方で HTTP をサポートします。WebSocket はストリーミング文字起こしに対応しています。

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。

HTTP と WebSocket はどちらも、 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、AWS 「 API リクエストの署名」を参照してください。

バッチ文字起こし

以下のヘッダーを使用して、バッチ HTTP リクエストを行うことができます。

- ホスト
- x-amz-target
- content-type
- x-amz-content-sha256
- x-amz-date
- 承認

`StartTranscriptionJob` リクエストの例を以下に示します。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
    },
    "OutputBucketName": "amzn-s3-demo-bucket",
    "OutputKey": "my-output-files/"
}
```

追加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のパラメータは、「共通パラメータ」セクションに記載されています。その他の署名要素は、AWS 署名バージョン 4 リクエストの要素で詳しく説明されています。

ストリーミング文字起こし

HTTP/2 と WebSocket を使ったストリーミング文字起こしは、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 セクションを確認することをおすすめします。

これらの方法について詳しくは、「HTTP/2 ストリームの設定」または「WebSocket ストリームの設定」を参照してください。

> **Note**
>
> ストリーミング文字起こしには SDK を使用することを強くおすすめします。サポートされている SDK のリストについては、「サポートされているプログラミング言語」を参照してください。

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。バッチではなくストリーミングを使用する利点には、アプリケーションでのリアルタイムの音声テキスト変換機能や、文字起こしに要する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
- WebSocket
- AWS マネジメントコンソール

でストリーミングオーディオを文字起こしするには AWS マネジメントコンソール、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、GitHub の「AWS サンプルリポジトリ」を参照してください。

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。

LimitExceededException エラーの処理

他の分散システムと同様に、 Amazon Transcribe にはリソースの過剰消費を検出し、それに応じて対応する保護メカニズムがあります。これらのメカニズムのいずれかがトリガーされると、`LimitExceededException` エラーが発生する可能性があります。このエラーには 3 つの異なる原因があります。

同時ストリームサービスクォータを超えました

これは最も一般的な原因です。これは、同時ストリームサービスクォータを超えたときに発生します。このエラーを解決するには、エクスポネンシャルバックオフで再試行します。この制限に一貫して達した場合は、Service Service Quotasクォータの引き上げをリクエストします。AWS サポートセンターに連絡してサポートを依頼することもできます。再試行戦略の詳細については、 SDK およびツールリファレンスガイドの「再試行動作」を参照してください。 AWS SDKs

最大セッション期間を超えました

このエラーは、ストリームがセッションの最大許容期間を超えた場合に発生します。これは増やすことのできない厳格な上限です。文字起こしを続行するには、新しいストリーミングセッションを開始します。

同時ストリームの数が速すぎる

これはまれな原因です。これは、ロードテスト中など、同時ストリームの数が速すぎる場合に発生する可能性があります。これは、調整可能なクォータを持たないシステムレベルの保護メカニズムです。このエラーを解決するには、エクスポネンシャルバックオフで再試行し、同時ストリームの数を徐々に増やします。再試行戦略の詳細については、 SDK およびツールリファレンスガイドの「再試行動作」を参照してください。 AWS SDKs AWS re:Post にアクセスするか、AWS プレミアムサポートに連絡することもできます。

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、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 マネジメントコンソール 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。このオブジェクトブロックには IsPartial フィールドもあります。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 The The Amazon. The Amazon is The Amazon is the large The Amazon is the largest The Amazon is the largest rainfall.	0.515 1.015 1.515 2.015 2.515 3.015 3.515 4.015 4.515 5.015	true true true true true true true true true true

「文字起こし」:	「EndTime」:	「IsPartial」:
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.	5.515 6.015 6.335 6.335	true true true false

部分結果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 でストリームを直接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 や WebSocket を直接使用するのではなく、SDK を使用することを強くおすすめします。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。

HTTP/2 ストリームの設定

で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、 Amazon Transcribe がデータフレームに署名するためにシード署名として使用する認可ヘッダー内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。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "myTranscribeHttp2Policy",
            "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`。
- 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: 最初にその入力の 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: データフレーム用に計算した文字列。

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 ストリームの設定

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、 Amazon Transcribe がデータフレームに署名するためのシード署名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。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "myTranscribeWebsocketPolicy",
            "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 に対して作成したも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。

```
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、 Amazon Transcribeへのリクエストを行います。

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

   以下は、 からのレスポンスです 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音声フレームの送信を開始できます。それぞれイベントストリームエンコーディング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、 Amazon Transcribe 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 フレームに対応します。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した後、 はクローズフレーム Amazon Transcribe 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ド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

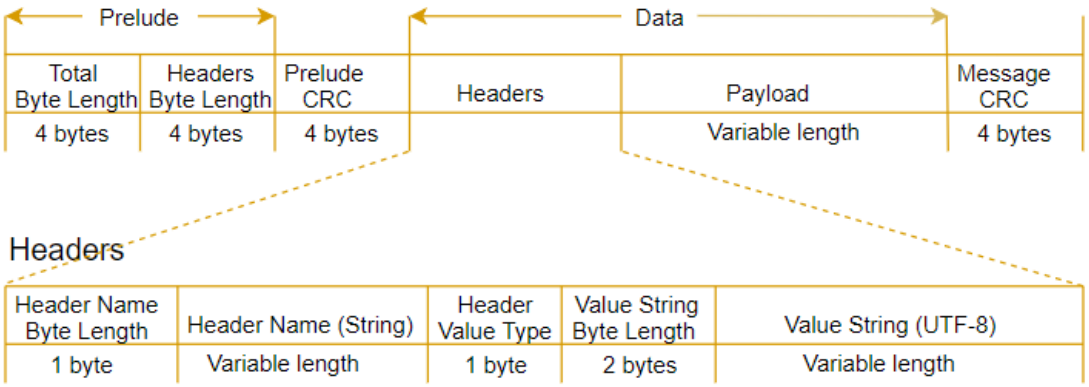
データセクションの構成要素は以下のとおりです。

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 はバッファの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文字起こしストリーミングの認可フレームです。 Amazon Transcribe では、リクエストのデータフレーム用に認可ヘッダーのチェーンを生成するためのシードとして、認可ヘッダーの値を使用します。

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。

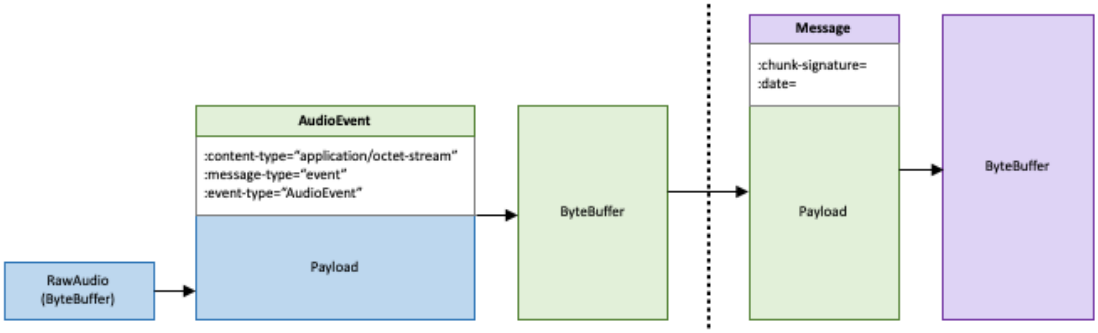
- ヘッダー名のバイト長: ヘッダー名のバイトの長さ。
- ヘッダー名: ヘッダータイプを示すヘッダの名前。有効な値については、次のフレームの説明を参照してください。
- ヘッダー値のタイプ: ヘッダー値を示す数値。以下の表に、ヘッダーに指定できる値と、ヘッダーが示す値を示します。
  - 0 - TRUE
  - 1 - FALSE
  - 2 - BYTE
  - 3 - SHORT
  - 4 - INTEGER
  - 5 - LONG
  - 6 - BYTE ARRAY
  - 7 – STRING
  - 8 - TIMESTAMP
  - 9 - UUID
- 値の文字列のバイ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


Message
:chunk-signature=
:date=
AudioEvent
:content-type="application/octet-stream"
:message-type="event"
:event-type="AudioEvent"
ByteBuffer
Payload
ByteBuffer
RawAudio
(ByteBuffer)
Payload
ByteBuffer

ジョブキューイング

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リクエストを送信できます。ジョブキューイングを使用しない場合、同時実行可能なリクエストのクォータに達すると、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。

ジョブキューイングは、文字起こしジョブと通話後分析ジョブリクエストの両方のオプションです。

ジョブキューイングを有効にすると、 は制限を超えるすべてのリクエストを含むキュー Amazon Transcribe を作成します。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。

キューに最大 10,000 件のジョブを追加できます。この制限を超えた場合、`LimitExceededConcurrentJobException` エラーが発生します。最適なパフォーマンスを維持するために、 はクォータの最大 90% (帯域幅比 0.9) Amazon Transcribe のみを使用してキューに入れられたジョブを処理します。これらはデフォルト値であり、リクエストに応じて増やすことができます。

> **Tip**
>
> Amazon Transcribe リソースのデフォルト制限とクォータのリストについては、 AWS 全般のリファレンスを参照してください。これらのデフォルトの一部は、リクエストに応じて増やすことができます。

ジョブキューイングを有効にしても、同時実行リクエストのクォータを超えないようにすると、すべてのリクエストが同時に処理されます。

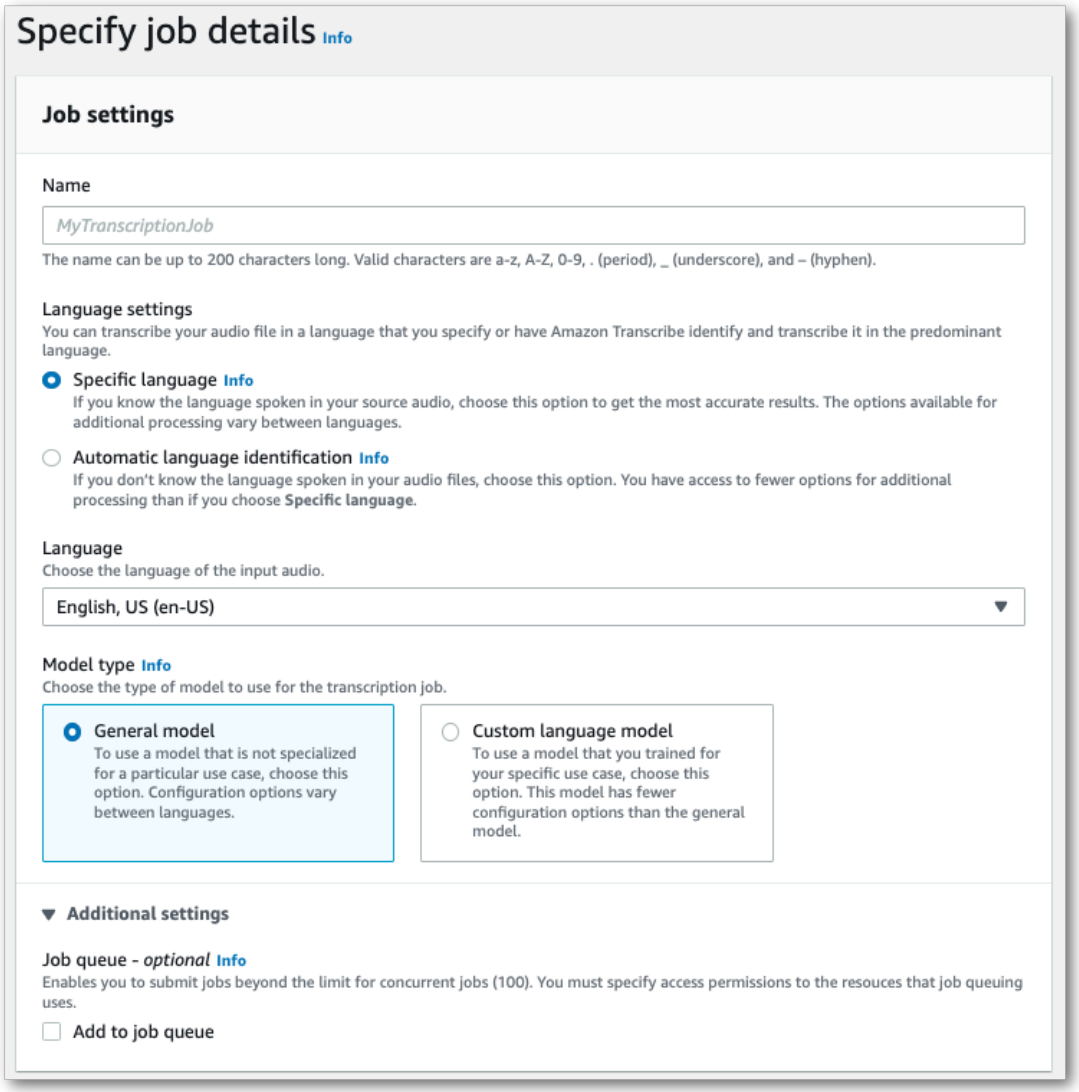
ジョブキューイングの有効化

AWS マネジメントコンソール、AWS CLI、または AWS SDK を使用して、ジョブキューイングを有効にできます。例については以下を参照してください。

AWS マネジメントコンソール

1. AWS マネジメントコンソール 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イングを有効にできます。



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 job-execution-settings パラメータ、AllowDeferredExecution サブパラメータを使用します。AllowDeferredExecution をリクエストに含める場合は、DataAccessRoleArn も含める必要があることに注意してください。

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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
--language-code en-US \
--job-execution-settings
 AllowDeferredExecution=true,DataAccessRoleArn=arn:aws:iam::111122223333:role/
ExampleRole
```

start-transcription-job コマンド、およびキューイングを有効にするリクエストボディを使用した別の例になります。

```
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含まれています。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-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、 を使用して AWS SDK for Python (Boto3) 、start_transcription_job メソッドの AllowDeferredExecution引数を使用してジョブキューイングを有効にします。AllowDeferredExecution をリクエストに含める場合は、DataAccessRoleArn も含める必要があることに注意してください。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。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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)
```

キューに入れられたジョブの進行状況は、 AWS マネジメントコンソール または GetTranscriptionJobリクエストを送信することで表示できます。ジョブがキューに入れられると、Status は QUEUED になります。ジョブが処理を開始するとステータスは IN_PROGRESS に変わり、処理が終了すると、COMPLETED または FAILED 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、タグを使用して請求書を動的カテゴリに分割できます。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。

では Amazon Transcribe、次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
- 通話分析通話後文字起こしジョブを開始する
- カスタム語彙
- カスタム医療語彙
- カスタム語彙フィルター
- 通話分析カテゴリ
- カスタム言語モデル

タグキーは最大 128 文字、タグ値は最大 256 文字です。どちらも大文字と小文字が区別されます。はリソースごとに最大 50 個のタグ Amazon Transcribe をサポートします。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。はシステム生成タグにこのプレフィックス AWS を予約`aws:`するため、タグは で始めることができないことに注意してくださ

い。aws:* タグを追加、変更、削除することはできません。このタグは、リソースあたりの上限タグ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。

ベストプラクティスを含むタグ付けの詳細については、「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制御

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。タグベースのアクセスコントロールの場合、 IAM ポリシーの条件要素にタグ情報を指定します。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: Amazon Transcribe リソースに割り当てたタグに基づいて、リソースへのアクセスを制御します。
  - 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に追加、変更、または削除できるタグを指定します。
- 認可プロセス: 認可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、「タグに基づく文字起こしジョブの表示」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。

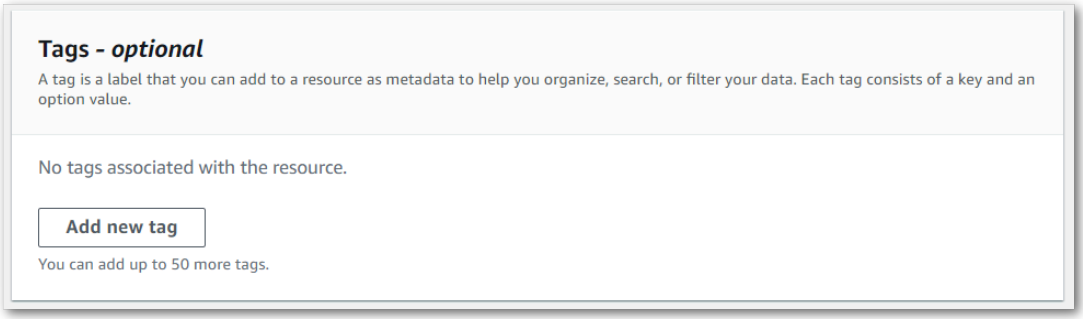
Amazon Transcribe リソースへのタグの追加

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。既存の作成 * と 開始 * の API を使用すると、文字起こしリクエストにタグを追加できます。

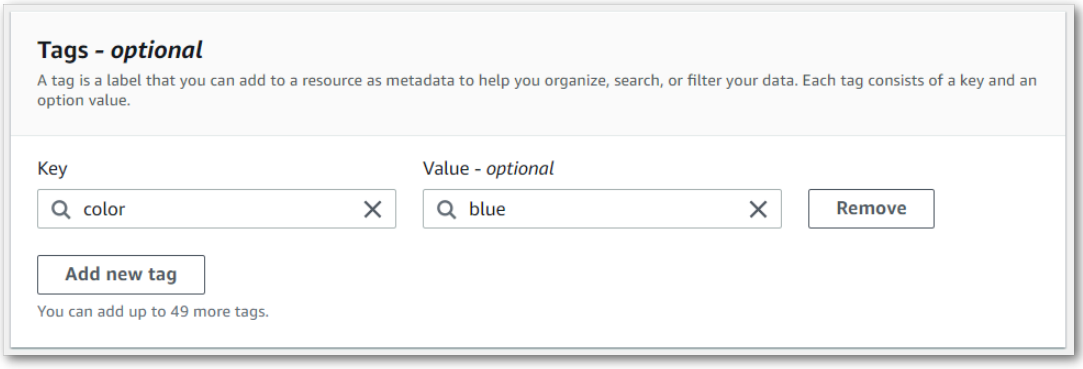
AWS マネジメントコンソール、AWS CLI、または AWS SDK を使用してタグの追加、変更、または削除を行うことができます。例については、次を参照してください。

AWS マネジメントコンソール

1. AWS マネジメントコンソール 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の下部までスクロールして、タグ - オプションボックスを見つけたら、[新しいタグを追加] を選択します。



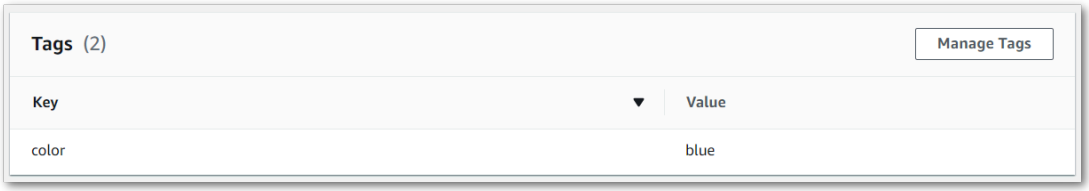
4. キー フィールドと、オプションで 値 フィールドに情報を入力します。



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。

[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。

6. 文字起こし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] を選択します。



AWS CLI

この例では、start-transcription-job コマンドと Tags パラメータを使用します。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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-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、変換ジョブの開始 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 には、次のリクエストボディが入含まれています。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-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 AWS SDK for Python (Boto3) 、 を使用して、start_transcription_job メソッドの Tags引数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。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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)

スピーカー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スピーカーを区別できます。 は最大 30 人の一意のスピーカーを区別し、一意の値 (spk_0 から) で各一意のスピーカーのテキストにラベルを付ける Amazon Transcribe ことができますspk_29。

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"
                },
```

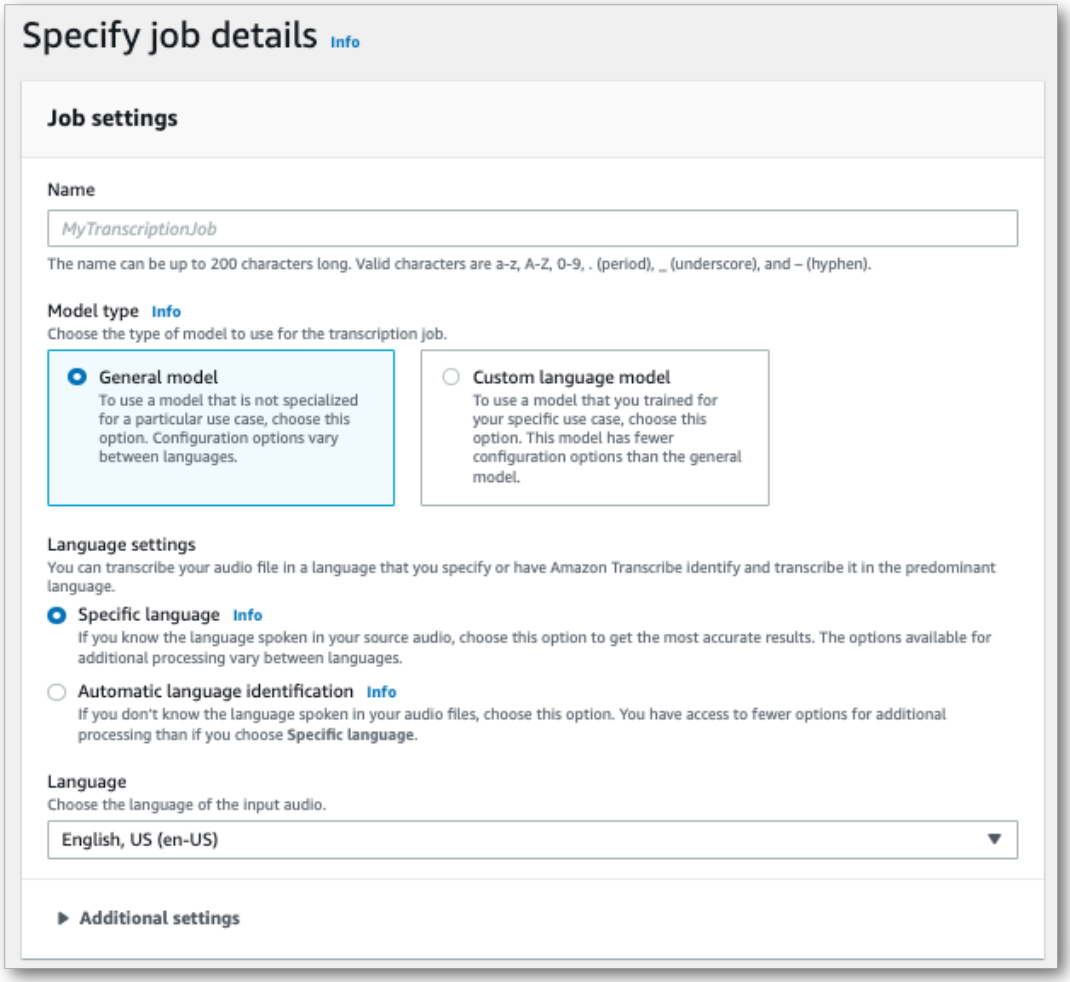
スピーカーパーティショニングを使用した完全な文字起こしの例 (2 人の話者の場合) を見るには、「ダイアライゼーション出力の例 (バッチ)」を参照してください。

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。

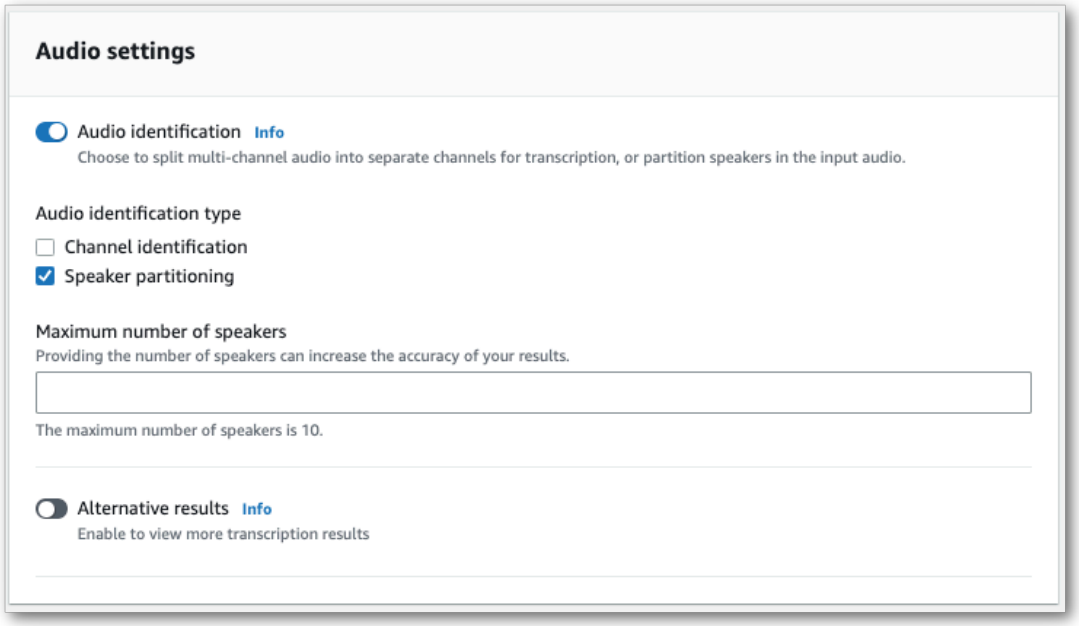
AWS マネジメントコンソール

1. AWS マネジメントコンソール 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。

   話者のパーティショニングを有効にするには、[オーディオ設定] で [音声識別] を選択します。次に、[話者のパーティショニング] を選択し、話者の数を指定します。



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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
--language-code en-US \
--settings ShowSpeakerLabels=true,MaxSpeakerLabels=3
```

別の例として、start-transcription-job コマンド、および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ボディを使用します。

```
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 には、次のリクエストボディが含まれています。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-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、「StartTranscriptionJob」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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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)
```

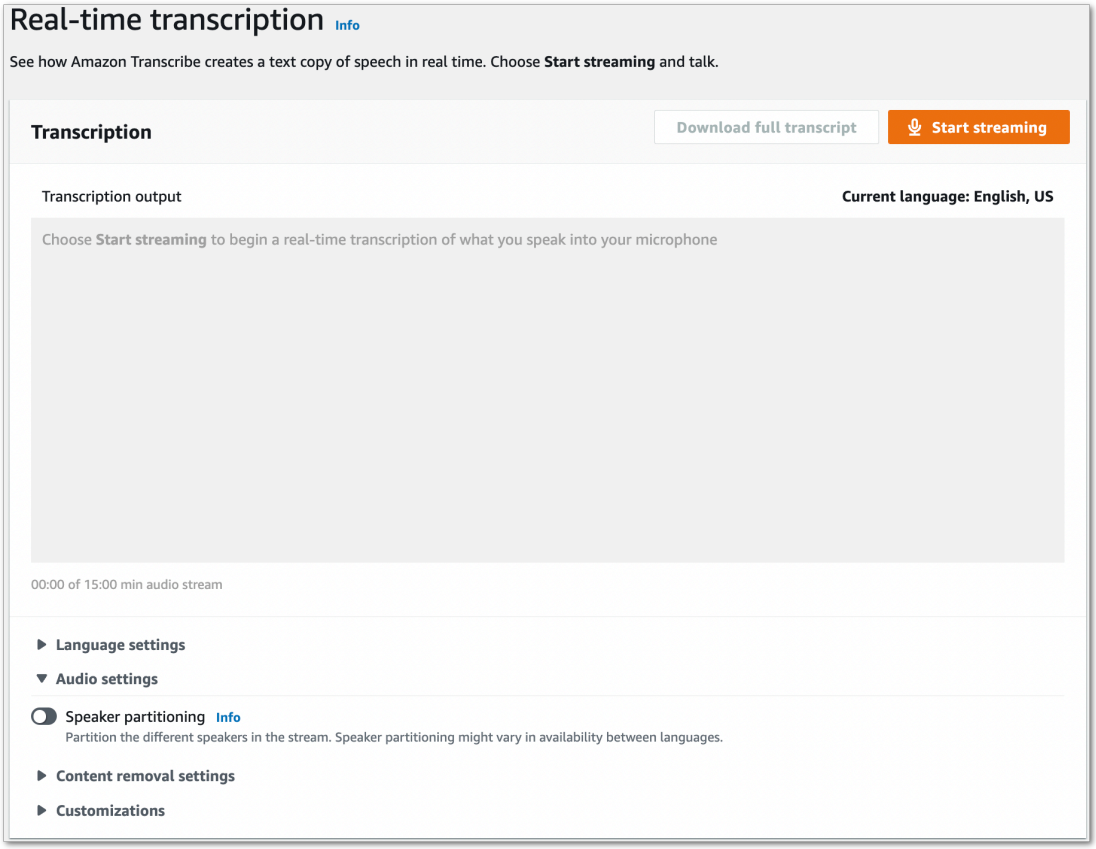
```
print(status)
```

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

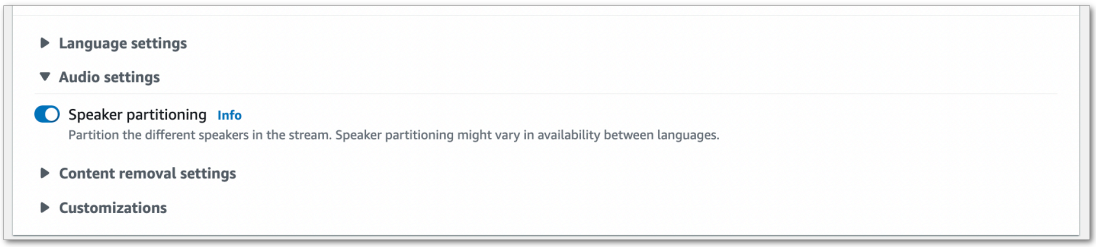
ストリーミング文字起こしでスピーカーをパーティション化するには、次の例を参照してください。

ストリーミング文字起こし

1. AWS マネジメントコンソール 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。 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-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 AWS API オペレーションに共通のパラメータは、「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 ストリームの設定。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 AWS API オペレーションに共通のパラメータは、「共通パラメータ」セクションに記載されています。

ダイアライゼーション出力の例 (バッチ)

ダイアライゼーションを有効にしたバッチ文字起こし出力の例を次に示します。

```
{
    "jobName": "my-first-transcription-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "I've been on hold for an hour. Sorry about that."
            }
        ],
        "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": [
        {
            "id": 0,
            "start_time": "4.87",
            "speaker_label": "spk_0",
            "end_time": "5.02",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "I've"
                }
            ],
            "type": "pronunciation"
        },
        {
            "id": 1,
            "start_time": "5.02",
            "speaker_label": "spk_0",
            "end_time": "5.17",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "been"
                }
            ],
            "type": "pronunciation"
        },
        {
            "id": 2,
            "start_time": "5.17",
            "speaker_label": "spk_0",
            "end_time": "5.29",
            "alternatives": [
                {
```

```
                    "confidence": "1.0",
                    "content": "on"
                }
            ],
            "type": "pronunciation"
        },
        {
            "id": 3,
            "start_time": "5.29",
            "speaker_label": "spk_0",
            "end_time": "5.64",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "hold"
                }
            ],
            "type": "pronunciation"
        },
        {
            "id": 4,
            "start_time": "5.64",
            "speaker_label": "spk_0",
            "end_time": "5.84",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "for"
                }
            ],
            "type": "pronunciation"
        },
        {
            "id": 5,
            "start_time": "6.11",
            "speaker_label": "spk_0",
            "end_time": "6.26",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "an"
                }
            ],
            "type": "pronunciation"
```

```
            },
            {
                "id": 6,
                "start_time": "6.26",
                "speaker_label": "spk_0",
                "end_time": "6.88",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hour"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 7,
                "speaker_label": "spk_0",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            },
            {
                "id": 8,
                "start_time": "8.49",
                "speaker_label": "spk_1",
                "end_time": "8.88",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Sorry"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 9,
                "start_time": "8.88",
                "speaker_label": "spk_1",
                "end_time": "9.05",
                "alternatives": [
```

```
                    {
                        "confidence": "0.902",
                        "content": "about"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 10,
                "start_time": "9.05",
                "speaker_label": "spk_1",
                "end_time": "9.24",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "that"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 11,
                "speaker_label": "spk_1",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ],
        "audio_segments": [
            {
                "id": 0,
                "transcript": "I've been on hold for an hour.",
                "start_time": "4.87",
                "end_time": "6.88",
                "speaker_label": "spk_0",
                "items": [
                    0,
                    1,
                    2,
                    3,
```

```
                    4,
                    5,
                    6,
                    7
                ]
            },
            {
                "id": 1,
                "transcript": "Sorry about that.",
                "start_time": "8.49",
                "end_time": "9.24",
                "speaker_label": "spk_1",
                "items": [
                    8,
                    9,
                    10,
                    11
                ]
            }
        ]
    },
    "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、チャネル識別を使用して、各チャネルから個別に音声を文字起こ Amazon Transcribe しできます。 は現在、3 つ以上のチャネルを持つオーディオをサポートしていません。

トランスクリプトでは、チャネルには ch_0 と ch_1 というラベルが割り当てられます。

チャネル識別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、channel_labels セクションが含まれます。このセクションには、チャネルごとにグループ化された各発話または句読点と、それに関連するチャネルラベル、タイムスタンプ、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
    },
```

あるチャネルにいる人が別のチャネルにいる人と同時に話す場合、各人がお互いに話している間、各チャネルのタイムスタンプが重複することに注意してください。

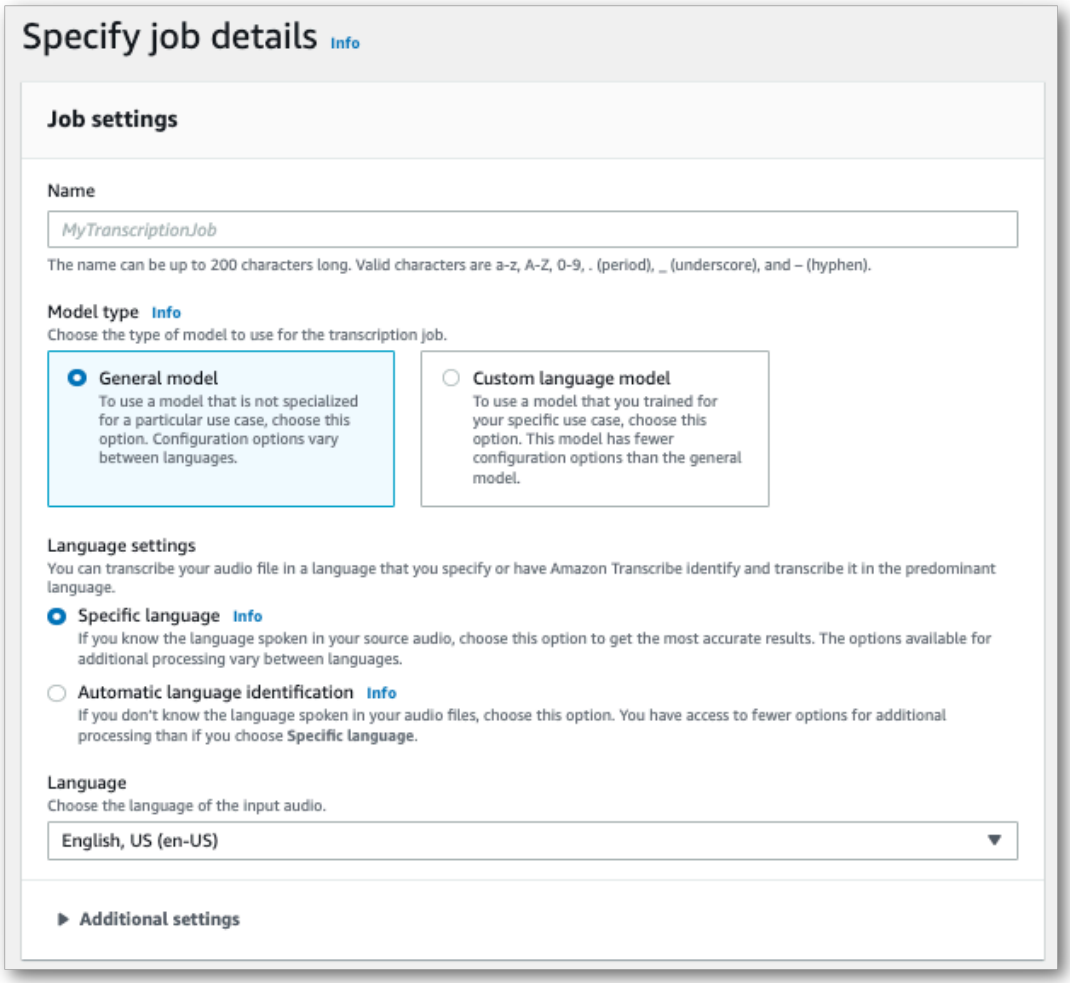
チャネル識別を含むトランスクリプトの完全な例については、「チャネル識別出力の例 (バッチ)」を参照してください。

バッチ文字起こしでのチャネル識別の使用

バッチ文字起こしでチャネルを識別するには、AWS マネジメントコンソール、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。

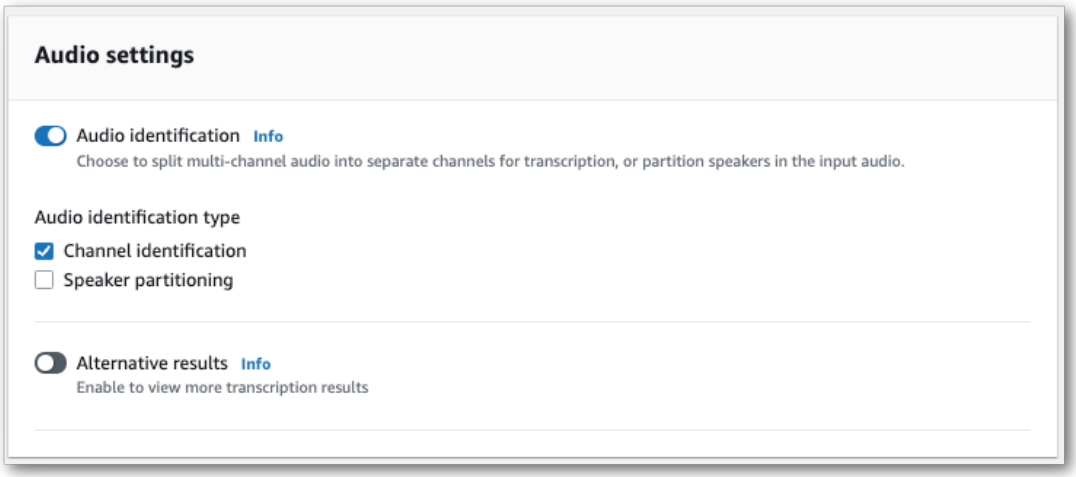
AWS マネジメントコンソール

1. AWS マネジメントコンソール 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 [チャネル識別] を選択します。



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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-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、およびそのジョブとのチャネル識別を可能に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-transcription-job.json
```

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

```
{
    "TranscriptionJobName": "my-first-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
    "OutputBucketName": "amzn-s3-demo-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ネルを識別します。詳細については、「StartTranscriptionJob」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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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)
```

ストリーミング文字起こしでのチャネル識別の使用

ストリーミング文字起こしでチャネルを識別するには、HTTP/2 または WebSocket を使用できます。例については以下を参照してください。

HTTP/2 ストリーム

この例では、文字起こしの出力でチャネルを分離する 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。 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-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のパラメータは、「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ネルを分離するための署名付き 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 ストリームの設定。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のパラメータは、「共通パラメータ」セクションに記載されています。

チャネル識別出力の例 (バッチ)

チャネル識別を有効にしたバッチ文字起こし出力の例を以下に示します。


```
{
    "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": [
            {
                "id": 0,
                "channel_label": "ch_0",
                "start_time": "4.86",
                "end_time": "5.01",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "I've"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 1,
                "channel_label": "ch_0",
                "start_time": "5.01",
                "end_time": "5.16",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "been"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 2,
                "channel_label": "ch_0",
                "start_time": "5.16",
                "end_time": "5.28",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "on"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 3,
                "channel_label": "ch_0",
                "start_time": "5.28",
```

```
                "end_time": "5.62",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hold"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 4,
                "channel_label": "ch_0",
                "start_time": "5.62",
                "end_time": "5.83",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "for"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 5,
                "channel_label": "ch_0",
                "start_time": "6.1",
                "end_time": "6.25",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "an"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 6,
                "channel_label": "ch_0",
                "start_time": "6.25",
                "end_time": "6.87",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hour"
```

```
                }
            ],
            "type": "pronunciation"
        },
        {
            "id": 7,
            "channel_label": "ch_0",
            "alternatives": [
                {
                    "confidence": "0.0",
                    "content": "."
                }
            ],
            "type": "punctuation"
        },
        {
            "id": 8,
            "channel_label": "ch_1",
            "start_time": "8.5",
            "end_time": "8.89",
            "alternatives": [
                {
                    "confidence": "1.0",
                    "content": "Sorry"
                }
            ],
            "type": "pronunciation"
        },
        {
            "id": 9,
            "channel_label": "ch_1",
            "start_time": "8.89",
            "end_time": "9.06",
            "alternatives": [
                {
                    "confidence": "0.9176",
                    "content": "about"
                }
            ],
            "type": "pronunciation"
        },
        {
            "id": 10,
            "channel_label": "ch_1",
```

```
                "start_time": "9.06",
                "end_time": "9.25",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "that"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 11,
                "channel_label": "ch_1",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ],
        "audio_segments": [
            {
                "id": 0,
                "transcript": "I've been on hold for an hour.",
                "start_time": "4.86",
                "end_time": "6.87",
                "channel_label": "ch_0",
                "items": [
                    0,
                    1,
                    2,
                    3,
                    4,
                    5,
                    6,
                    7
                ]
            },
            {
                "id": 1,
                "transcript": "Sorry about that.",
                "start_time": "8.5",
```

```
                "end_time": "9.25",
                "channel_label": "ch_1",
                "items": [
                    8,
                    9,
                    10,
                    11
                ]
            }
        ]
    },
    "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、メディアで話されてい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。言語をより速く識別 Amazon Transcribe できればするほど、ストリームの最初の数秒でデータが失われる変更が少なくなります。

> **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。トランスクリプトの翻訳など Amazon Translate、他の サービスを使用できます。

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 CLI、 AWS マネジメントコンソール、および AWS Python SDK の使用例については、「」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、te-IN、th-TH、tr-TR をサポートしています。

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 、オーディオを正しい言語に一致させるときに指定した言語のみの使用に制限されるため、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、 は指定した言語コードから最も近い言語の一致 Amazon Transcribe 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。たとえば、メディアが米国英語 (en-US) で、Amazon Transcribe 言語コード zh-CN、fr-FR、および を指定した場合de-DE、 Amazon Transcribe はメディアをドイツ語 (de-DE) に一致させ、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。

言語識別を他の Amazon Transcribe 機能と組み合わせる

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、言語識別とコンテンツリダクションを併用する場合、編集が可能な言語は米国英語 (en-US) または米国スペイン語 (es-US) に限定されます。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**
>
> コンテンツリダクションを有効にして自動言語識別を使用しているときに、音声に米国英語 (en-US) または英国スペイン語 (es-US) 以外の言語が含まれている場合は、米国英語または英国スペイン語のコンテンツのみがトランスクリプトで編集されます。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。たとえば、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 である Amazon Transcribe 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」を参照してください。

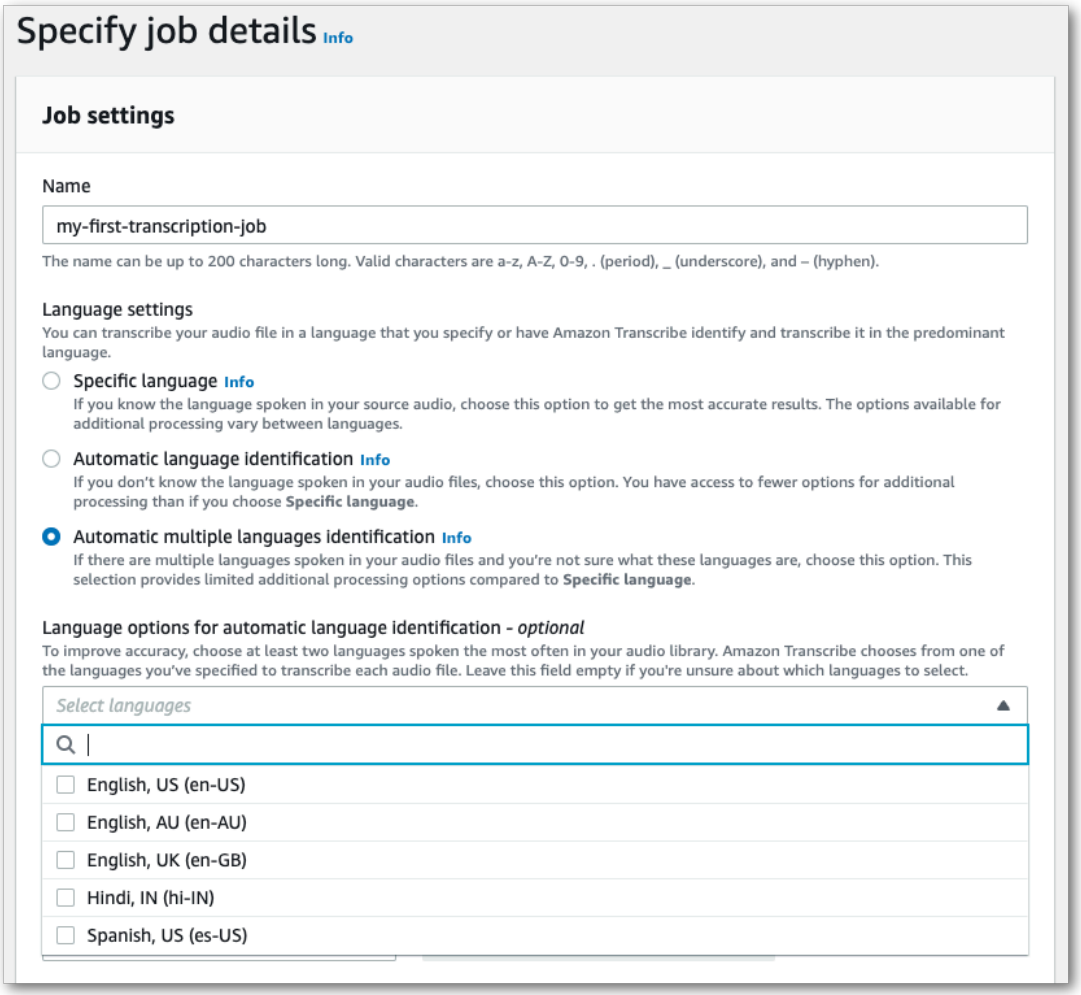
バッチ文字起こしによる言語識別の使用

バッチ文字起こしジョブで、AWS マネジメントコンソール、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。

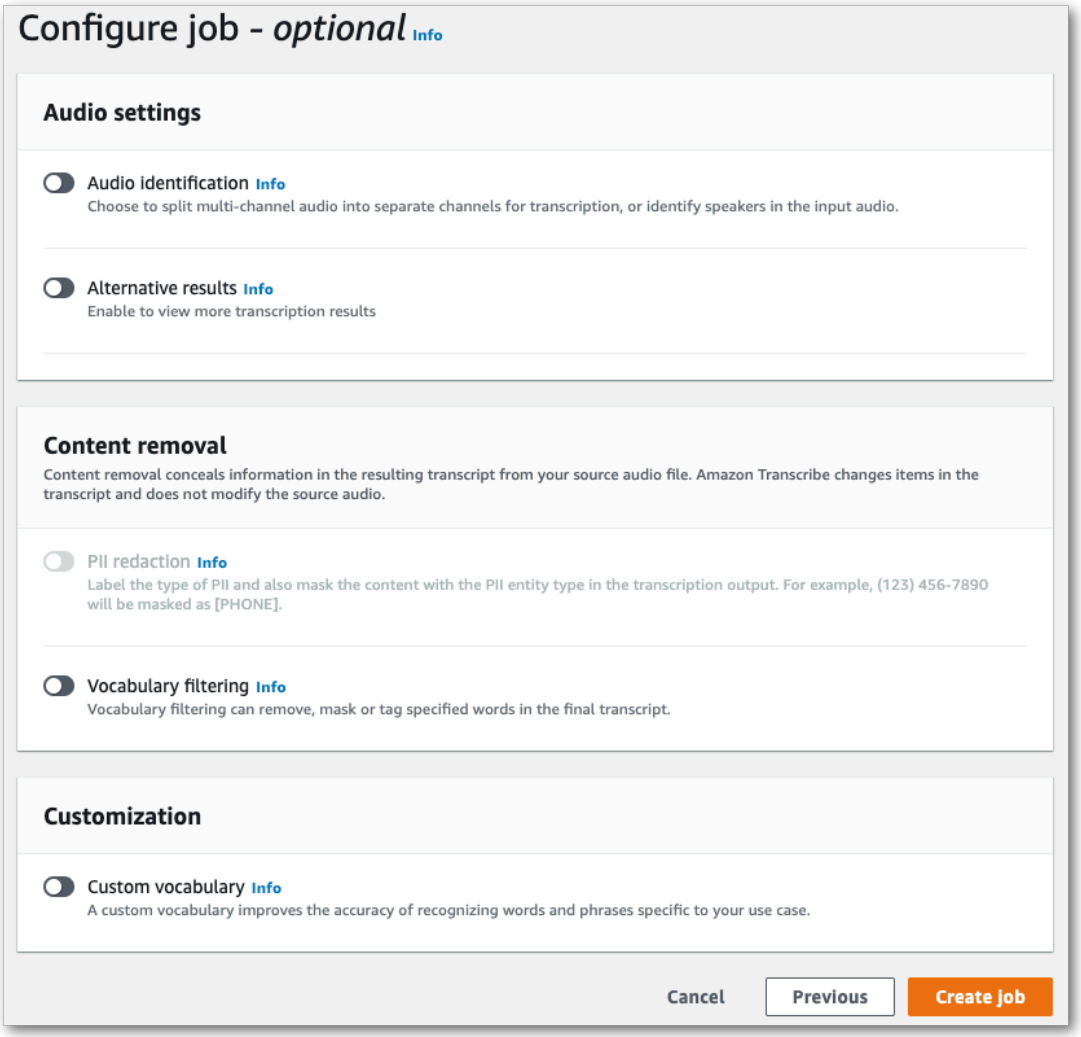
AWS マネジメントコンソール

1. AWS マネジメントコンソール 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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-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、および言語を識別するリクエストボディを使用した別の例になります。

```
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、次のリクエストボディが含まれています。

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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
   },
  "OutputBucketName": "amzn-s3-demo-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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
   },
   "OutputBucketName": "amzn-s3-demo-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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
```

```
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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は、メディアストリームで話されている主要言語を識別できます。 では、言語を識別するために少なくとも 1 秒の音声 Amazon Transcribe 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。トランスクリプトの翻訳など Amazon Transcribe、他の サービスを使用できます。

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**
>
> 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。たとえば、メディアが米国英語 (en-US) で、 Amazon Transcribe 言語コード zh-CN、fr-FR、および を指定した場合de-DE、 Amazon Transcribe はメディアをドイツ語 (de-DE) に一致させ、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。

メディアに 2 つのチャネルが含まれている場合、 Amazon Transcribe は各チャネルで話されている主要言語を識別できます。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。多言語識別では、PCM のみがサポートされています。

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。

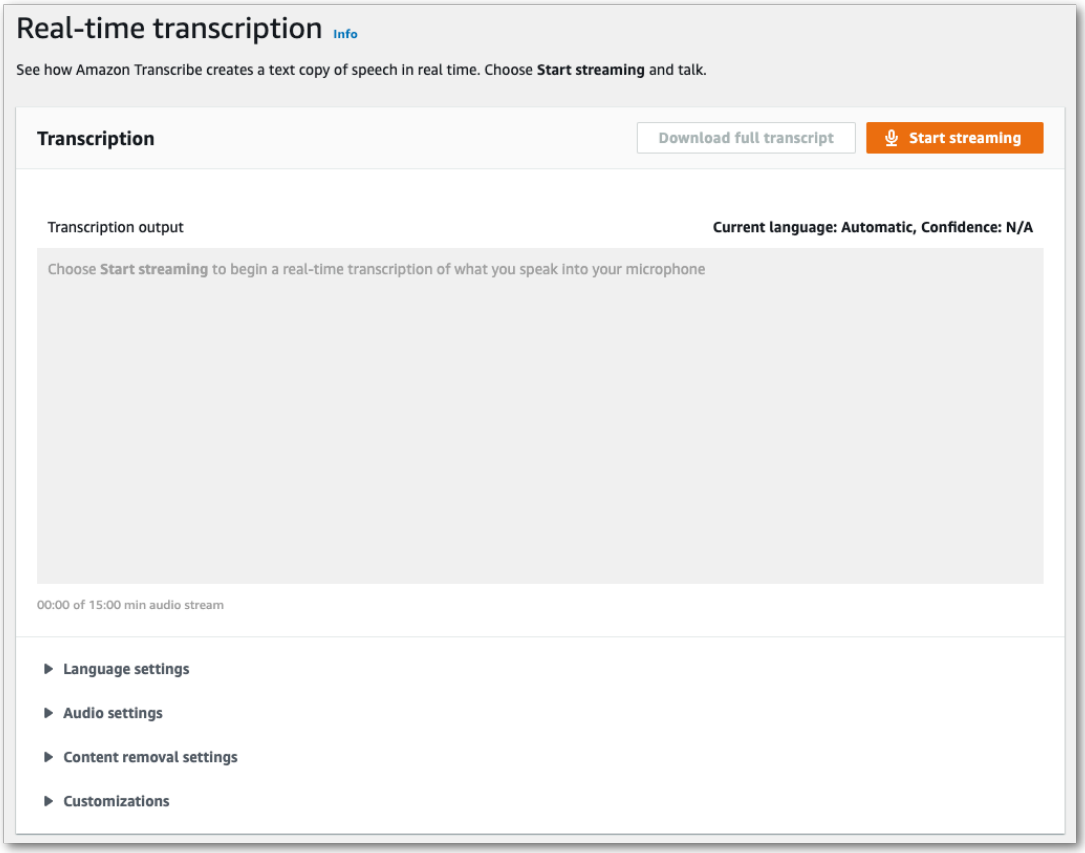
> **Note**
>
> 現在、リダクションとカスタム言語モデルは多言語識別ではサポートされていません。

ストリーミングメディアでの言語識別を使用

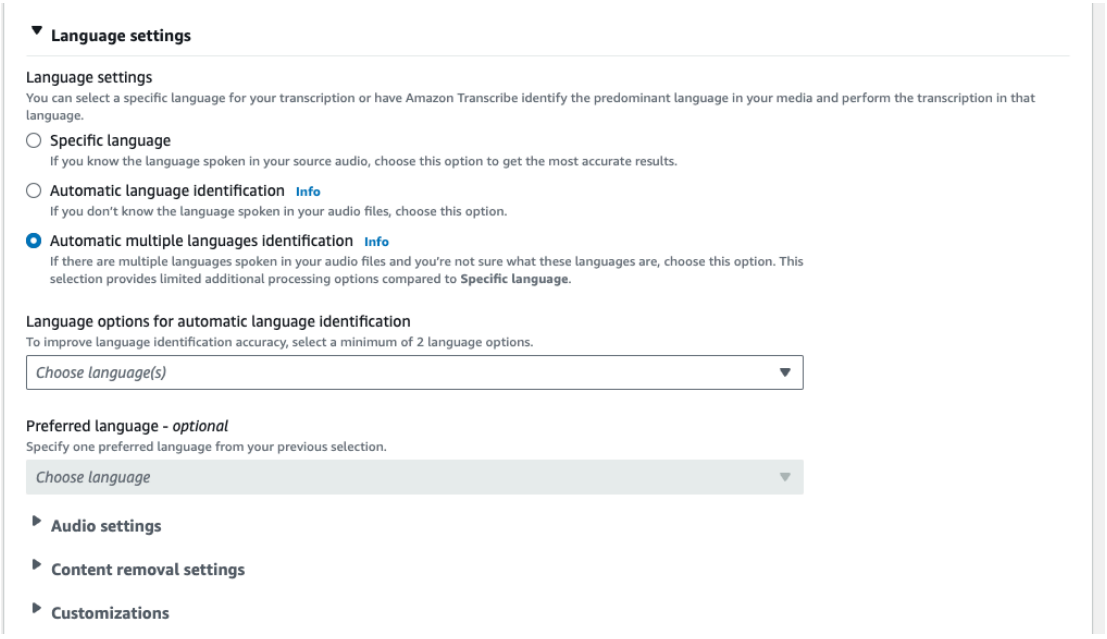
バッチ文字起こしジョブで AWS マネジメントコンソール、HTTP/2、または WebSocket を使用して、自動言語識別を使用できます: 例については、次を参照してください。

AWS マネジメントコンソール

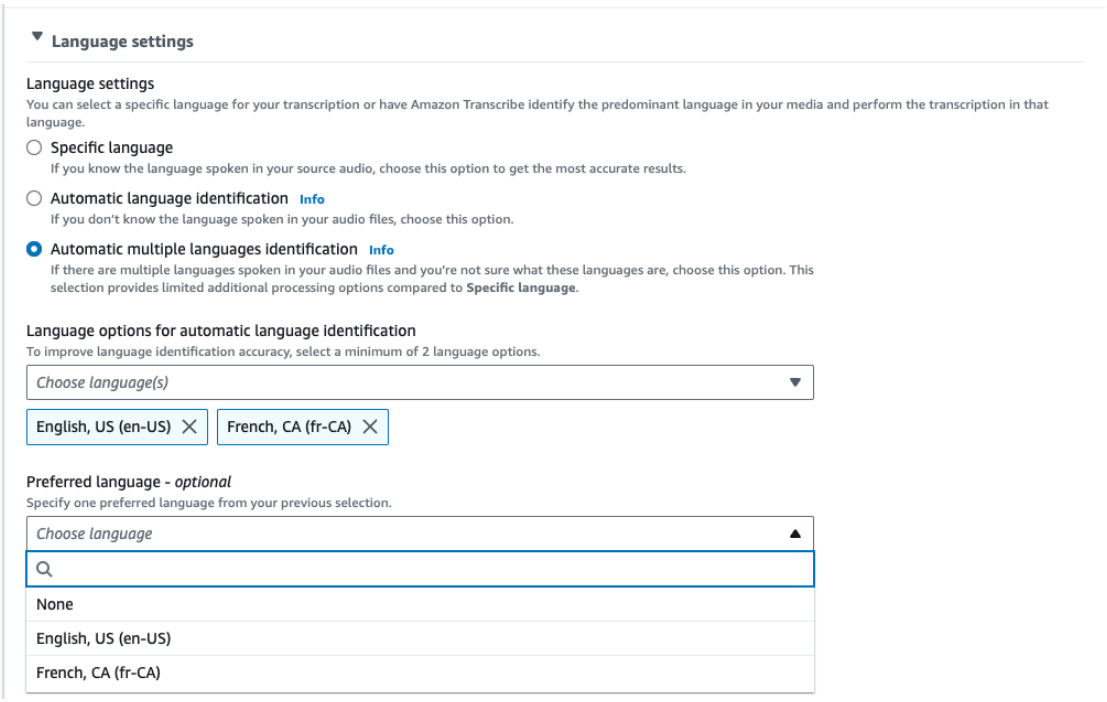
1. AWS マネジメントコンソール にサインインします。
2. ナビゲーションペインで、[リアルタイム文字起こし] を選択します。言語設定にスクロールして、最小化されている場合はこのフィールドを展開します。



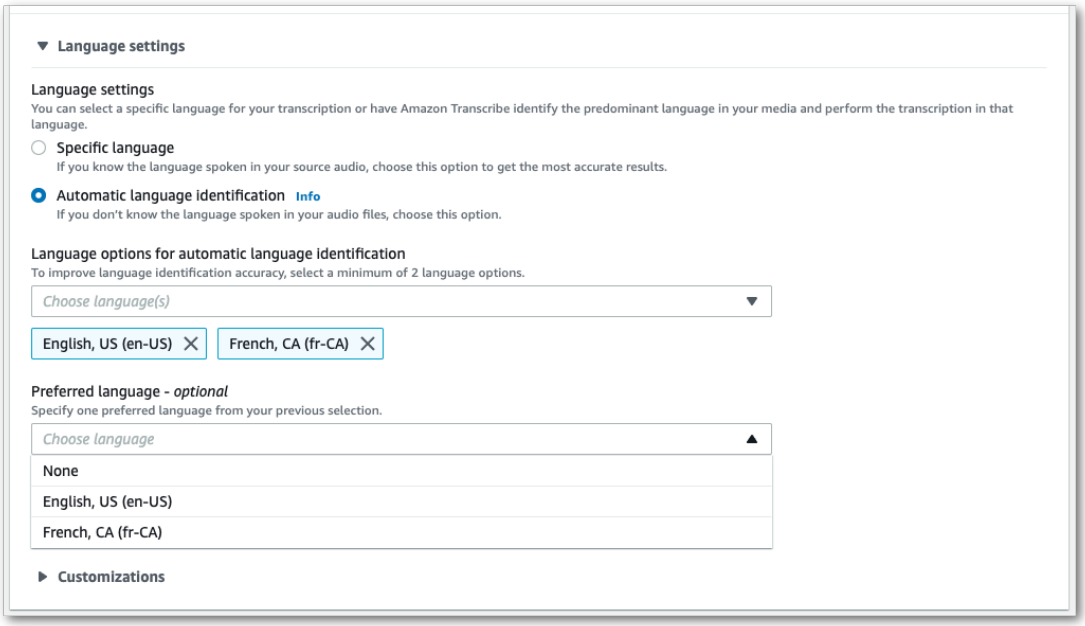
3. [自動言語識別] または [複数言語の自動識別] を選択します。



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 en-GB 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。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。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-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。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。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-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のパラメータは、「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。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 ストリームの設定。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。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 ストリームの設定。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のパラメータは、「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

は音声を Amazon Transcribe 書き起こすと、同じトランスクリプトの異なるバージョンを作成し、各バージョンに信頼スコアを割り当てます。一般的な文字起こしでは、信頼スコアが最も高いバージョンしか得られません。

代替文字起こしを有効にすると、 は信頼レベルが低い他のバージョンの文字起こし Amazon Transcribe を返します。最大 10 個の代替文字起こしを返すように選択できます。が Amazon Transcribe 識別する選択肢よりも多くの選択肢を指定すると、実際の選択肢の数のみが返されます。

代替案はすべて同じ文字起こし出力ファイルにあり、セグメントレベルで表示されます。セグメントは、スピーカーの変更や音声の一時停止など、音声の自然な間のことです。

代替文字起こしは、バッチ文字起こしでのみ使用できます。

文字起こし出力は次のように構成されています。コード例の省略記号 (`...`) 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. 指定された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. 前の `transcript` セクションの各単語の信頼スコア。

```
"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. 代替の文字起こしは、文字起こし出力の segments 部分にあります。各セグメントの代替案は、信頼スコアの降順で並べられています。

```
"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。

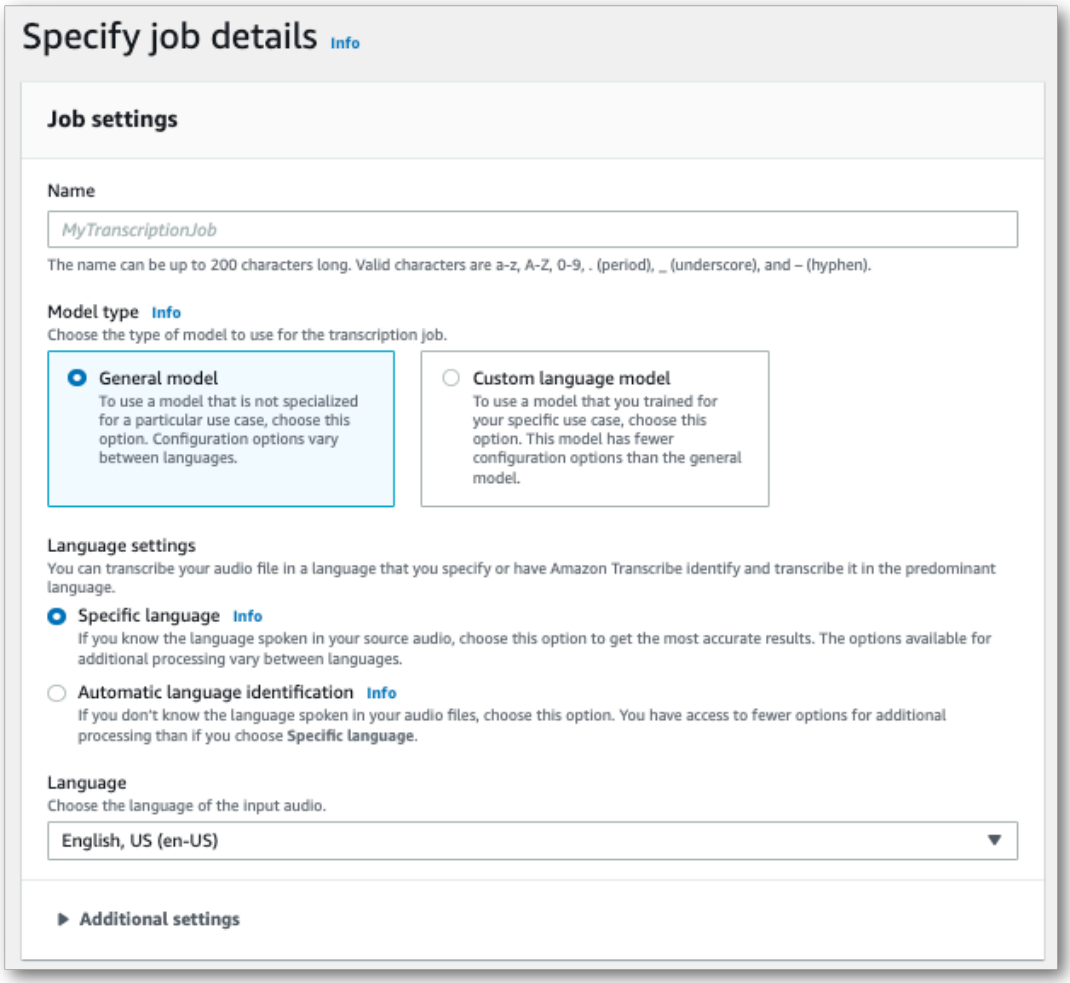
```
"status": "COMPLETED"
}
```

代替文字起こしのリクエスト

AWS マネジメントコンソール、AWS CLI、または AWS SDK を使用して代替文字起こし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。

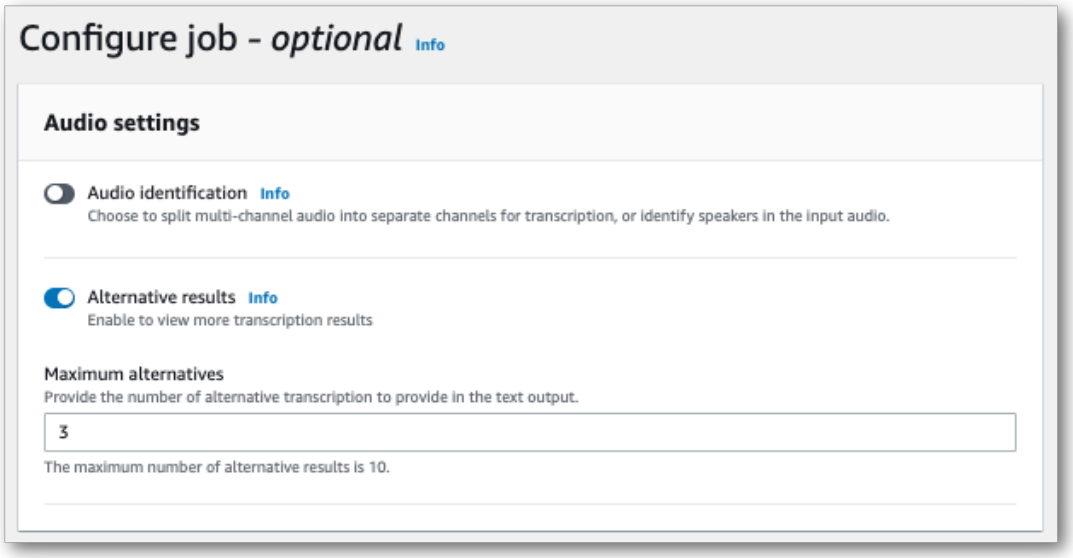
AWS マネジメントコンソール

1. AWS マネジメントコンソール 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。

   「代替結果」を選択し、トランスクリプトに必要な代替文字起こし結果の最大数を指定します。



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 コマンドと ShowAlternatives パラメータを使用します。詳細については、「StartTranscriptionJob」および「ShowAlternatives」を参照してください。

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

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

別の例として、start-transcription-job コマンド、および代替文字起こし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://filepath/my-first-alt-transcription-job.json
```

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

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

AWS SDK for Python (Boto3)

次の例では AWS SDK for Python (Boto3) 、 を使用して、start_transcription_job メソッドの ShowAlternatives引数を使用して代替文字起こしをリクエストします。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。

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

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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)
```

カスタム語彙および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、頭字語、技術用語、専門用語などのドメイン固有または非標準用語が含まれている場合、これらの用語 Amazon Transcribe は文字起こし出力で正しくキャプチャさ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 カスタム言語モデル を作成します。

- カスタム語彙 は、あらゆるコンテキストにおける特定の単語の認識と書式の両方を調整し、強化するように設計されています。これには、 Amazon Transcribe に単語と、オプションで発音と表示形式を提供することが含まれます。

  Amazon Transcribe がトランスクリプトで特定の用語を正しくレンダリングしていない場合は、これらの用語の表示 Amazon Transcribe 方法を示すカスタム語彙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適です。
- カスタム言語モデル は、用語に関連するコンテキストを把握するように設計されています。これには Amazon Transcribe 、大量のドメイン固有のテキストデータの提供が含まれます。

  Amazon Transcribe が技術用語を正しくレンダリングしていない場合、またはトランスクリプトで間違った同音符を使用している場合は、ドメイン固有の言語 Amazon Transcribe を学習す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」(流氷) と「フロー」(直線的な流れ) のどちらを使用するかを学習できます。

  このコンテキストに応じたアプローチは、ドメイン固有の音声を大量に文字起こしするのに最も適しています。カスタム言語モデルでは、カスタム語彙だけの場合よりも精度を大幅に向上させることができます。API は同じリクエストでカスタム言語モデルとカスタム語彙の両方を受け入れますが、カスタム言語モデルがアクティブな場合、カスタム語彙効果 (`DisplayAs`レンダリングなど) は文字起こし出力に適用されません。

> **Important**
>
> カスタム言語モデルが同じリクエストでアクティブになっている場合、カスタム語彙機能 (`DisplayAs`フィールドなど) は文字起こし出力に適用されません。これは、バッチ文字起こしとストリーミング文字起こしの両方に適用されます。

を使用してカスタム語彙を作成する方法の動画デモについては AWS マネジメントコンソール、「カスタム語彙の使用」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動画デモについては、「カスタム言語モデル (CLM) の使用による文字起こし精度の向上」を参照してください。

AWS Machine Learningブログで詳しく知る

カスタム語彙

- を使用した F1 レースのライブ文字起こし Amazon Transcribe

カスタム言語モデル

- speech-to-textへのパフォーマンスを向上させるカスタム言語モデルの構築 Amazon Transcribe
- Amazon Transcribeのカスタム言語モデルにより、講義の文字起こしの精度を向上

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

- あたり最大 100 個のカスタム語彙ファイルを持つことができます 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 マネジメントコンソール、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 マネジメントコンソール。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 マネジメントコンソール、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 マネジメントコンソール、 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** です。 頭字語の場合は、発音する文字をすべて	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。今後、この列のサポートは廃止される予定です。	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。今後、この列のサポートは廃止される予定です。	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列の内容 Amazon Transcribe を使用して出力を決定します。

フレーズ	SoundsLike	IPA	DisplayAs
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。			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、必ず 4 つの列ヘッダー (Phrase, SoundsLike, IPA, and DisplayAs) を含めてください。Phrase 列には、各行に必ずエントリを含めてください。IPA と SoundsLike による発音入力機能はサポート終了となりました。これらの列は空のままにしておいてください。これらの列の値はすべて無視されます。
- 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。今後、この列のサポートは廃止される予定です。
- 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 列には次の書式設定ルールがあります。
  - ピリオド (.)、アポストロフィ ()、またはハイフン (') で始めることはできません-。たとえば、**-hello**と **.test**は無効です。
  - アポストロフィ (') またはハイフン () で終わることはできません-。たとえば、 **hello-**と **world'**は無効です。
  - 繰り返しハイフン (--)、繰り返しピリオド (..)、繰り返しアポストロフィ () を含めることはできません''。たとえば、 **well--known**と **can''t**は無効です。
  - ピリオドは、頭字語 (などのピリオドで区切られた 1 文字) を示すためにのみ使用できます**A.B.C.**。ピリオドは、2 つ以上の連続する非特殊文字 (例: **AB.C**が無効) の後には表示できず、3 つ以上の連続する非特殊文字 (例: **A.BCD** が無効) の後には表示できません。
  - 言語の文字セットにリストされている文字のみがサポートされます。
- 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。LF と の両方のCRLF行末がサポートされています。
- 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。

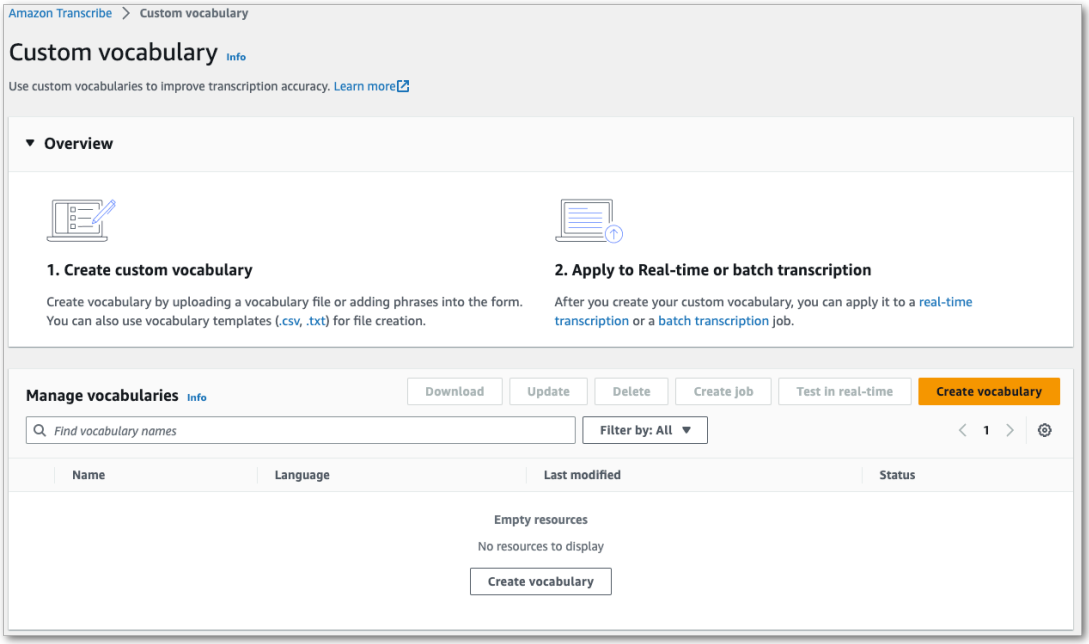
```
Phrase          [TAB]SoundsLike         [TAB]IPA                [TAB]DisplayAs
Los-Angeles     [TAB]                   [TAB]                   [TAB]Los Angeles
Eva-Maria       [TAB]                   [TAB]                   [TAB]
A.B.C.-s        [TAB]                   [TAB]                   [TAB]ABCs
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 マネジメントコンソール

1. AWS マネジメントコンソール 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。

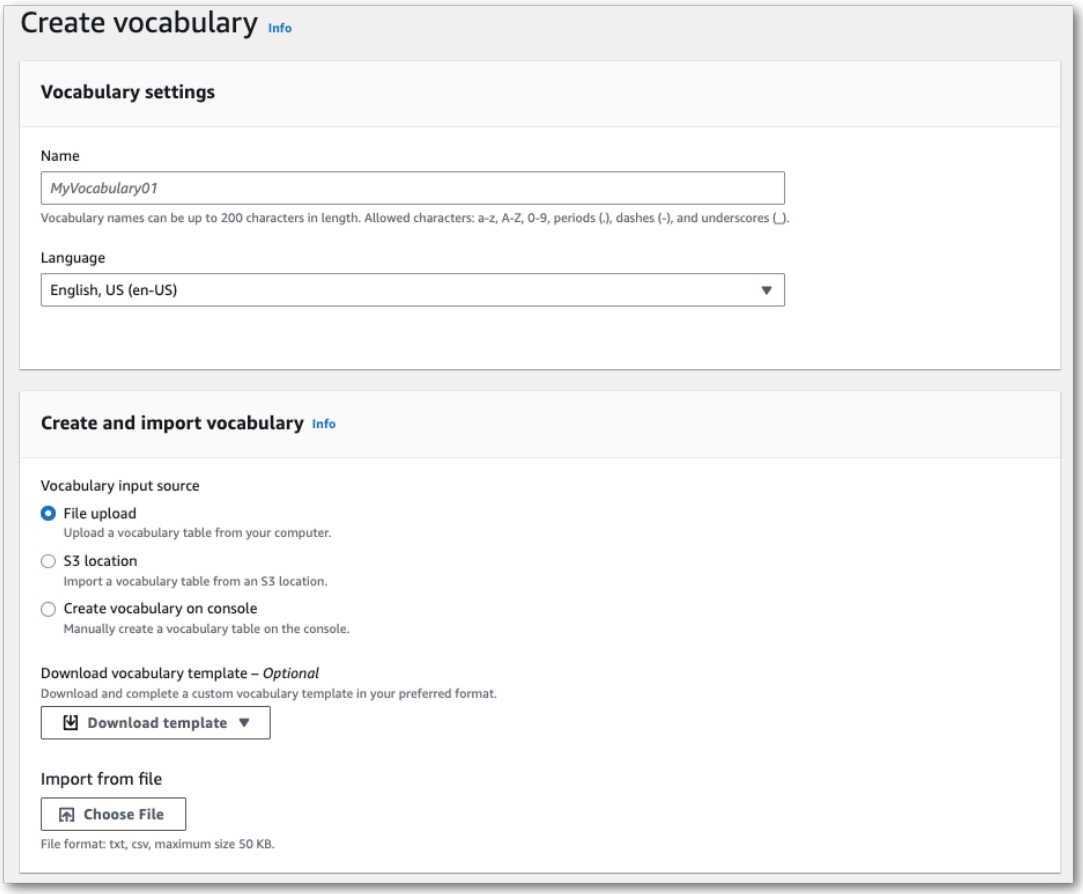


「語彙の作成」ページに移動します。新しいカスタム語彙の名前を入力します。

次の 3 つの選択肢があります。

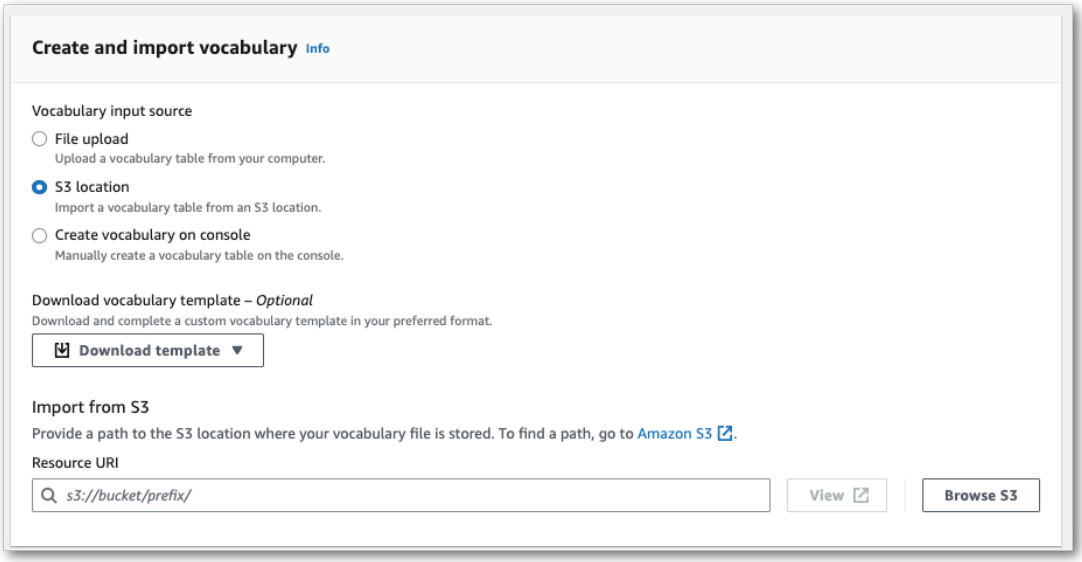
a. コンピュータから txt または csv ファイルをアップロードします。

   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。



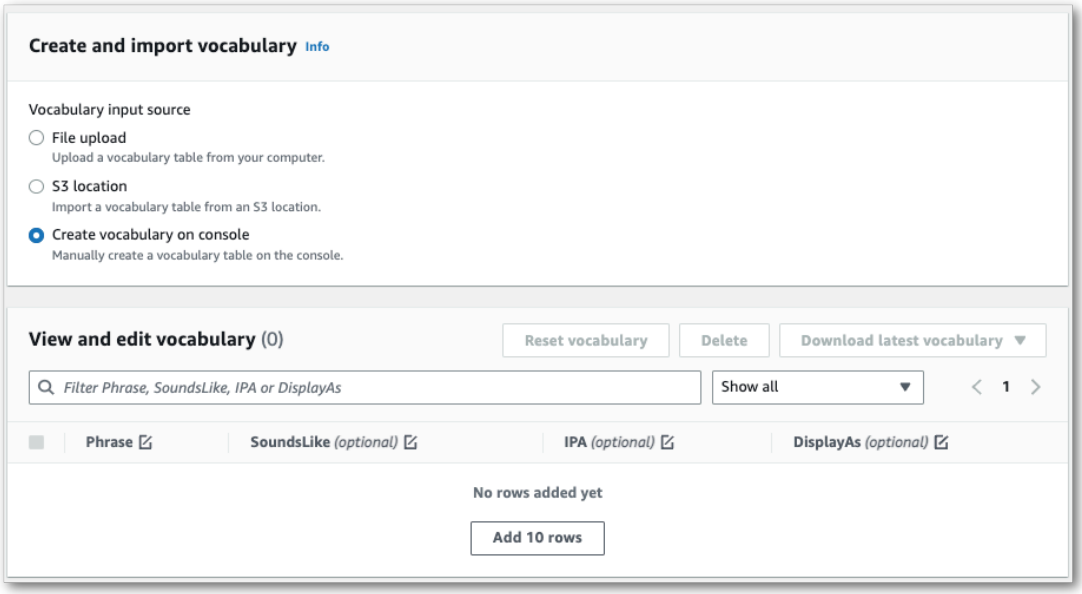
b. Amazon S3 場所から txt または csv ファイルをインポートします。

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。

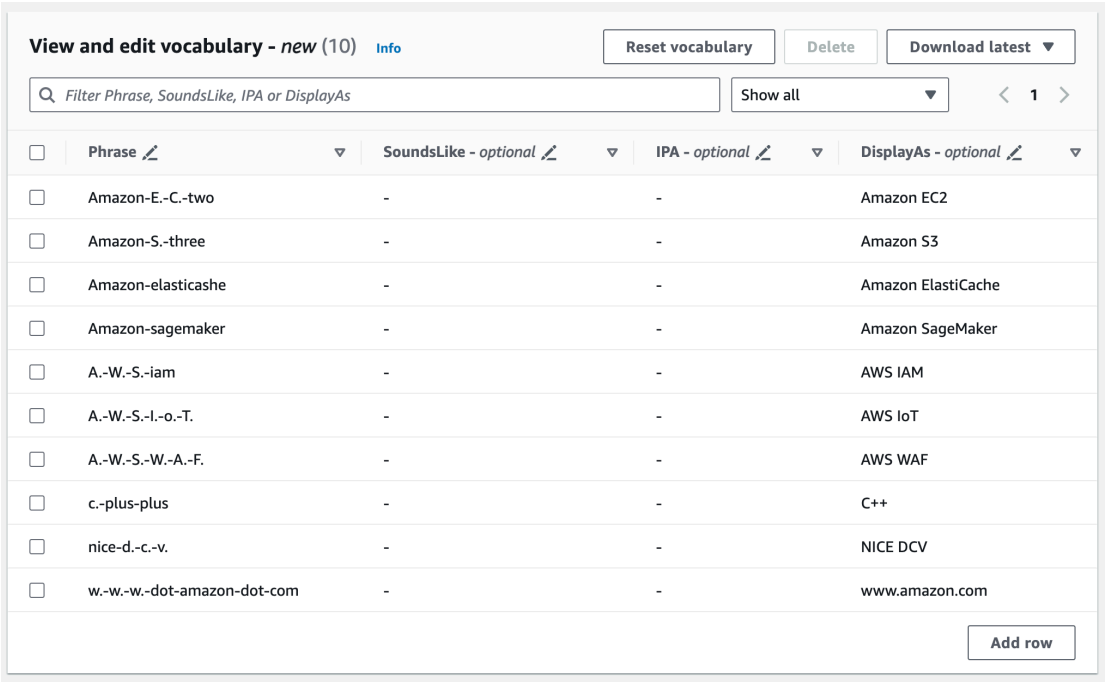


c. コンソールで語彙を手動で作成します。

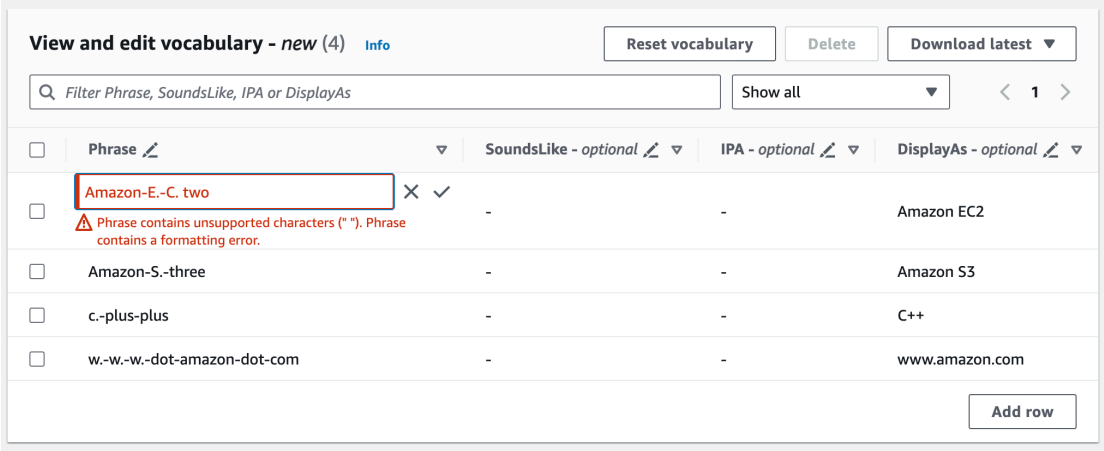
語彙の表示と編集ペインまでスクロールし、[10 行追加] を選択します。用語を手動で入力できるようになりました。



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。

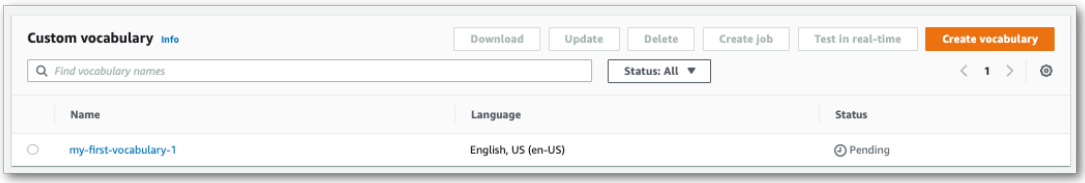


エラーがあると詳細なエラーメッセージが表示されるので、語彙を処理する前に問題を修正できます。[語彙の作成] を選択する前にすべてのエラーを修正しないと、語彙のリクエストは失敗するので注意してください。

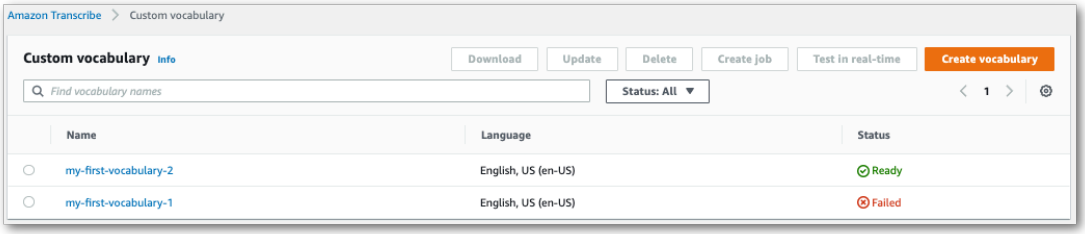


チェックマーク (✓) を選択して変更を保存するか、「X」を選択して変更を破棄します。

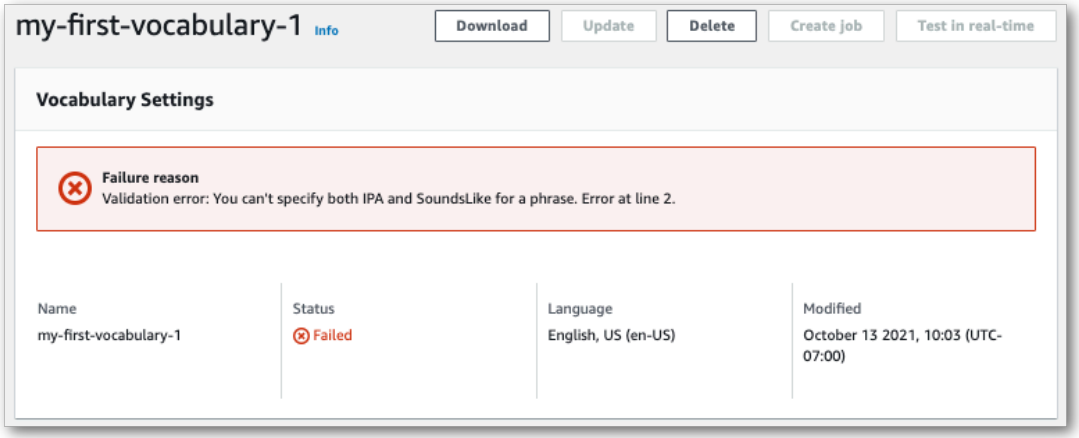
5. オプションで、カスタム語彙にタグを追加します。すべてのフィールドを入力し、語彙に問題がなければ、ページの一番下にある [語彙の作成] を選択します。カスタム語彙のページに戻ると、カスタム語彙のステータスを確認できます。ステータスが「保留中」から「準備完了」に変わったら、カスタム語彙を文字起こしに使用できます。



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 VocabularyName Settingsフィールドに を設定するか、ドロップダウンリストからカスタム語 AWS マネジメントコンソール彙を選択します。

```
aws transcribe create-vocabulary \
--vocabulary-name my-first-vocabulary \
--vocabulary-file-uri s3://amzn-s3-demo-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、次のリクエストボディが含まれています。

```
{
  "VocabularyName": "my-first-vocabulary",
  "VocabularyFileUri": "s3://amzn-s3-demo-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 VocabularyName Settingsフィールドに を設定するか、ドロップダウンリストからカスタム語 AWS マネジメントコンソール彙を選択します。

機能固有の例、シナリオ例、クロスサービス例など、 AWS 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。

```
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://amzn-s3-demo-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のポリシーの例。

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が予定されているため、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することを強くおすすめします。

AWS マネジメントコンソール、、または AWS SDKs を使用して AWS CLI、リストからカスタム語彙を作成できます。

- AWS マネジメントコンソール: カスタム語彙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。行区切りまたはカンマ区切りのエントリを使用できます。リストはテキスト (*.txt) ファイルとして保存する必要があります。LF と の両方のCRLF行末がサポートされています。
- AWS CLI および AWS SDK: `Phrases` フラグを使用して、API コール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。

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 **`Andorra-la-Vella`** とします。

以下は 2 つの有効なリスト形式の例です。メソッド固有の例については、「カスタム語彙リストの作成」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。

カスタム語彙リストは、CreateMedicalVocabulary オペレーションではサポートされていません。医療用のカスタム語彙を作成する場合は、テーブル形式を使用する必要があります。手順については、「テーブルを使用してカスタム語彙を作成する」を参照してください。

カスタム語彙リストの作成

で使用するカスタム語彙リストを処理するには 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。

```
aws transcribe create-vocabulary \
--vocabulary-name my-first-vocabulary \
--language-code en-US \
--phrases {CLI,Eva-Maria,ABCs}
```

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-list.json
```

ファイル my-first-vocab-list.json には、次のリクエストボディが含まれています。

```
{
  "VocabularyName": "my-first-vocabulary",
  "LanguageCode": "en-US",
  "Phrases": [
        "CLI","Eva-Maria","ABCs"
  ]
}
```

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リスト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。

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。

```
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**
>
> 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のポリシーの例。

カスタム語彙の使用

カスタム語彙を作成したら、それを文字起こしのリクエストに含めることができます。例については、以下のセクションを参照してください。

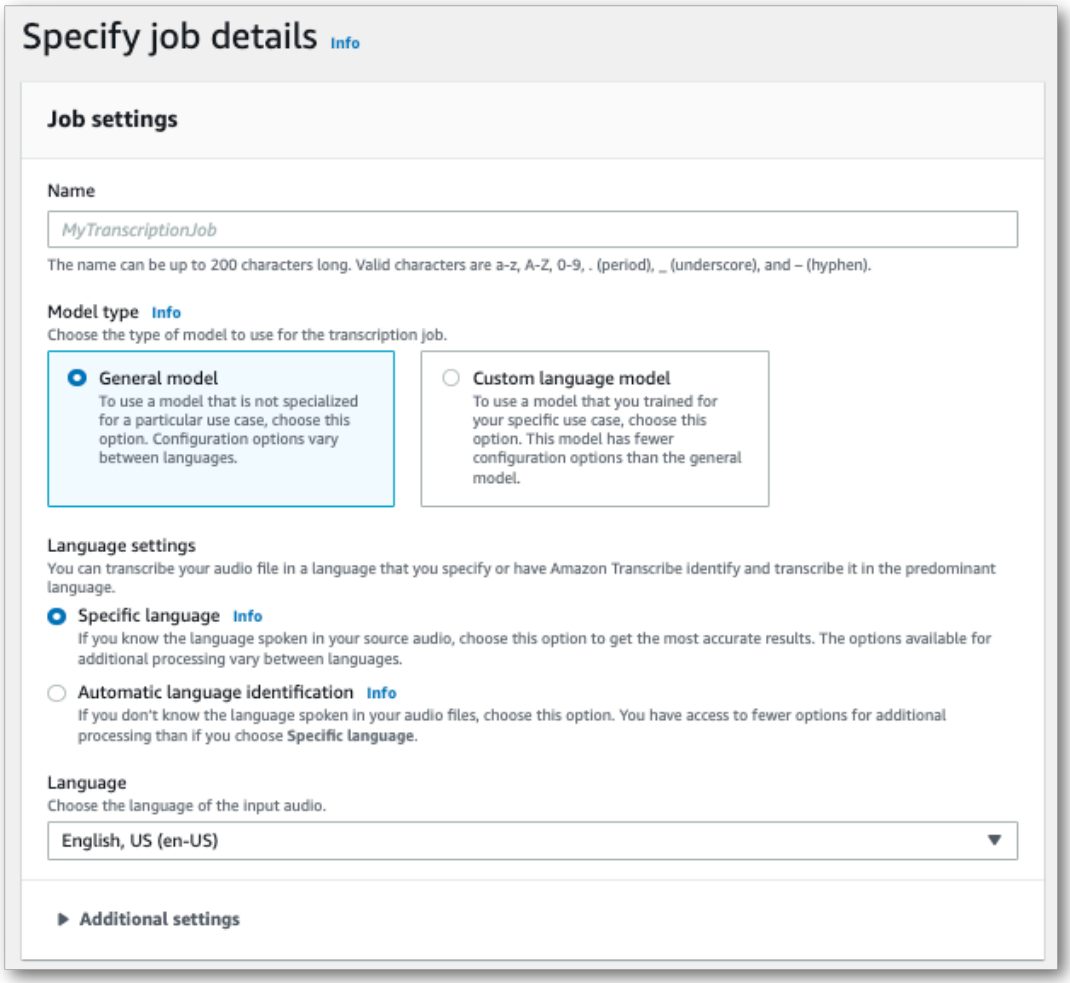
リクエストに含めるカスタム語彙の言語は、メディアに指定した言語コードと一致する必要があります。言語が一致しない場合、カスタム語彙は文字起こしに適用されず、警告やエラーも発生しません。

バッチ文字起こしでカスタム語彙を使用する

バッチ文字起こしでカスタム語彙を使用するには、以下の例を参照してください。

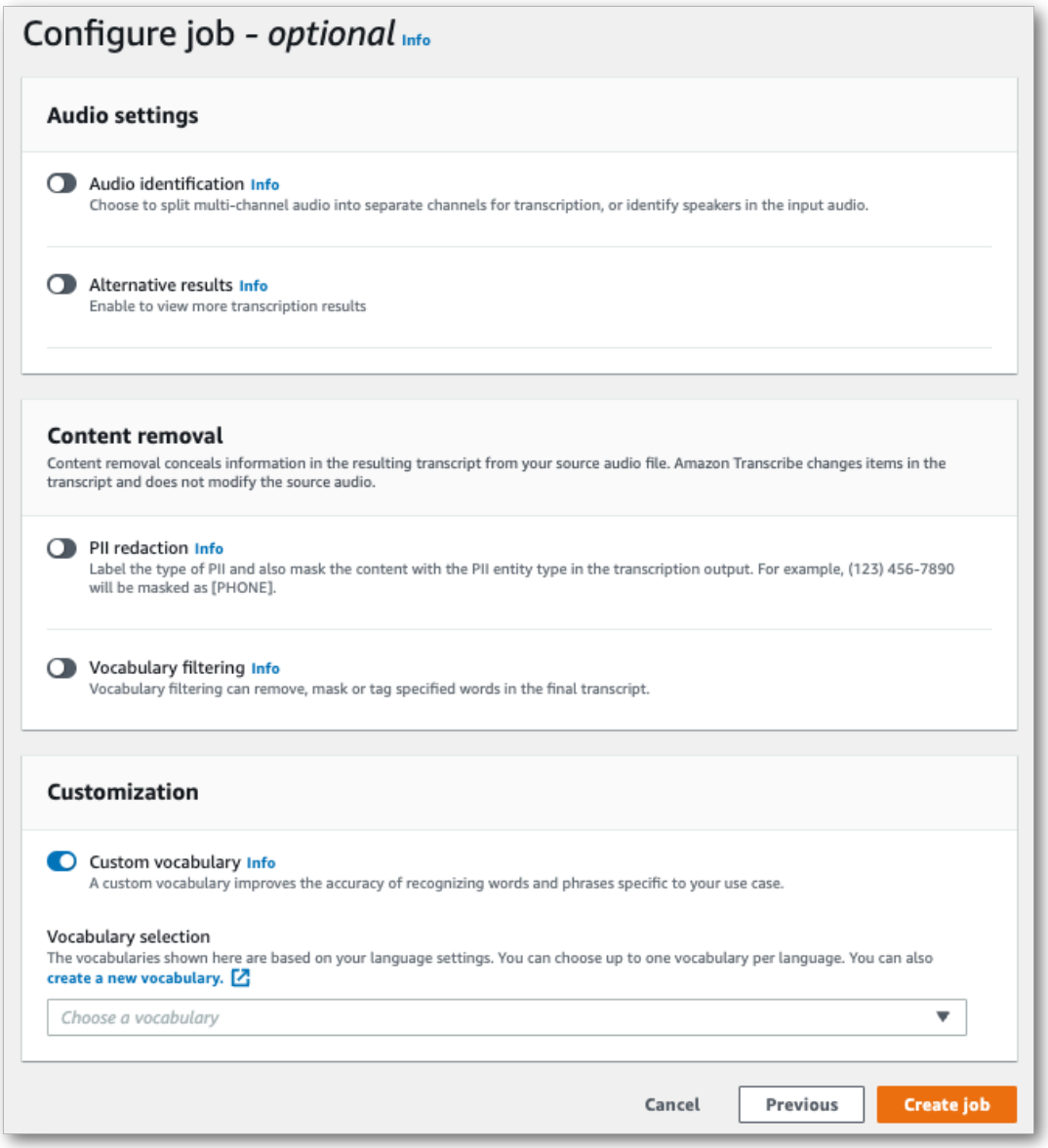
AWS マネジメントコンソール

1. AWS マネジメントコンソール 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。



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、他のフィールドも追加し、[次へ] を選択します。

3. ジョブの設定ページの下部にあるカスタマイズパネルで、カスタム語彙をオンに切り替えます。



4. ドロップダウンメニューから [カスタム語彙] を選択します。

   [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 Settings パラメータ、VocabularyName サブパラメータを使用します。詳細については、「StartTranscriptionJob」および「Settings」を参照してください。

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

```
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
--language-code en-US \
--settings VocabularyName=my-first-vocabulary
```

以下は、start-transcription-job コマンド、およびそのジョブでカスタム語彙を含むリクエストボディを使用した別の例です。

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

ファイル my-first-vocabulary-job.json には、次のリクエストボディが含まれています。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-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) 、 を使用して、start_transcription_job メソッドの Settings引数を使用してカスタム語彙を含めます。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。

```
from __future__ import print_function
import time
import boto3
```

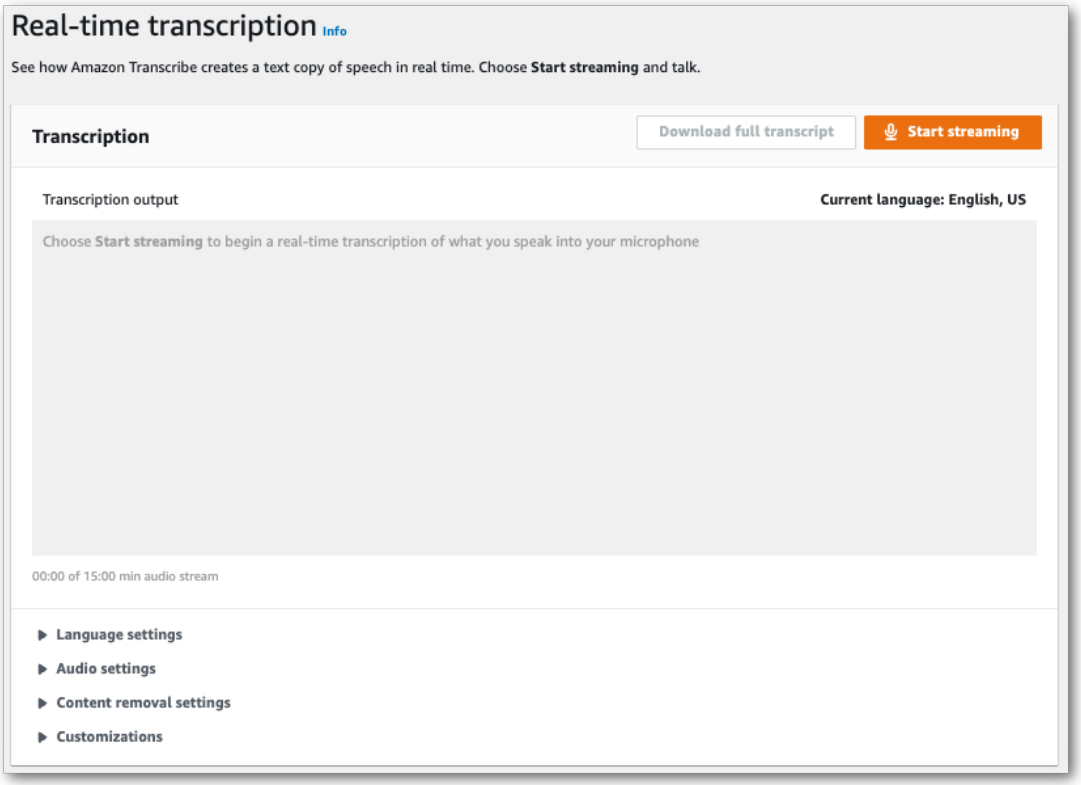
```
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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語彙の使用

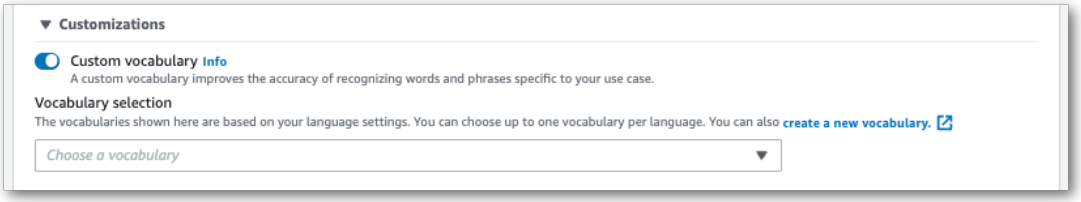
ストリーミング文字起こしでカスタム語彙を使用するには、以下の例を参照してください。

AWS マネジメントコンソール

1. AWS マネジメントコンソール 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。カスタマイズまでスクロールして、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. カスタム語彙をオンにして、ドロップダウンメニューから [カスタム語彙] を選択します。



   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、カスタム語彙を含む 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。 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-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のパラメータは、「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、カスタム語彙を WebSocket 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。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 ストリームの設定。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のパラメータは、「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に対する文字起こしの精度を向上させるため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コンテンツも含まれます。たとえば、科学会議の議事録の文字起こしをする場合、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できない可能性があります。このような場合は、専門分野で使用されている特殊な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識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コンテキストを学習します。これには、単語がいつどのように使われているか、ある単語が他の単語とどのような関係にあるかなどが含まれます。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、モデルは「氷の流れ」よりも「流氷」という単語である可能性が高いことを学習するかもしれません。

カスタム言語モデルでサポートされる言語を確認するには、「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。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、さまざまなタイプ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音声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、音声と同じコンテキスト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。これはドメイン内の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、文字起こししたい音声とまったく同じ用語、使用法、コンテキストがあります。

正確なトランスクリプトがない場合は、ジャーナル記事、技術レポート、ホワイトペーパー、会議議事録、取扱説明書、ニュース記事、ウェブサイトコンテンツ、および音声と同様のコンテキストで使用される用語が含まれるテキストを使用します。これはドメイン関連の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、音声で話されている用語を含む大量のテキストデータが必要になります。最大 2 GB のテキストデータ Amazon Transcribe を提供してモデルをトレーニングできます。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、最大 200 MB のテキストデータ Amazon Transcribe を提供してモデルをチューニングできます。これはチューニングデータと呼ばれます。

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、これらの用語が使用されるコンテキストを学習 Amazon Transcribe するように に教えることです。堅牢なモデルを作成するには、 Amazon Transcribe は大量の関連テキストデータが必要になることがあります。2 GB の上限まで、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すすめします。

チューニングデータの目的は、トレーニングデータから学習したコンテキストとの関連を絞り込み、最適化することです。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。

トレーニングデータと、オプションでチューニングデータをどのように選択するかを決めるのは、お客様です。それぞれのケースは一意的で、持っているデータのタイプ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。

両方のデータタイプを選択する場合は、トレーニングデータとチューニングデータが重複しないようにします。トレーニングデータとチューニング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すすめします。一般的なシナリオを優先順位の高いものからご紹介します

- 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ます。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強することを検討します。この場合は、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ように確保します。
- ドメイン内のトランスクリプトテキストがない場合は、ドメイン関連のテキストをすべて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最も効果的でないシナリオです。

モデルを作成する準備ができたら、「カスタム言語モデルの作成」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を作成する前に、次のことを行う必要があります。

- データの準備 データはプレーンテキスト形式で保存し、特殊文字は使用できません。
- データを Amazon S3 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すすめします。
- Amazon Transcribe が Amazon S3 バケットにアクセスできることを確認します。データを使用するには、アクセス許可を持つ IAM ロールを指定する必要があります。

データの準備

すべてのデータを 1 つのファイルにコンパイルすることも、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ります。

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はいくつ使ってもかまいません。100,000 語のファイルを 1 つアップロードしても、10,000 語のファイルを 10 個アップロードするの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していることを確認します。

- すべて作成したいモデルと同じ言語である必要があります。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、すべてのテキストデータが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は含まれません。
- これらのデータを合計したサイズは、トレーニングデータで最大 2 GB、チューニングデータで最大 200 MB です。

これらの基準のいずれかが満たされない場合、モデルは動作しません。

データのアップロード

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ます。

バケットの URI は以下のようになります。

- `s3://amzn-s3-demo-bucket/my-model-training-data/`
- `s3://amzn-s3-demo-bucket/my-model-tuning-data/`

トレーニングデータとチューニング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、追加した場合は、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、 Amazon S3 バケットへのアクセス許可を持つ IAM ロールを指定する必要があります。トレーニングデータを配置した Amazon S3 バケットにアクセスできるロールがまだない場合は、作成する必要があります。ロールを作成した後、ロールにアクセス許可を付与するポリシーをアタッチできます。ポリシーをユーザーにアタッチしないください。

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のポリシーの例」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、IAM 「ID (ユーザー、ユーザーグループ、ロール)」を参照してください。

キーポリシーの詳細については、以下を参照してください。

- IAMでのポリシーとアクセス許可
- 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

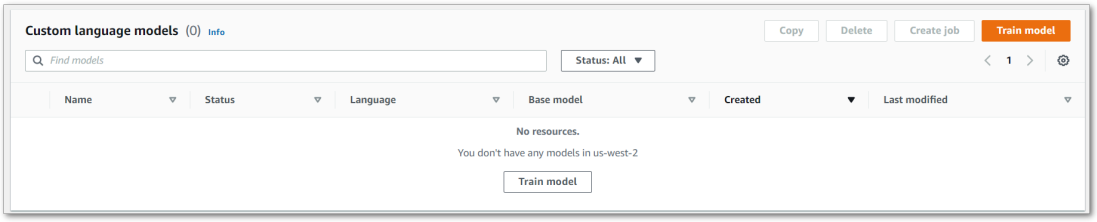
カスタム言語モデルを作成するときは、ベースモデルを選択する必要があります。2 つのベースモデルのオプションがあります。

- `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。

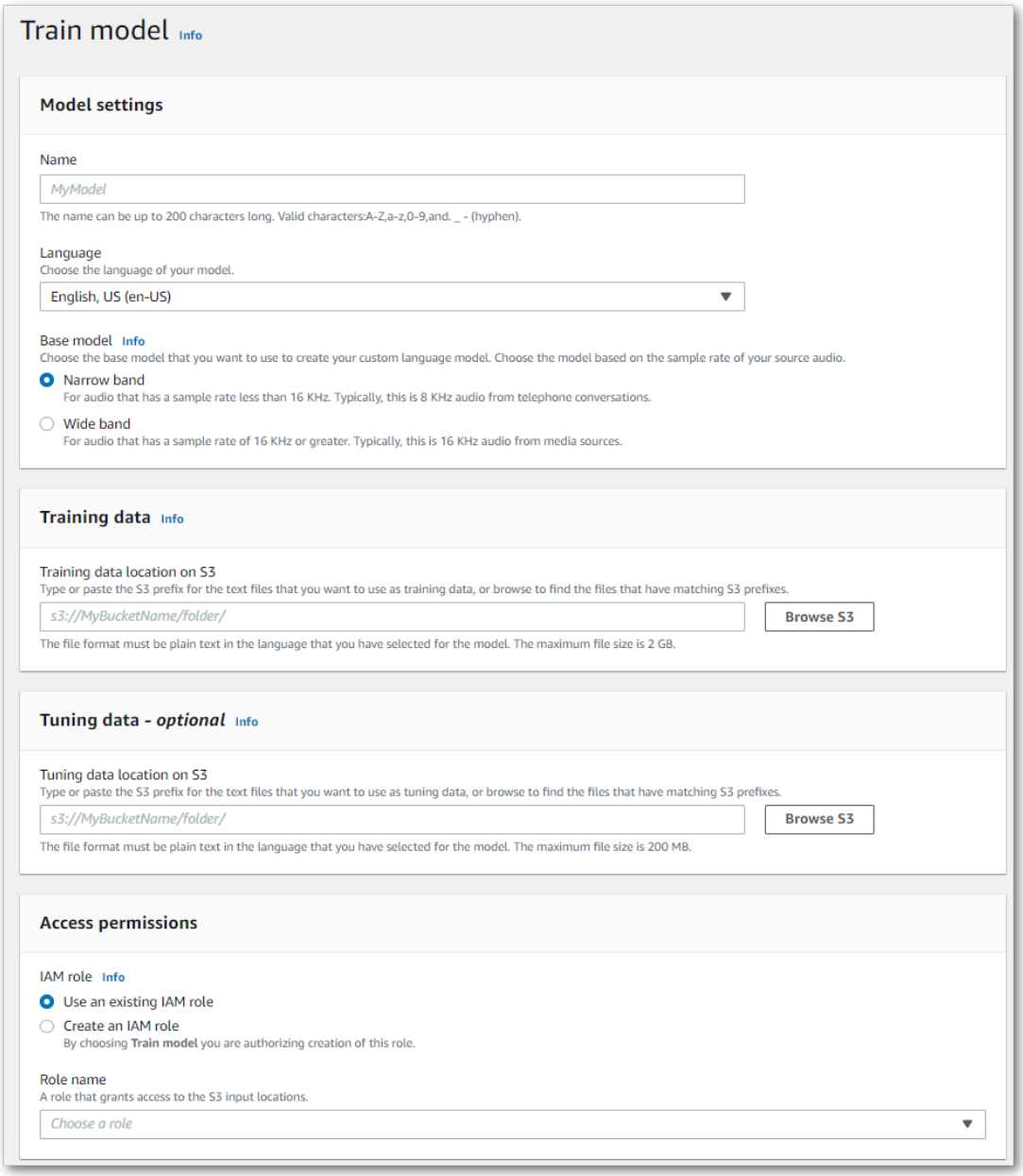
カスタム言語モデルは AWS マネジメントコンソール、 AWS CLI、、または AWS SDKs.;以下の例を参照してください。

AWS マネジメントコンソール

1. AWS マネジメントコンソール 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。



これにより、モデルのトレーニングページに移動します。名前を追加し、言語を指定し、モデルに使用するベースモデルを選択します。次に、そのパスをトレーニングに追加し、オプションでチューニングデータも追加します。データにアクセスするためのアクセス許可を持つ IAM 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、ページ下部にある [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://amzn-s3-demo-bucket/my-clm-training-
data/,TuningDataS3Uri=s3://amzn-s3-demo-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ています。

```
{
  "BaseModelName": "NarrowBand",
  "ModelName": "my-first-language-model",
  "InputDataConfig": {
        "S3Uri": "s3://amzn-s3-demo-bucket/my-clm-training-data/",
        "TuningDataS3Uri"="s3://amzn-s3-demo-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 メソッドを使用して 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。

```
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://amzn-s3-demo-bucket/my-clm-training-data/',
        'TuningDataS3Uri':'s3://amzn-s3-demo-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 は、カスタム言語モデルで使用できるベースモデルを継続的に更新します。これらの更新を活用するには、6 ~ 12 か月ごとに新しいカスタム言語モデルのトレーニングを行うことをおすすめします。

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 を使用して`DescribeLanguageModel`リクエストを実行し、レスポンスで`UpgradeAvailability`フィールドを見つけます。

`UpgradeAvailability` が `true` の場合、モデルはベースモデルの最新バージョンを実行し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使用

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、以下のセクションを参照してください。

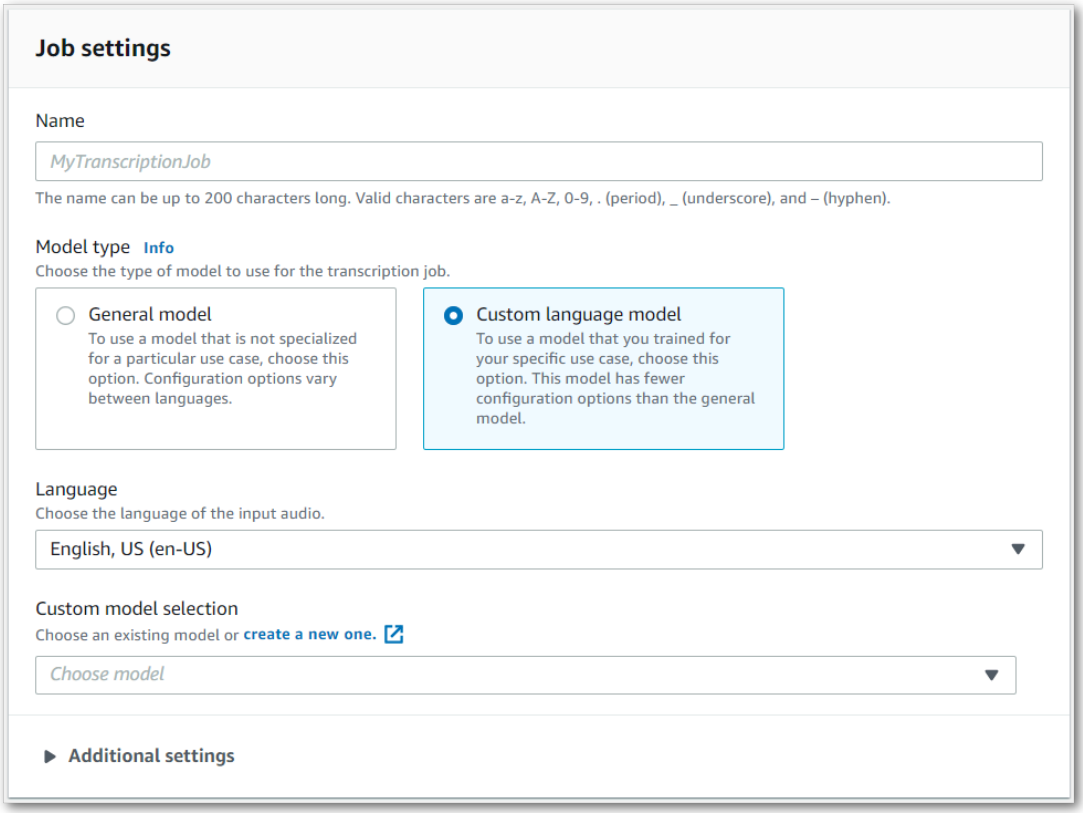
リクエストに含めるモデルの言語は、メディアに指定した言語コードと一致する必要があります。言語が一致しない場合、カスタム言語モデルは文字起こしに適用されず、警告やエラーも発生しません。

バッチ文字起こしでカスタム言語モデルを使用する

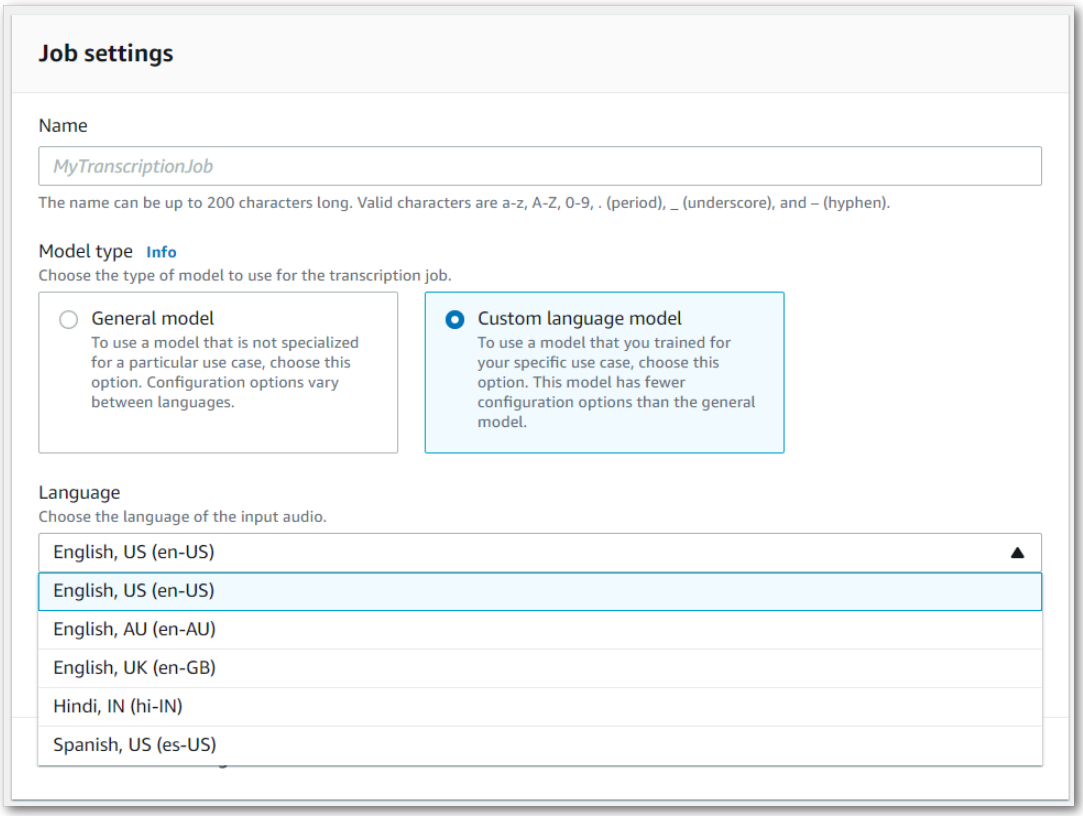
バッチ文字起こしでカスタム言語モデルを使用するには、以下の例を参照してください。

AWS マネジメントコンソール

1. AWS マネジメントコンソール 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] を選択します。

   入力データパネルに入力ファイル Amazon S3 の場所を追加します。

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、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 ModelSettings パラメータ、VocabularyName サブパラメータを使用します。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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
```

```
--language-code en-US \
--model-settings LanguageModelName=my-first-language-model
```

以下は、start-transcription-jobコマンド、および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た別の例です。

```
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 には、次のリクエストボディが含まれています。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-bucket",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "ModelSettings": {
        "LanguageModelName": "my-first-language-model"
   }
}
```

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 ModelSettings引数を使用するカスタム言語モデルを含めます。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。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
```

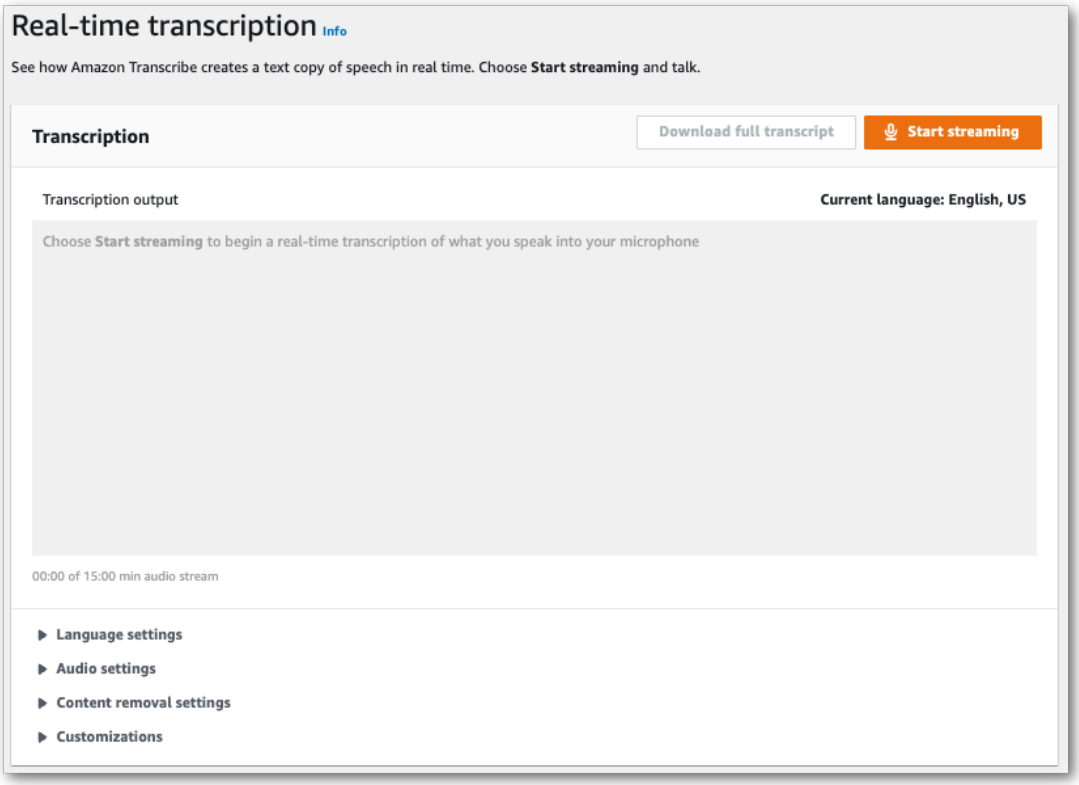
```
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-bucket',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    ModelSettings = {
        'LanguageModelName': 'my-first-language-model'
   }
)

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)
```

ストリーミング文字起こしでカスタム言語モデルを使用する

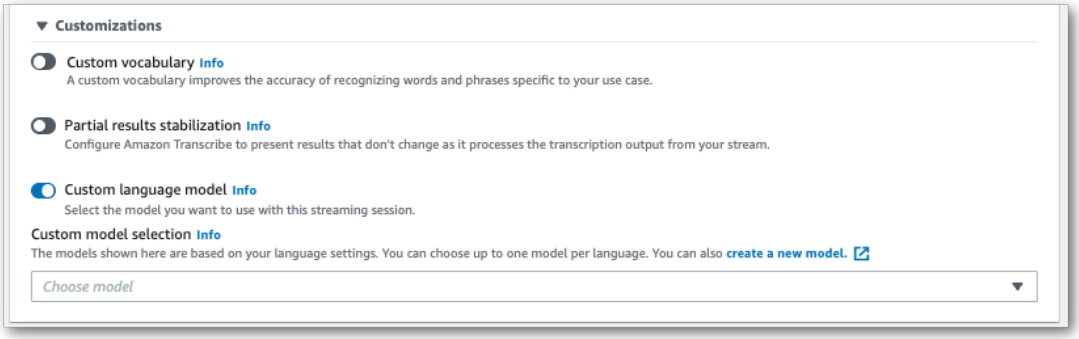
ストリーミング文字起こしでカスタム言語モデルを使用するには、以下の例を参照してください。

AWS マネジメントコンソール

1. AWS マネジメントコンソール 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。カスタマイズまでスクロールして、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] をオンにして、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、カスタム言語モデルで 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。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のパラメータは、「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 ストリーミングでカスタム言語モデルを適用する署名付き 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。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 ストリームの設定。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のパラメータは、「共通パラメータ」セクションに記載されています。

カスタム語彙フィルターを使用して単語を削除、マスキング、またはフラグ付けする

カスタム語彙フィルターは、文字起こし出力で修正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、不快な言葉や冒涜的な用語の削除があります。ただし、カスタム語彙フィルターは完全にカスタム仕様なので、好きな単語を選択できます。たとえば、新製品の発売が間近に迫っている場合、会議の議事録に製品名をマスクすることができます。この場合、製品名は発売まで秘密にしておきながら、ステークホルダーに最新の情報を提供します。

語彙フィルタリングには mask、remove、および tag の 3 つの表示方法があります。次の例を参照して、それぞれの仕組みを確認します。

- 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: 指定した各単語にタグ ("vocabularyFilterMatch": true) を追加しますが、単語そのものは変更しません。タグ付けにより、トランスクリプトの置換や編集をすばやく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、カスタム語彙フィルターと適用するフィルタリング方法を指定できます。 Amazon Transcribe 次に、指定したフィルタリング方法に従って、文字起こしに表示される単語の完全一致を変更します。

カスタム語彙フィルターは、バッチおよびストリーミング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ーを作成する方法については、「語彙フィルターを作成する」を参照してください。カスタム語彙フィルターを適用する方法については、「カスタム語彙フィルターの使用」を参照してください。

> **Note**
>
> Amazon Transcribe は人種的に機密性の高い用語を自動的にマスクしますが、AWS テクニカルサポートに連絡してこのデフォルトのフィルターをオプトアウトできます。

語彙フィルタリングのチュートリアルについては、「語彙フィルターの使用」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

カスタム語彙フィルターの作成には次の 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する。
   - このアプローチは AWS マネジメントコンソール、、 AWS CLI、または AWS SDKsで使用できます。
   - を使用する場合は AWS マネジメントコンソール、カスタム語彙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 AWS SDKs を使用する場合は、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、リクエストに Amazon S3 URI を含める必要があります。
2. カンマ区切りの単語のリストを API リクエストに直接含めます。

- このアプローチは、`Words`パラメータを使用して AWS CLI または AWS SDKsで使用できます。

各方法の例については、「カスタム語彙フィルターの作成」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際の注意事項

- 単語では、大文字と小文字が区別されません。例えば、「curse」と「CURSE」は同じとみなされます。
- 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」または「swearing」という単語が含まれている場合、これらはフィルタリングされません。フィルターの対象となるのは「swear」のインスタンスのみです。そのため、フィルタリング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、トランスクリプトでは「submarine」は変更されません。
- 各エントリには 1 つの単語のみ入力できます (スペースなし)。
- カスタム語彙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である必要があります。
- あたり最大 100 個のカスタム語彙フィルターを持つことができ AWS アカウント 、それぞれ最大 50 KB のサイズにすることができます。
- 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。

カスタム語彙フィルターの作成

で使用するカスタム語彙フィルターを処理するには Amazon Transcribe、次の例を参照してください。

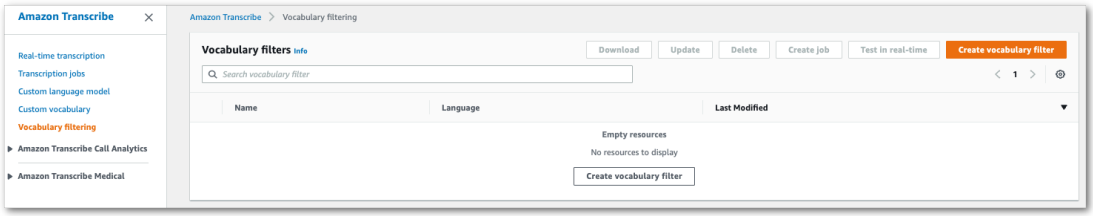
AWS マネジメントコンソール

続行する前に、カスタム語彙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。オプションで、ファイルを Amazon S3 バケットにアップロードできます。

1. AWS マネジメントコンソール にサインインします。

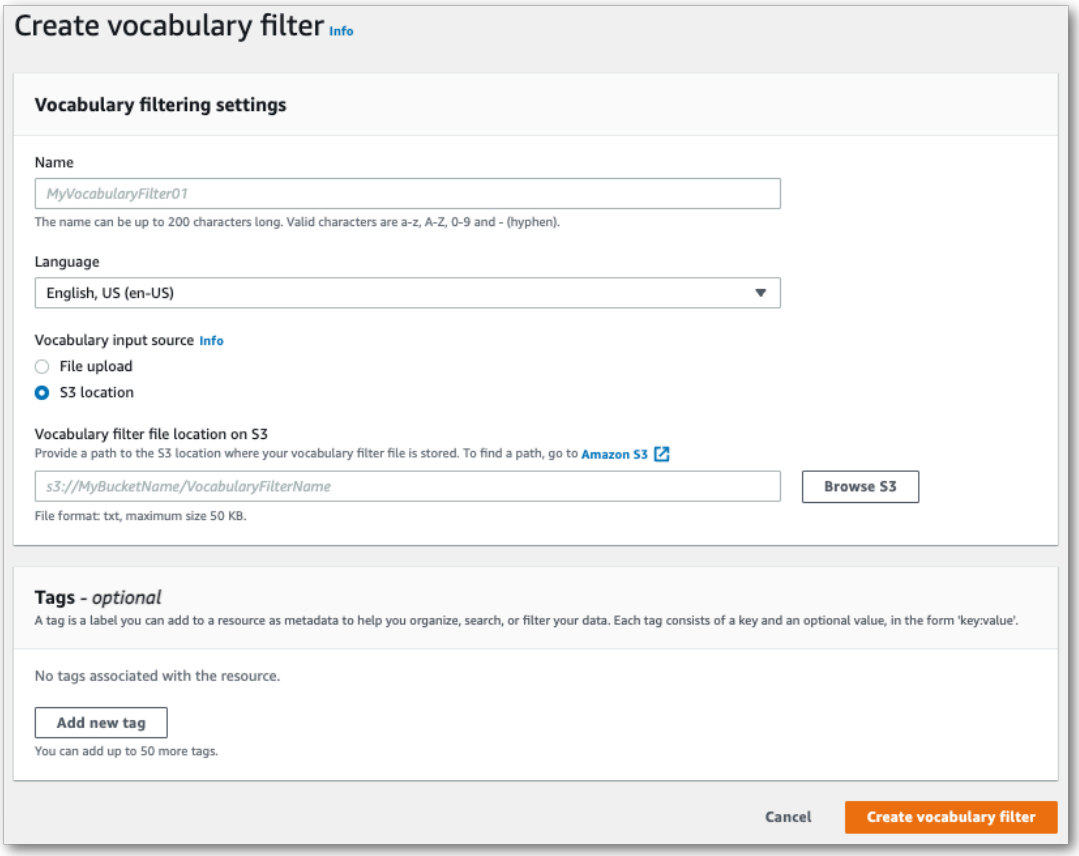
2. ナビゲーションペインで、[語彙フィルタリング] を選択します。語彙フィルターページが開き、既存のカスタム語彙フィルタ–を表示したり、新しい語彙フィルターを作成したりできます。

3. [語彙フィルターの作成] を選択します。



これにより「語彙フィルターの作成」ページに移動します。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 [ファイルアップロード] または [S3 ロケーション] 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、カスタム語彙フィルターにタグを追加します。すべてのフィールドを入力したら、ページの一番下にある [語彙フィルターの作成] 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-vocary-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語彙フィルターに加工します。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単語リストをリクエストに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: 単語リストをテキストファイルとして保存して Amazon S3 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ファイルの URI をリクエストに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://amzn-s3-demo-bucket/my-vocabulary-filters/my-
vocabulary-filter.txt
```

ここでは、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 には、次のリクエストボディが含まれています。

オプション 1: Words パラメータを使用して単語リストをリクエストに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: 単語リストをテキストファイルとして保存して Amazon S3 バケットにアップロードし、VocabularyFilterFileUri パラメータを使用してファイルの URI をリクエストに含めることができます。

```
{
  "VocabularyFilterName": "my-first-vocabulary-filter",
  "LanguageCode": "en-US",
  "VocabularyFilterFileUri": "s3://amzn-s3-demo-bucket/my-vocabulary-filters/my-
vocabulary-filter.txt"
}
```

> **Note**
>
> VocabularyFilterFileUri をリクエストに含めると、Words は使用できません。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_filter メソッドを使用してカスタム語彙フィルターを作成します。詳細については、「CreateVocabularyFilter」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: Words パラメータを使用して単語リストをリクエストに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: 単語リストをテキストファイルとして保存して Amazon S3 バケットにアップロードし、VocabularyFilterFileUri パラメータを使用してファイルの URI をリクエストに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://amzn-s3-demo-bucket/my-vocabulary-filters/my-
vocabulary-filter.txt'
)
```

> **Note**
>
> VocabularyFilterFileUri をリクエストに含めると、Words は使用できません。どちらか一方を選択する必要があります。

> **Note**
>
> カスタム語彙フィルターファイル用に新しい Amazon S3 バケットを作成する場合は、CreateVocabularyFilter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のポリシーの例。

カスタム語彙フィルターの使用

カスタム語彙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、以下のセクションを参照してください。

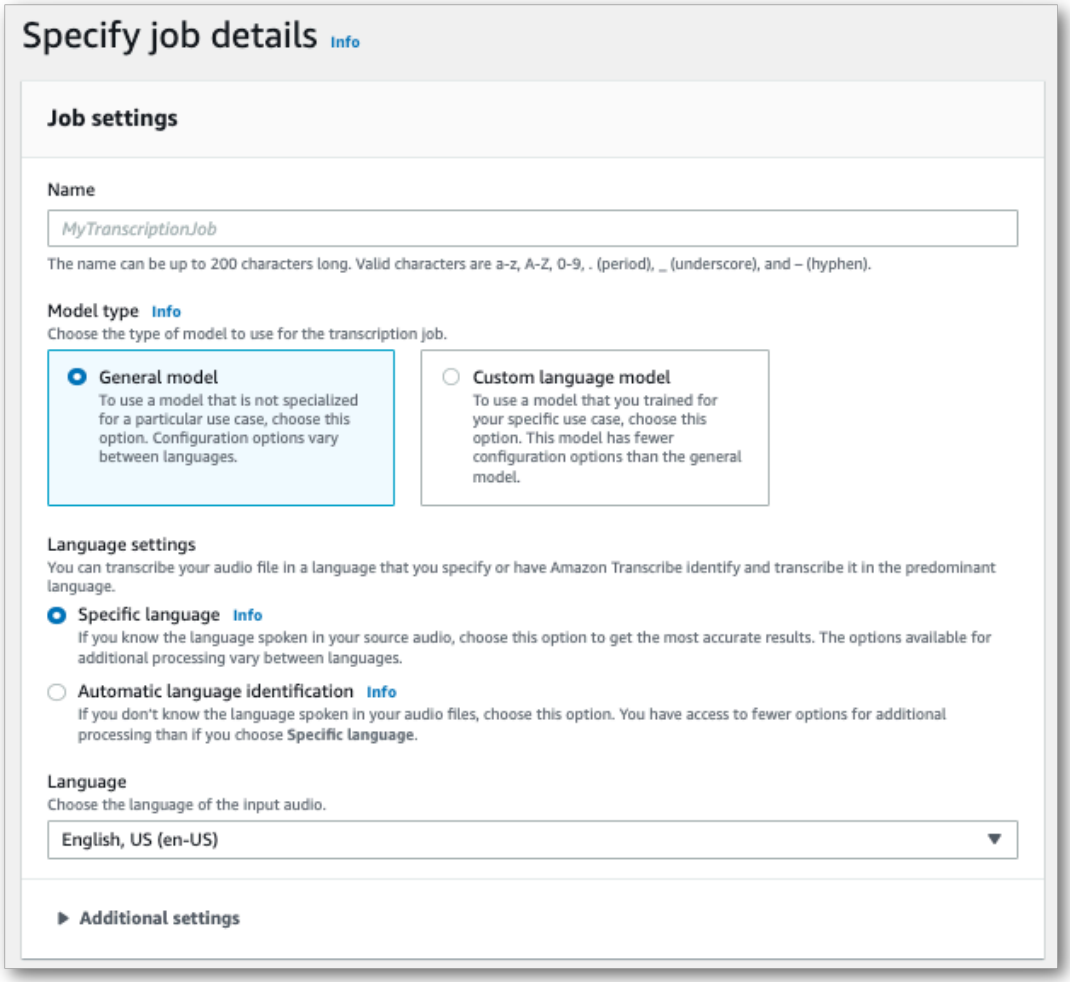
リクエストに含めるカスタム語彙フィルターの言語は、メディアに指定した言語コードと一致する必要があります。言語識別を使用して複数の言語オプションを指定する場合は、指定した言語ごとに 1 つのカスタム語彙フィルターを含めることができます。カスタム語彙フィルターの言語が音声で特定された言語と一致しない場合、フィルターは文字起こしに適用されず、警告やエラーも発生しません。

バッチ文字起こしでカスタム語彙フィルターを使用する

バッチ文字起こしでカスタム語彙フィルターを使用するには、以下の例を参照してください。

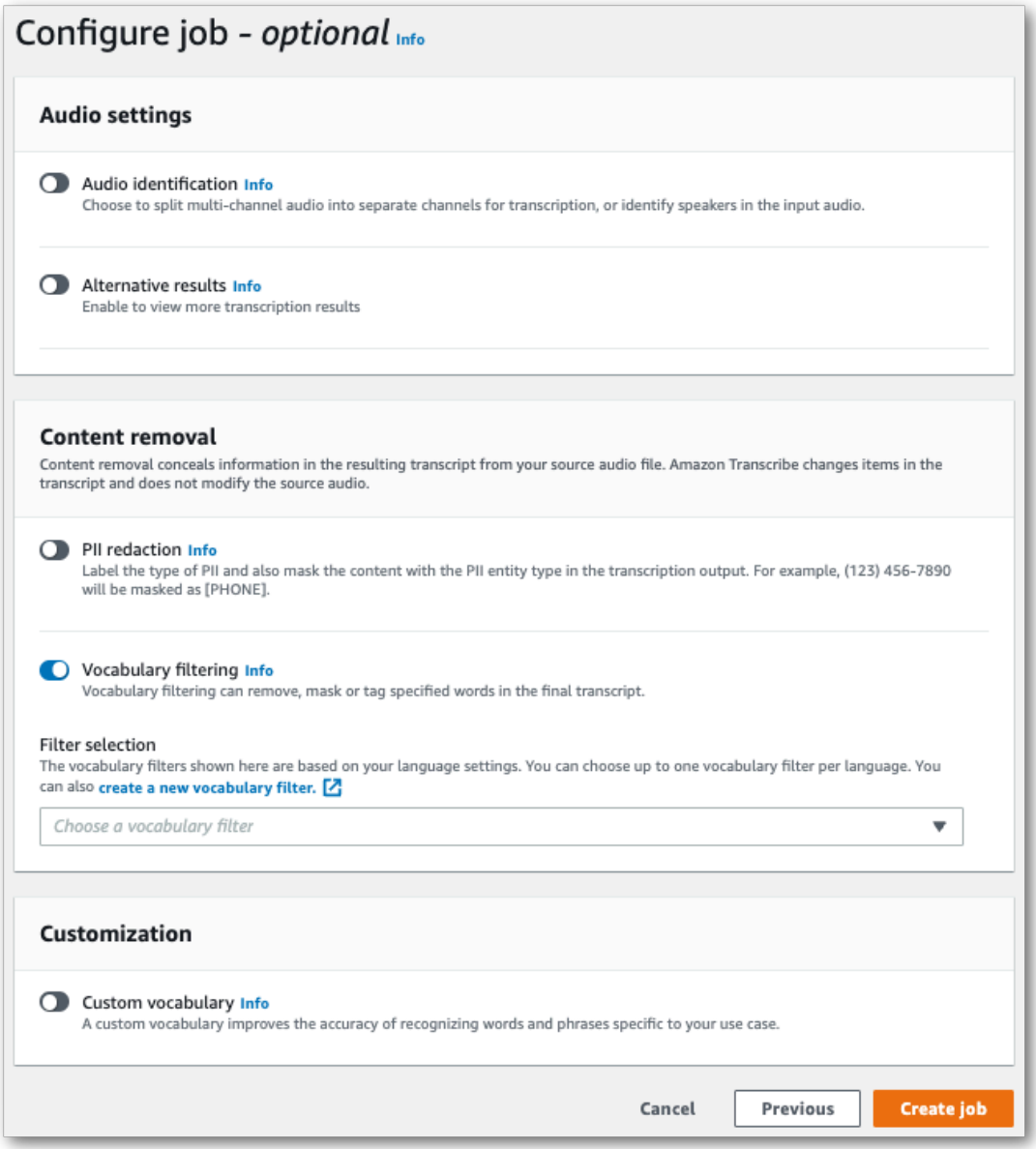
AWS マネジメントコンソール

1. AWS マネジメントコンソール 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。

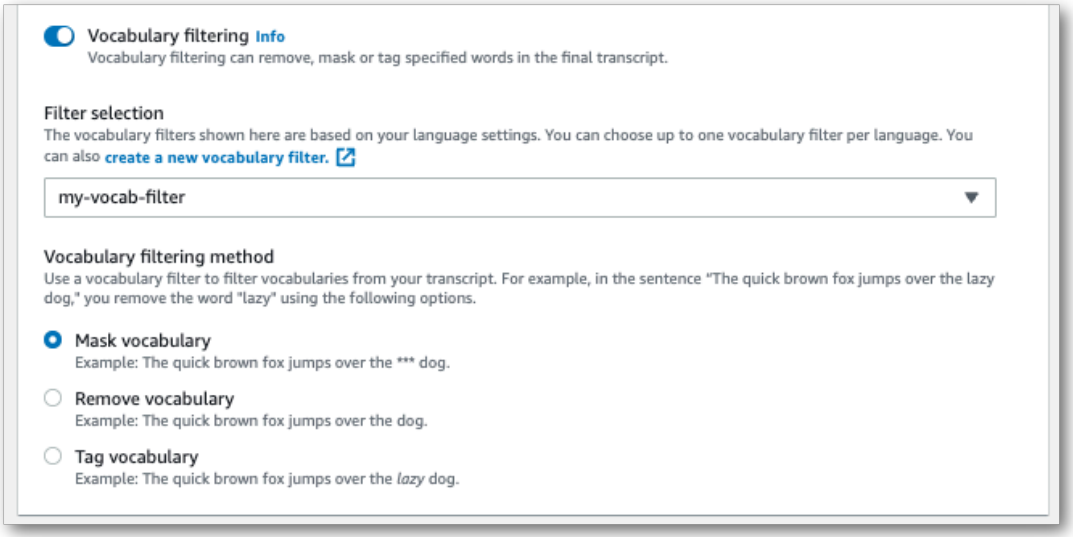


ジョブに名前を付け、入力メディアを指定します。オプションで、他のフィールドも追加し、[次へ] を選択します。

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語彙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 Settings パラメータ、VocabularyFilterName および VocabularyFilterMethod サブパラメータを使用します。詳細については、「StartTranscriptionJob」および「Settings」を参照してください。

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
--language-code en-US \
--settings VocabularyFilterName=my-first-vocabulary-filter,VocabularyFilterMethod=mask
```

以下は、start-transcription-job コマンド、およびそのジョブでカスタム語彙フィルターを含むリクエストボディを使用した別の例です。

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

ファイル mmy-first-vocabulary-filter-job.json には、次のリクエストボディが入含まれています。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-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) 、 を使用して、start_transcription_job メソッドの Settings引数を使用するカスタム語彙フィルターを含めます。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。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-bucket',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Settings = {
        'VocabularyFilterName': 'my-first-vocabulary-filter',
        'VocabularyFilterMethod': 'mask'
   }
)
```

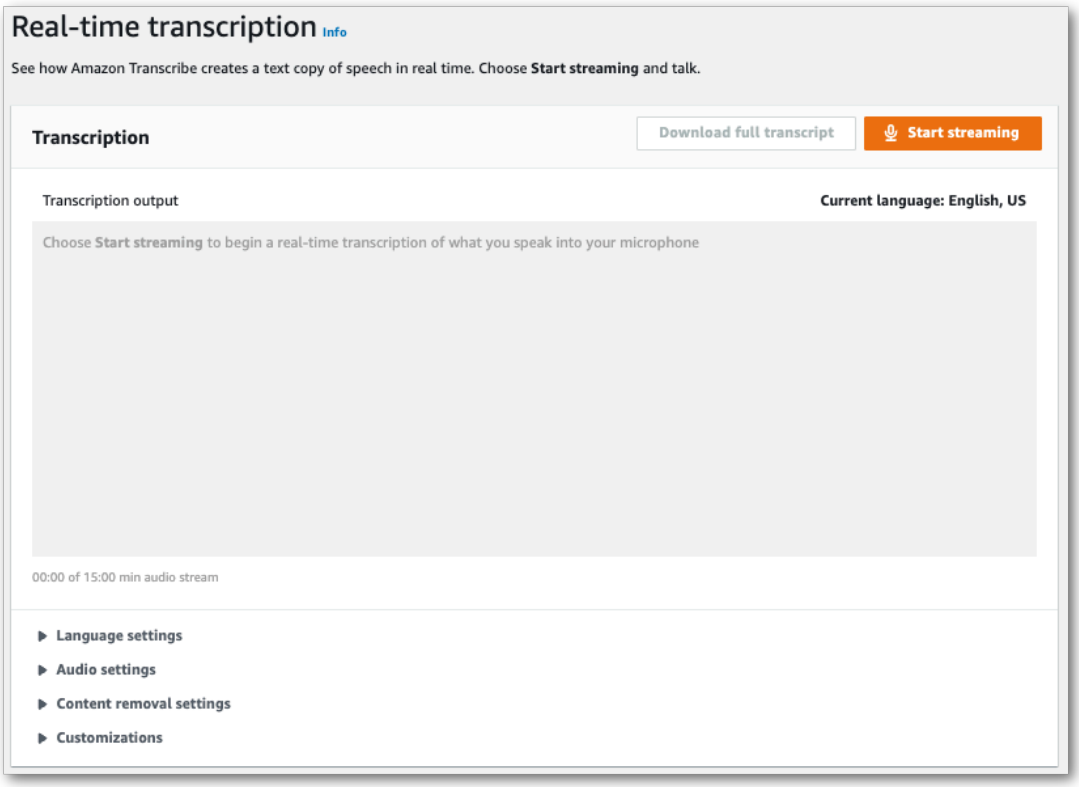
```
while True:
    status = transcribe.get_transcription_job(TranscriptionJobName = job_name)
    if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

ストリーミング文字起こしでのカスタム語彙フィルターの使用

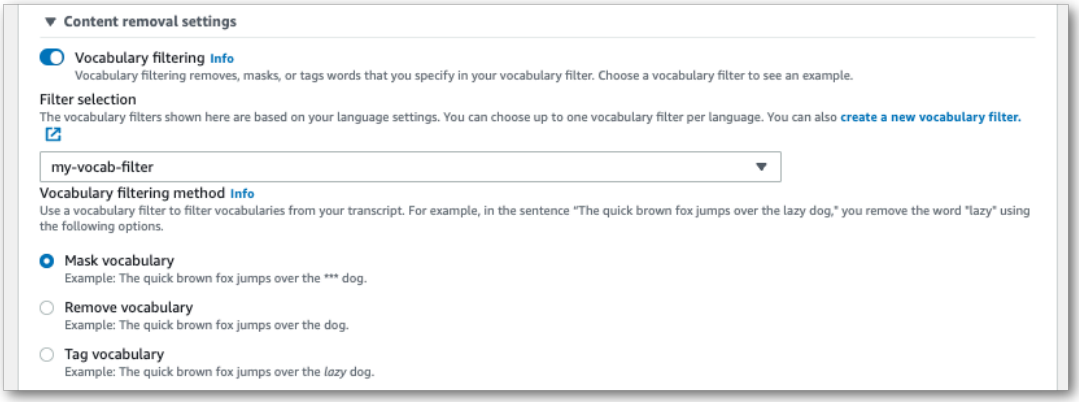
ストリーミング文字起こしでカスタム語彙フィルターを使用するには、以下の例を参照してください。

AWS マネジメントコンソール

1. AWS マネジメントコンソール 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. 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語彙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、カスタム語彙フィルターとフィルター方法を含む 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。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のパラメータは、「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、カスタム語彙フィルターを WebSocket 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。 Amazon Transcribeでの WebSocket ストリームの使用の詳細については、「WebSocket ストリームの設定」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
&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のパラメータは、「共通パラメータ」セクションに記載されています。

有害な音声を検出

有害音声検出は、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、ピアツーピアの対話を伴うソーシャルメディアプラットフォームの管理に役立つように設計されています。有害な音声の使用は、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。有害な音声にフラグを立てることで、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 Transcribe Toxicity Detection は、音声とテキストの両方のキューを活用して、セクシャルハラスメント、ヘイトスピーチ、脅威、虐待、冒涜、侮辱、グラフィックを含む 7 つのカテゴリにまたがる音声ベースの有害コンテンツを特定および分類します。 Amazon Transcribe 有害性検出では、テキストだけでなく、トーンやピッチなどの音声キューも使用して、発話に含まれる有害な意図に的を絞ります。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のコンテンツモデレーションシステムから改善された点です。

Amazon Transcribe は有害な音声にフラグを付け、分類します。これにより、手動で処理する必要があるデータの量を最小限に抑えることができます。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会話を迅速かつ効率的に管理できます。

有害な音声の分類には以下が含まれます。

- 不敬: 無礼、下品、攻撃的な単語やフレーズ、または頭字語を含む言葉。
- ヘイトスピーチ: 人種、民族、性同一性、宗教、性的指向、能力、出身国、その他のアイデンティティグループなど、アイデンティティに基づいて個人またはグループを批判、侮辱、否定する発言。
- セクシャル: 体の一部、身体的特徴、性別への直接的または間接的な言及により、性的関心、活動、性的嗜好を示す発言。
- 侮辱: 屈辱的、嘲笑的、侮辱的、または軽蔑的な言葉を含む発言。この種の発言は、「いじめ」とも呼ばれます。
- 暴力または脅し: 個人または集団に対して苦痛や痛み、敵意を与えることを意図する脅迫的な発言。
- グラフィックスピーチ: 視覚的に説明的で詳細、不快かつ生々しい画像を使った発言。この種の言葉は、受け手の不快感を増幅させるために、意図的に冗長になることが多いのです。
- ハラスメントや虐待: 相手を侮辱したり対象化したりする発言など、受け手の心理的健康に影響を与えることを意図した発言。この種の言葉は、「ハラスメント」とも呼ばれます。

有毒性検出では、音声セグメント (自然な一時停止の間の音声) 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。信頼度スコアの値は 0～1 です。信頼スコアが大きいほど、コンテンツが関連したカテゴリ内の有害な音声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有毒性検出の閾値を設定できます。

> **Note**
>
> 有毒性検出は、米国英語 (en-US) のバッチ文字起こしでのみ利用できます。

JSON 形式の出力例を表示します。

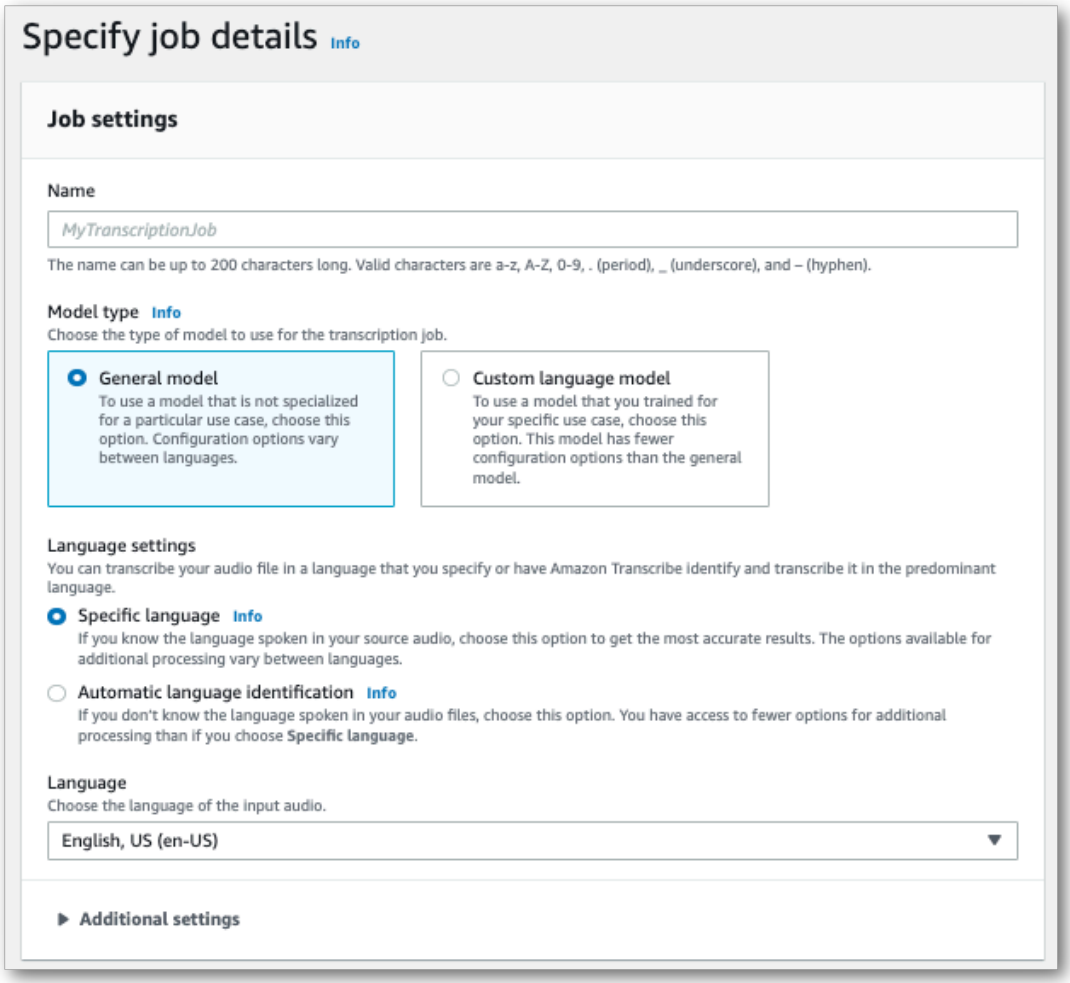
有害な音声検出機能の使用

バッチ文字起こしでの有害な音声の検出機能の使用

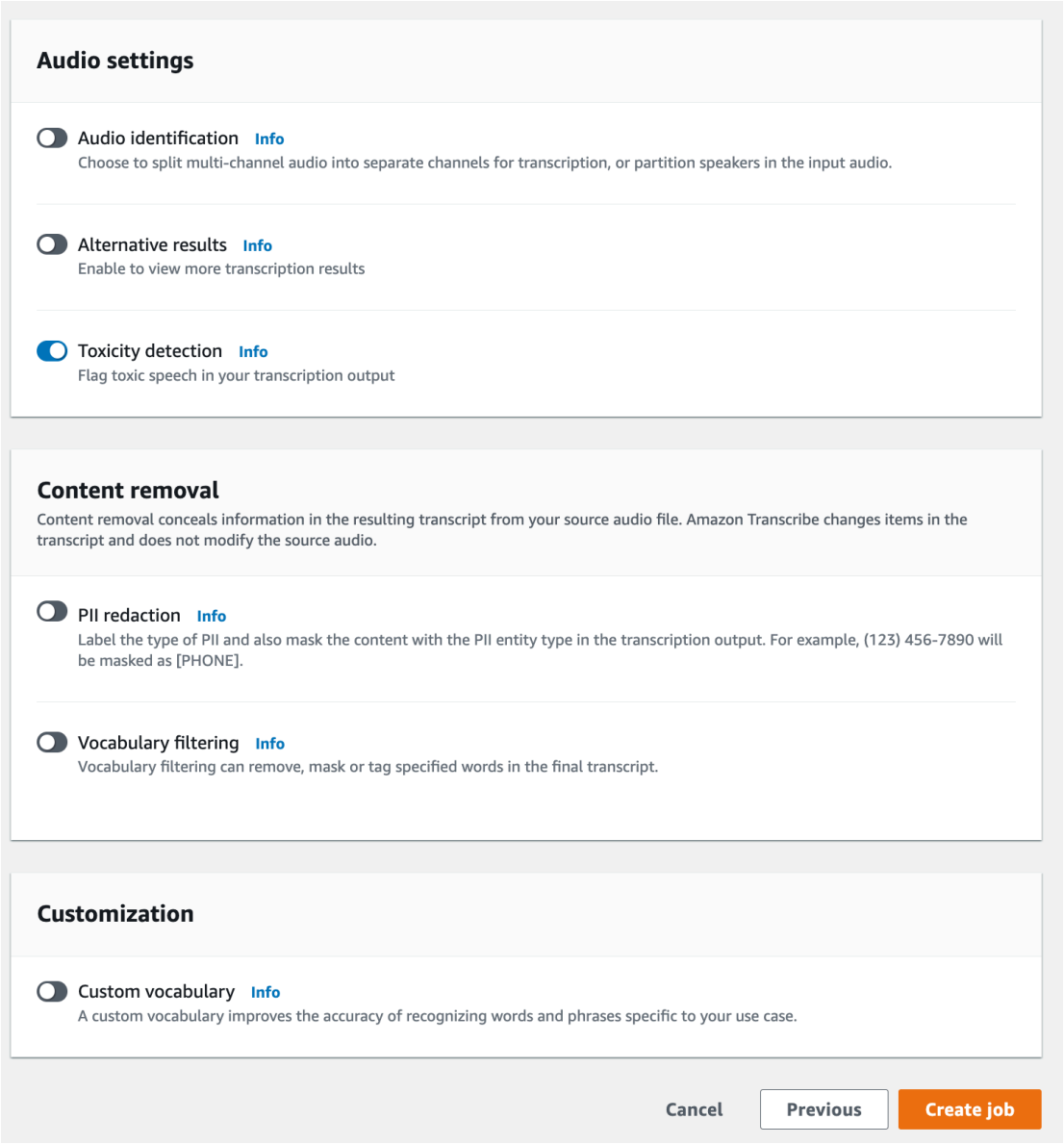
有害な音声の検出とバッチ文字起こしを組み合わせて使用する方法については、以下の例を参照してください。

AWS マネジメントコンソール

1. AWS マネジメントコンソール 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」ページでは、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有害性検出には対応していません。[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、[有害性検出] を選択します。



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。
5. 文字起こしジョブが完了すると、文字起こしジョブの詳細ページにあるダウンロードドロップダウンメニューから、トランスクリプトをダウンロードできます。

AWS CLI

この例では、start-transcription-job コマンドと `ToxicityDetection` パラメータを使用します。詳細については、「`StartTranscriptionJob`」および「`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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
--language-code en-US \
--toxicity-detection ToxicityCategories=ALL
```

別の例として、start-transcription-job コマンドと、有毒性検出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://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入含まれています。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-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) 、 を使用して start_transcription_job メソッドToxicityDetectionの を有効にします。詳細については、「StartTranscriptionJob」および「ToxicityDetection」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。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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)
```

出力の例

有害な音声にはタグが付けられ、文字起こし出力で分類されます。有害な音声の各インスタンスは分類され、信頼スコア (0～1 の値) が割り当てられます。信頼値が大きいほど、コンテンツが指定されたカテゴリ内の有害な音声である可能性が高くなります。

出力例 (JSON)

以下は、分類された有害な音声とそれに関連する信頼スコアを示す JSON 形式の出力例です。

```
{
    "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。編集 Amazon Transcribe できる PII のタイプは、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。各文字起こしの PII リストを確認するには、「バッチジョブで PII を編集する」および「リアルタイムストリームの PII の編集または識別」を参照してください。ストリーミング文字起こしでは、PII を編集せずにフラグを立てるオプションもあります。出力例については、「PII 識別の出力例」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。オリジナルのメディアを削除した場合、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。

バッチ文字起こしを使用した PII リダクションの詳細については、「バッチジョブで PII を編集する」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 リダクションまたは識別の詳細については、「リアルタイムストリームの PII の編集または識別」を参照してください。

> **Important**
>
> リダクション機能は、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、機械学習の予測特性により、 はトランスクリプト内の機密データのすべてのインスタンスを特定および削除できない Amazon Transcribe 場合があります。編集された出力がお客様のニーズを満たすものであることを確認するために、出力内容を見直すことを強くおすすめします。
>
> リダクション機能は、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。

の秘匿化機能のビデオチュートリアルについては、 Amazon Transcribe「コンテンツ秘匿化を使用して PII を識別および編集する」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文字起こし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文字起こしの本文[PII]の Amazon Transcribe に置き換えます。また、文字起こし出力の単語ごとの部分で、編集された PII の種類を表示することもできます。出力サンプルについては、「編集された出力例 (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、英語の方言: 米国 (en-US)、スペイン語の方言: 米国 (es-US)、フランス語の方言: フランス語 (fr-FR)、カナダ (fr-CA)、ドイツの方言: ドイツ (de-DE)、スイス (de-CH)、イタリアの方言: イタリア (it-IT)、ポルトガルの方言: ポルトガル (pt-PT)、ブラジル () で利用できますpt-BR。リダクションは言語識別と互換性がありません。

秘匿化されたトランスクリプトと秘匿化されていない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。 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の Amazon S3 バケットにトランスクリプト Amazon Transcribe 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。
AGE	個人の年齢 (時間の数値や単位を含む)。たとえば、「40 歳」というフレーズでは、「40 歳」を年齢として Amazon Transcribe 認識しています。
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。
AWS_ACCESS_KEY	シークレットアクセスキーに関連付けられている一意の識別子。アクセスキー ID とシークレットアクセスキーを使用して、プログラムによる AWS リクエストに暗号で署名します。

PII タイプ	説明
AWS_SECRET_KEY	アクセスキーに関連付けられた一意の識別子。アクセスキー ID とシークレットアクセスキーを使用して、プログラムによる AWS リクエストに暗号で署名します。
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。
CA_HEALTH_NUMBER	カナダの医療保健番号で、個人が医療給付を受けるために必要な 10 桁の固有識別番号です。
CA_SOCIAL_INSURANCE_NUMBER	カナダ社会保険番号 (SIN) は 9 桁の一意の識別子であり、個人が政府のプログラムや特典にアクセスするために必要です。
CREDIT_DEBIT_CVV	VISA、MasterCard、Discover のクレジットカードとデビットカードに記載されている 3 桁のカード確認コード (CVV)。American Express のク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。たとえば、 は 01/21、01/2021、Jan 2021 などの有効期限を認識 Amazon Transcribe できます。

PII タイプ	説明
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、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。
DATE_TIME	日付には、年、月、日、曜日、または時刻を含めることができます。たとえば、 は「2020 年 1 月 19 日」または「午前 11 時」を日付として Amazon Transcribe 認識します。 Amazon Transcribe は部分的な日付、日付範囲、および時間間隔を認識します。また「the 1990s (1990 年代) 」などの 10 年間も認識されます。
DRIVER_ID	運転免許証に割り当てられる番号。運転免許証は、個人が公道で1台または複数の自動車を運転することを許可する公式文書です。運転免許証番号は英数字です。
EMAIL	efua.owusu@email.com などのメールアドレス。
INTERNATIONAL_BANK_ACCOUNT_NUMBER	国際銀行口座番号の形式は国によって異なります。詳細については、www.iban.com/structure を参照してください。
IP_ADDRESS	IPv4 アドレス (198.51.100.0 など)。
LICENSE_PLATE	車両のナンバープレートは、車両が登録されている州または国によって発行されます。乗用車の形式は通常 5 ~ 8 桁で、大文字と数字で構成されます。形式は発行国または国の所在地によって異なります。

PII タイプ	説明
MAC_ADDRESS	メディアアクセスコントロール (MAC) アドレスは、ネットワークインターフェイスコントローラー (NIC) に割り当てられる一意の識別子です。
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 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。たとえば、 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。
PASSPORT_NUMBER	個人のパスポートに割り当てられた一意の識別子。形式は通常、文字と数字の組み合わせを含み、国によって異なります。
PASSWORD	パスワードとして使用される英数字の文字列 (「*very20special#pass*」など)。
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)。
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。

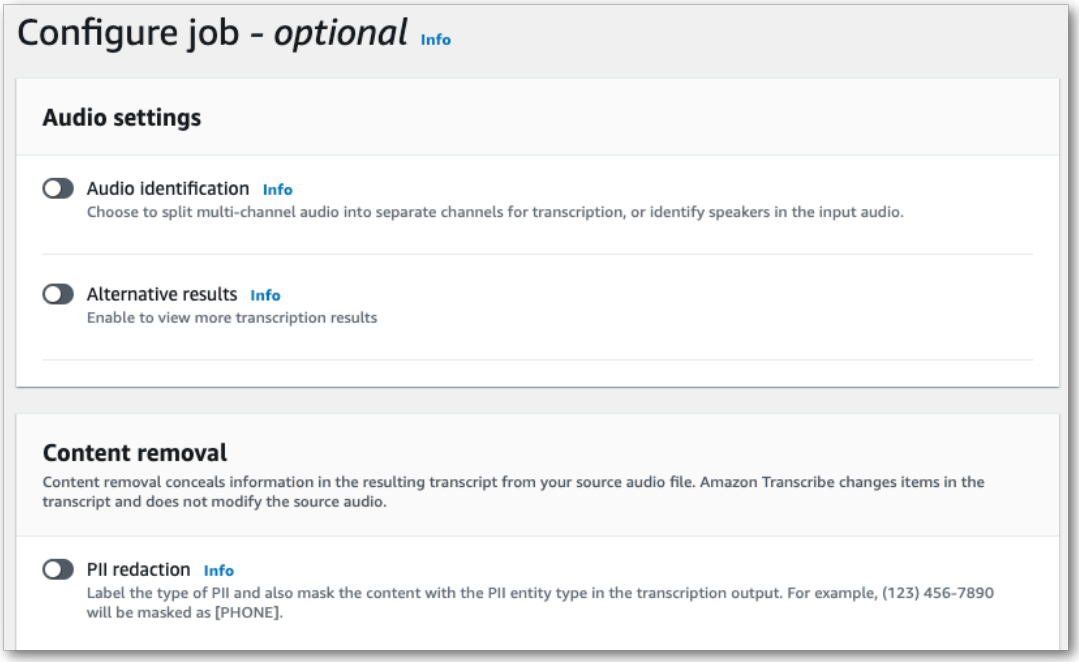
PII タイプ	説明
SWIFT_CODE	SWIFT コードは、特定の銀行または支店を指定するために使用する銀行識別コード (BIC) の標準形式です。銀行は、これらのコードを国際電信送金などの送金に使用します。SWIFT コードは 8 文字または 11 文字で構成されます。11 桁のコードは特定のブランチを指し、8 桁のコード (または「XXX」で終わる 11 桁のコード) は本社または主要オフィスを指します。
URL	ウェブアドレス (www.example.com など)。
US_INDIVIDUAL_TAX_IDENTIFICATION_NUMBER	米国個人納税者識別番号 (ITIN) は、「9」で始まる 9 桁の数字で、4 桁目に「7」または「8」が含まれます。ITIN は、3 桁目と 4 桁目の後にスペースまたはダッシュでフォーマットできます。
USERNAME	ログイン名、スクリーンネーム、ニックネーム、ハンドル名など、アカウントを識別するユーザー名。
VEHICLE_IDENTIFICATION_NUMBER	車両識別番号 (VIN) は、車両を一意に識別します。VIN の内容と形式は ISO 3779 仕様で定義されています。VIN のコードと形式は国ごとに異なります。

バッチ文字起こしジョブは AWS マネジメントコンソール、、 AWS CLI、または AWS SDK を使用して開始できます。

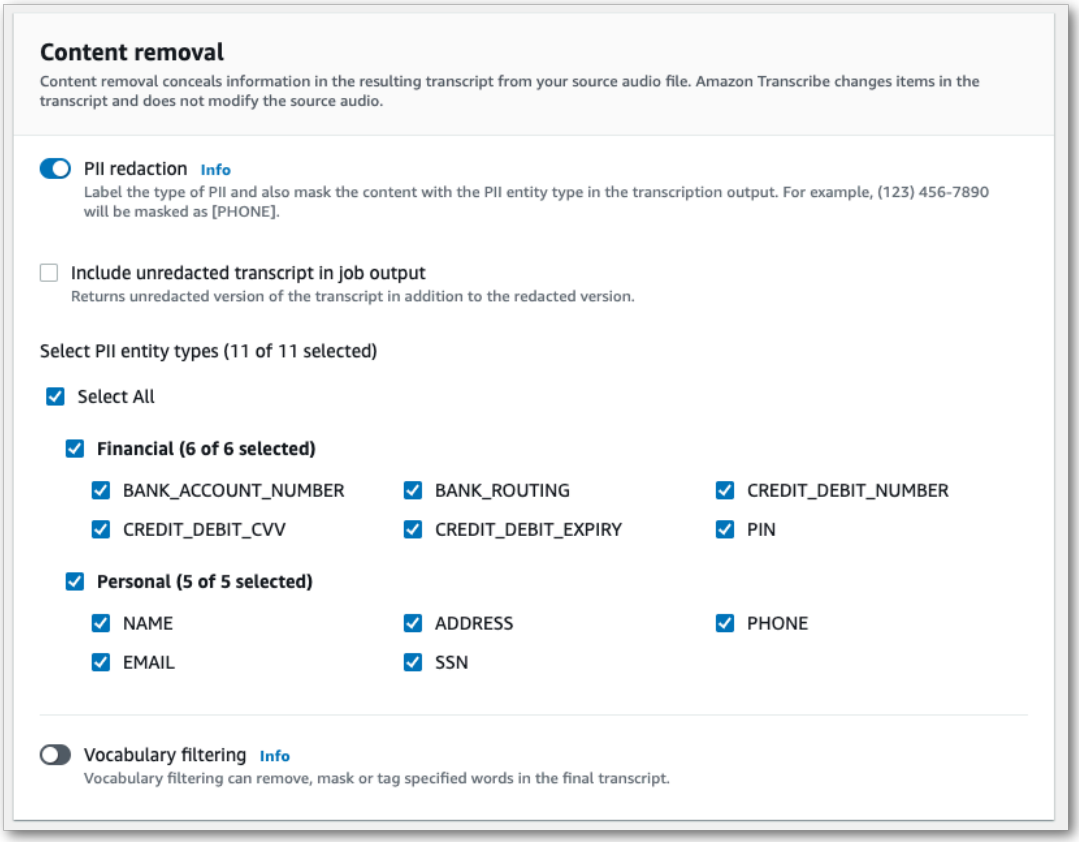
AWS マネジメントコンソール

1. AWS マネジメントコンソール 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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
--language-code en-US \
--content-redaction
 RedactionType=PII,RedactionOutput=redacted,PiiEntityTypes=NAME,ADDRESS,BANK_ACCOUNT_NUMBER
```

以下は start-transcription-job メソッドを使用した別の例で、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、次のリクエストボディが含まれています。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
      "MediaFileUri":  "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-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 AWS SDK for Python (Boto3) 、 を使用して、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 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。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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、アジア AWS リージョンパシフィック (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) 、および米国西部 (北カリフォルニア) でのみサポートされています。

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Entitiesオブジェクトの文字起こし結果で PII に Amazon Transcribe 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、スコットランド語 (en-AB)、オーストラリア (en-AU)、カナダ (en-CA)、英国 (en-GB)、アイルランド (en-IE)、インド (en-IN)、ニュージーランド (en-NZ)、米国 (en-US)、ウェールズ (en-WL)、南アフリカ (en-ZA)、スペイン語方言: 米国 (es-US)、スペイン (es-ES)、フランス語方言: フランス語 (fr-FR)、カナダ (fr-CA)、ポルトガル方言: ポルトガル (pt-PT)、ブラジル (pt-BR)、イタリア方言: イタリア (it-IT)、ドイツ方言: ドイツ (de-DE)、スイス (de-CH) で利用できます。

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、MasterCard、Discover のクレジットカードとデビットカードに記載されている 3 桁のカード確認コード (CVV)。American Express

PII タイプ	説明
	のク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。たとえば、 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、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 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。たとえば、 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)。

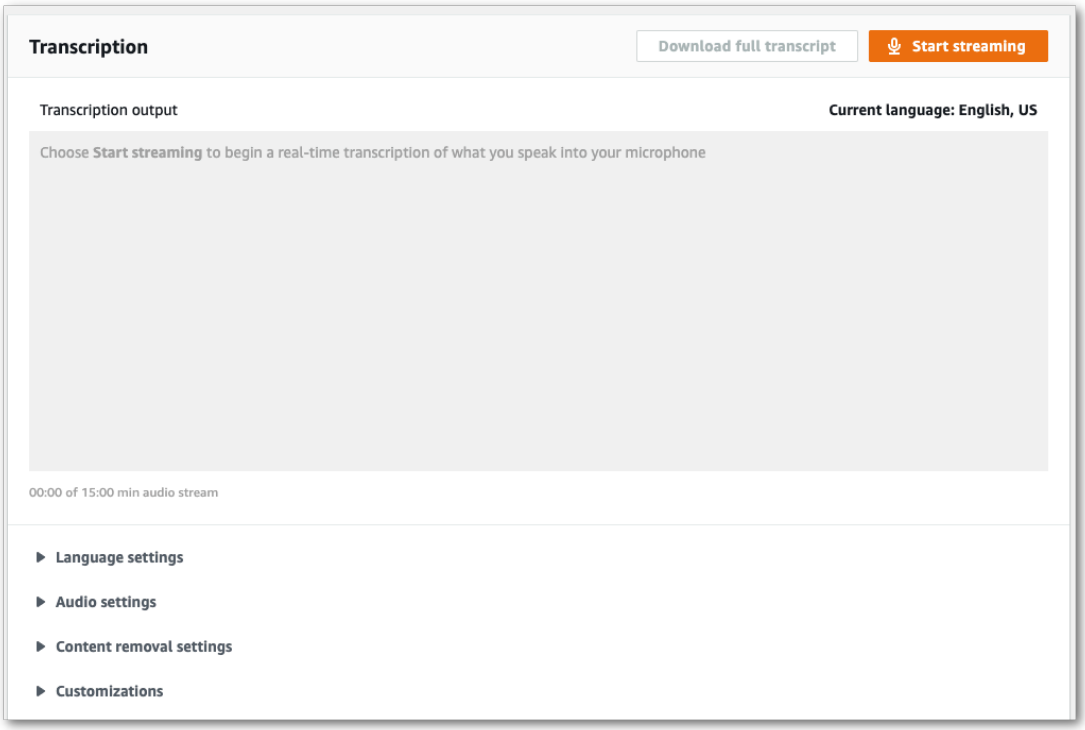
PII タイプ	説明
SSN	社会保障番号 (SSN) は、米国市民、永住者、および一時的な労働居住者に発行される 9 桁の番号です。 は、最後の 4 桁のみが存在する場合に社会保障番号 Amazon Transcribe も認識します。
AGE	個人の年齢 (時間の数値や単位を含む)。たとえば、「40 歳」というフレーズでは、「40 歳」を年齢として Amazon Transcribe 認識しています。
DATE_TIME	日付には、年、月、日、曜日、または時刻を含めることができます。たとえば、 は「2020 年 1 月 19 日」または「午前 11 時」を日付として Amazon Transcribe 認識します。 Amazon Transcribe は部分的な日付、日付範囲、および時間間隔を認識します。また「the 1990s (1990 年代) 」などの 10 年間も認識されます。
LICENSE_PLATE	車両のナンバープレートは、車両が登録されている州または国によって発行されます。乗用車の形式は通常 5 ~ 8 桁で、大文字と数字で構成されます。形式は発行国または国の所在地によって異なります。
PASSPORT_NUMBER	個人のパスポートに割り当てられた一意の識別子。形式は通常、文字と数字の組み合わせを含み、国によって異なります。
PASSWORD	パスワードとして使用される英数字の文字列 (「*very20special#pass*」など)。
USERNAME	ログイン名、スクリーンネーム、ニックネーム、ハンドル名など、アカウントを識別するユーザー名。

PII タイプ	説明
VEHICLE_IDENTIFICATION_NUMBER	車両識別番号 (VIN) は、車両を一意に識別します。VIN の内容と形式は ISO 3779 仕様で定義されています。VIN のコードと形式は国ごとに異なります。

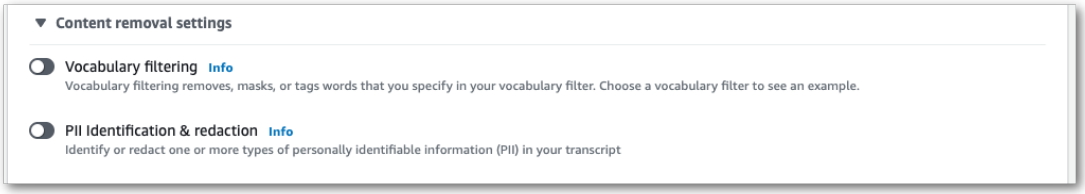
ストリーミング文字起こしは AWS マネジメントコンソール、、WebSocket、または HTTP/2 を使用して開始できます。

AWS マネジメントコンソール

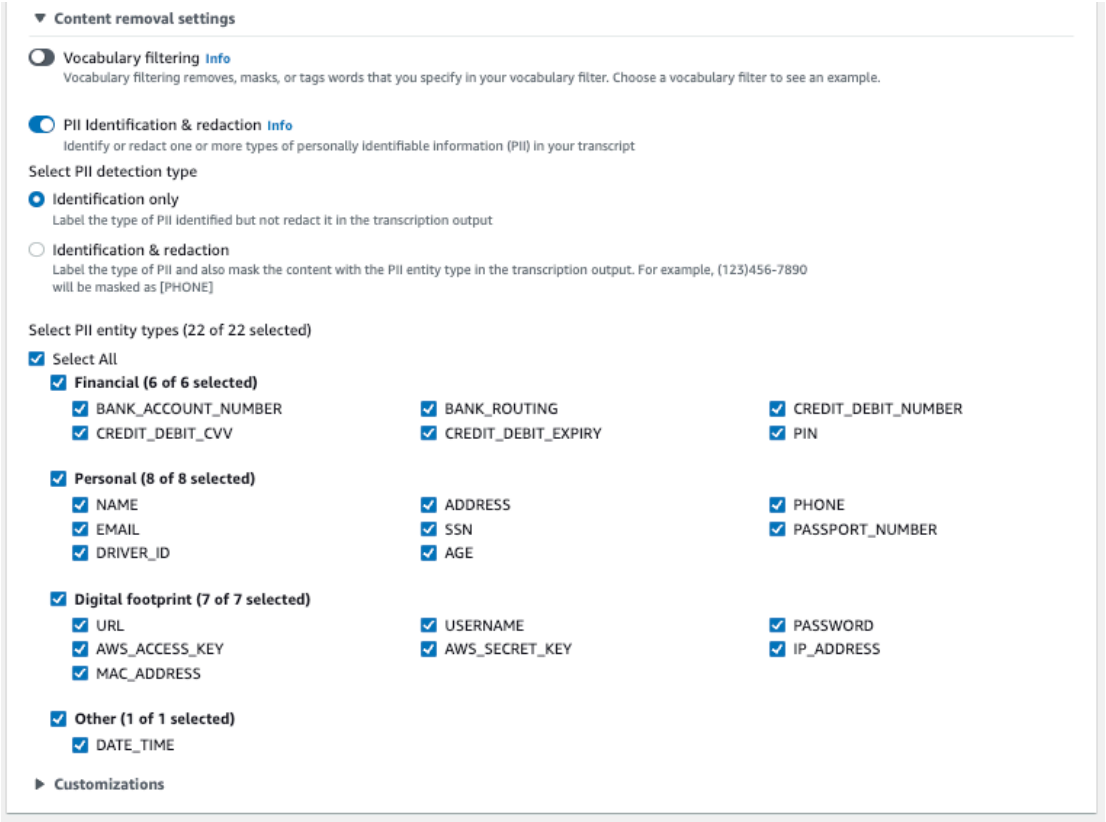
1. AWS マネジメントコンソール にサインインします。
2. ナビゲーションペインで、[リアルタイム文字起こし] を選択します。コンテンツ削除の設定 にスクロールして、最小化されている場合はこのフィールドを展開します。



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 ストリームの設定。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 AWS API オペレーションに共通のパラメータは、「共通パラメータ」セクションに記載されています。

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。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 AWS API オペレーションに共通のパラメータは、「共通パラメータ」セクションに記載されています。

> **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) 、および米国西部 (オレゴン) 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。

コンテンツリダクション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、pronunciation の type 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 には 1.0 の confidence があります。この信頼値は、このトランスクリプトで発せられた単語 Amazon Transcribe が「良い」であることを が 100% 確信していることを示します。[PII] タグの confidence 値は、リダクションのフラグを付けた発話が確実に PII であるという信頼度を示します。次のトランスクリプト出力では、 confidenceの は、トランスクリプトで編集したエンティティ Amazon Transcribe が PII であることを 99.99% 確信している0.9999ことを示します。

編集された出力例 (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": [
            {
                "id": 0,
                "start_time": "2.86",
                "end_time": "3.35",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Good"
                    }
                ],
                "type": "pronunciation"
            },
            Items removed for brevity
            {
                "id": 8,
                "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": [
            {
                "id": 0,
                "start_time": "2.86",
                "end_time": "3.35",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Good"
                    }
                ],
                "type": "pronunciation"
            },
            Items removed for brevity
            {
                "id": 8,
                "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、ストリーミング文字起こしジョブで使用できる追加機能です。特定された PII は、各セグメントの Entities 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 Transcribe は、ビデオ字幕として使用するために WebVTT (*.vtt) および SubRip (*.srt) 出力をサポートします。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。文字起こしリクエストで字幕を有効にしても、音声に会話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 Transcribe は、字幕出力0に のデフォルトの開始インデックスを使用します。これは、 のより広く使用されている値とは異なります1。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、 AWS マネジメントコンソール または API リクエストで指定できます。

誤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、どの開始インデックスが必要かを確認してください。どの値を使うべきかわからない場合は、1 を選択することをおすすめします。詳細については、「`Subtitles`」を参照してください。

字幕でサポートされる機能は次のとおりです。

- コンテンツのリダクション — 編集されたコンテンツは、字幕と通常の文字起こし出力ファイルで「PII」として反映されます。音声は変更されません。
- 語彙フィルター — 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。
- 話者ダイアライゼーション — 特定の字幕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。これは WebVTT 形式と subRip 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、文字起こし出力と同じ Amazon S3 場所に保存されます。

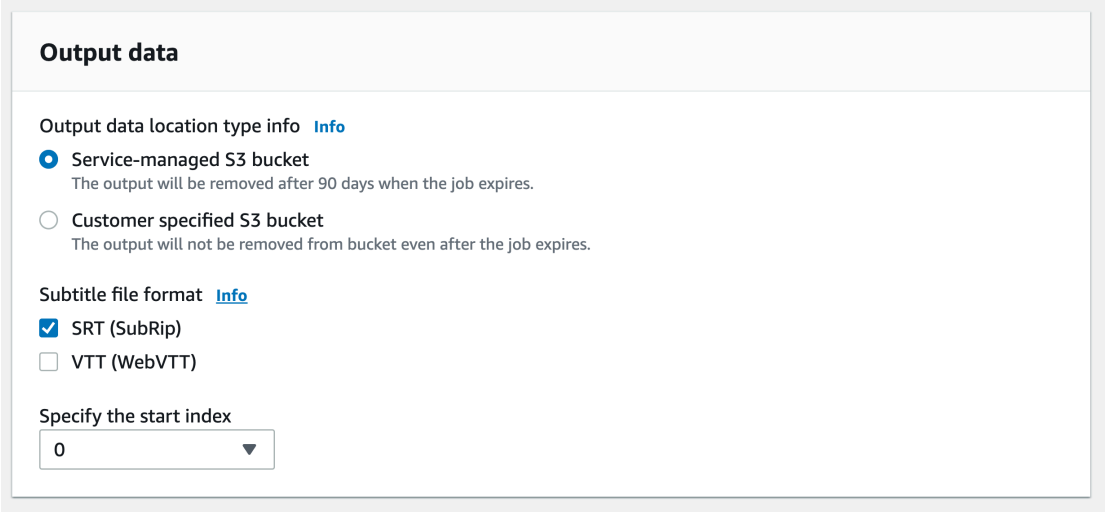
字幕作成のビデオチュートリアルについては、「Amazon Transcribe Video Snacks: コードを記述せずに動画字幕を作成する」を参照してください。

字幕ファイルの生成

AWS マネジメントコンソール、AWS CLI、または AWS SDK 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 マネジメントコンソール

1. AWS マネジメントコンソール 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。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、開始インデックスの値を選択します。 Amazon Transcribe デフォルトは ですが0、1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 を選択することをおすすめします。



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 Subtitles パラメータを使用します。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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
--language-code en-US \
--subtitles Formats=vtt,srt,OutputStartIndex=1
```

別の例として、文字起こしジョブの開始 コマンド、およびそのジョブ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 には、次のリクエストボディが入含まれています。

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-bucket",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "Subtitles": {
        "Formats": [
            "vtt","srt"
        ],
        "OutputStartIndex": 1
   }
}
```

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 Subtitles引数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。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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、顧客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、通話後の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。

Call Analytics で利用可能な機能と、 Amazon Transcribe および Amazon Transcribe Medical の機能を比較するには、機能表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。
- ジョブキューイング 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。周波数が 8kHz を超えるオーディオファイルの場合、オーディオ継続時間は 2 時間以下である必要があります。特定の圧縮された非 WAV オーディオファイル形式では、ファイルサイズ制限がより小さくなる可能性があることにご注意ください。
- 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 のデモ」を参照してください。また、トランスクリプトを読みやすい形式に変換するための JSON to Word document アプリケーションも提供しています。

利用可能なリージョンとクォータ

コール分析は、以下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

通話の生成要約では、通話全体の簡潔な概要を作成し、通話の理由、問題の解決ステップ、次のステップなどの主な要素を特定します。

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。指定された期間に会話の 50% 以上が (2 人のスピーカー間のback-and-forthに) 指定されたセンチメントと一致する場合、 はセンチメントの一致 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**
>
> 通話後の PII リダクションは、米国英語 (`en-US`) およびスペイン語 (`es-US`) でサポートされます。

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。

感情分析は初期状態で機能するため、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には対応していません。

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。

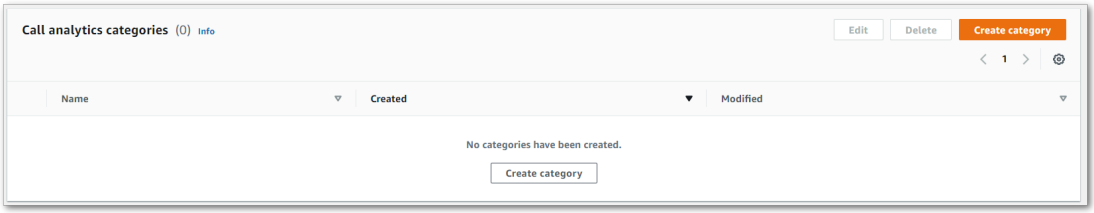
通話後カテゴリとリアルタイムカテゴリ

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。

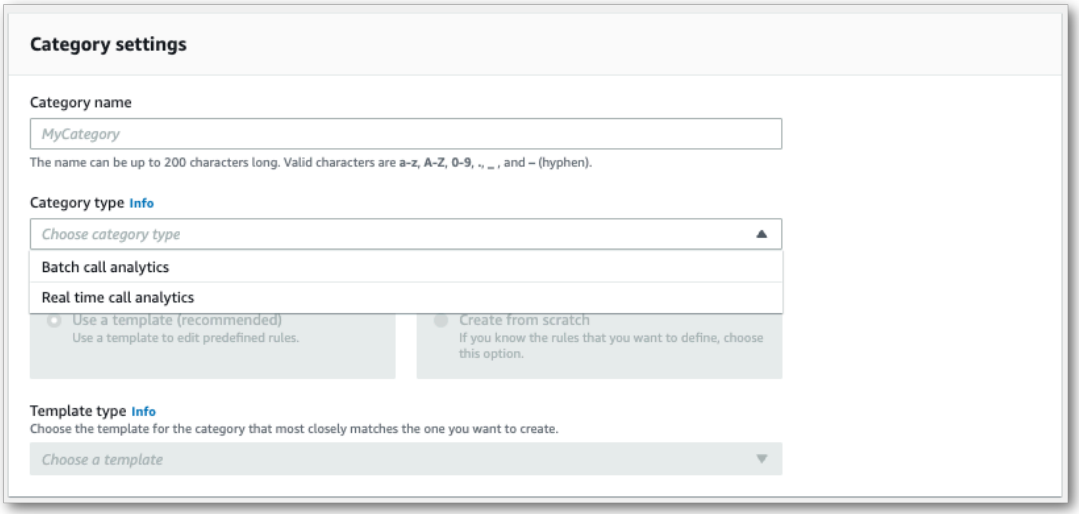
通話後分析用の新しいカテゴリを作成するには、AWS マネジメントコンソール、AWS CLI、または AWS SDK を使用できます。例については以下を参照してください。

AWS マネジメントコンソール

1. ナビゲーションペインで、Amazon Transcribe 分析の呼び出し Amazon Transcribe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。

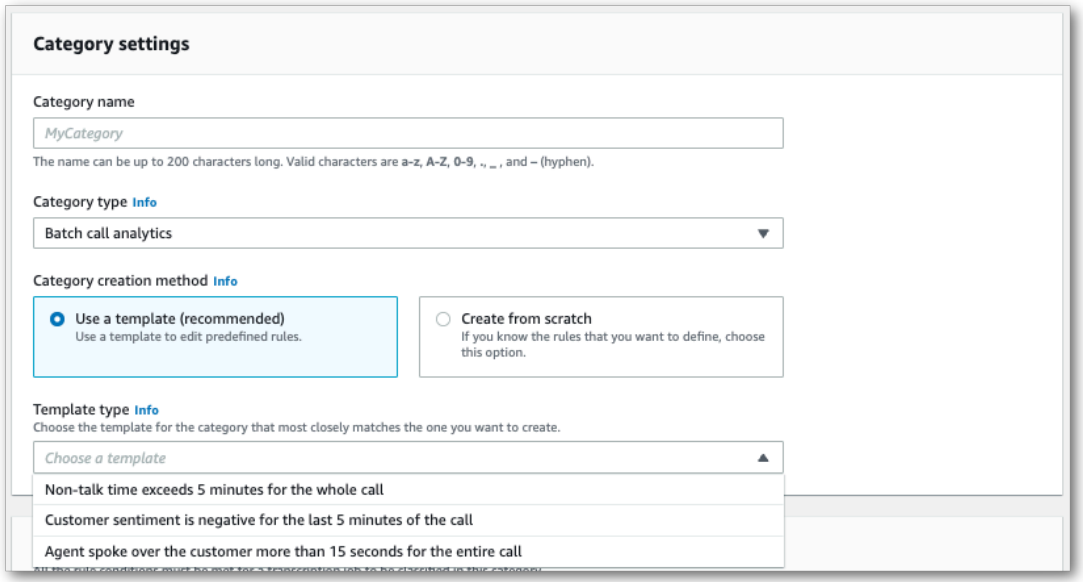


3. [カテゴリの作成ページ] が表示されます。カテゴリの名前を入力し、カテゴリタイプのドロップダウンメニューで [バッチコール分析] を選択します。

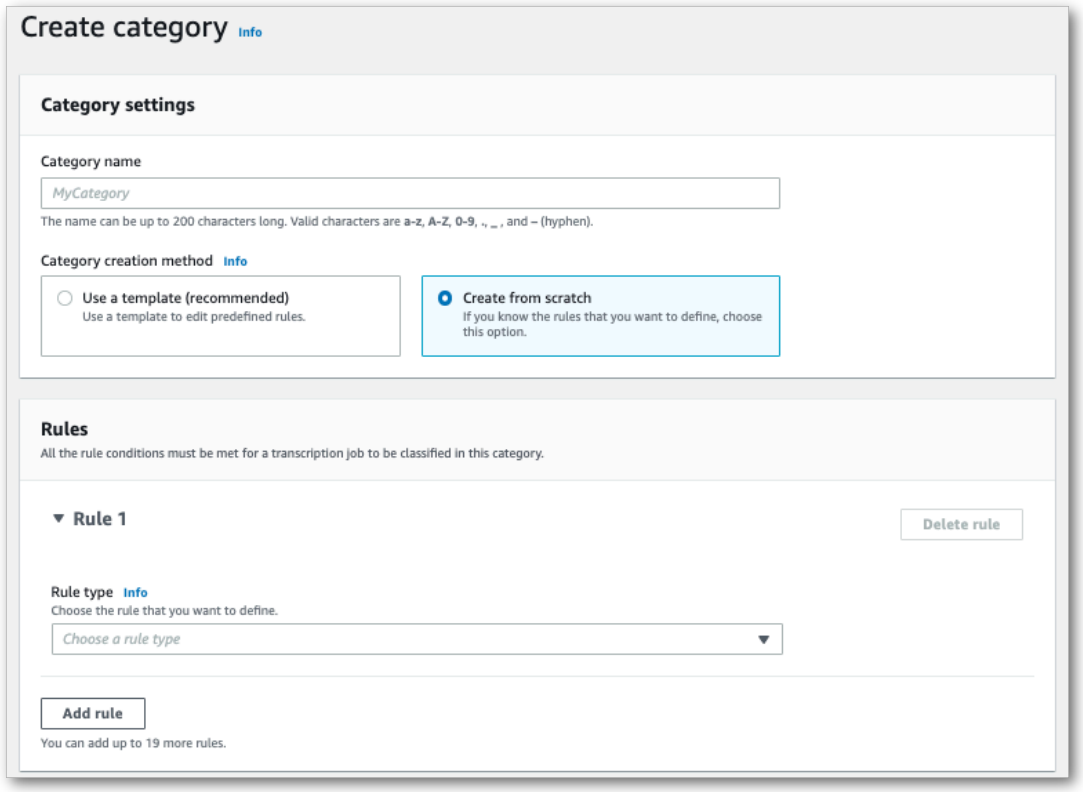


4. テンプレートを選択してカテゴリを作成することも、一から作成することもできます。

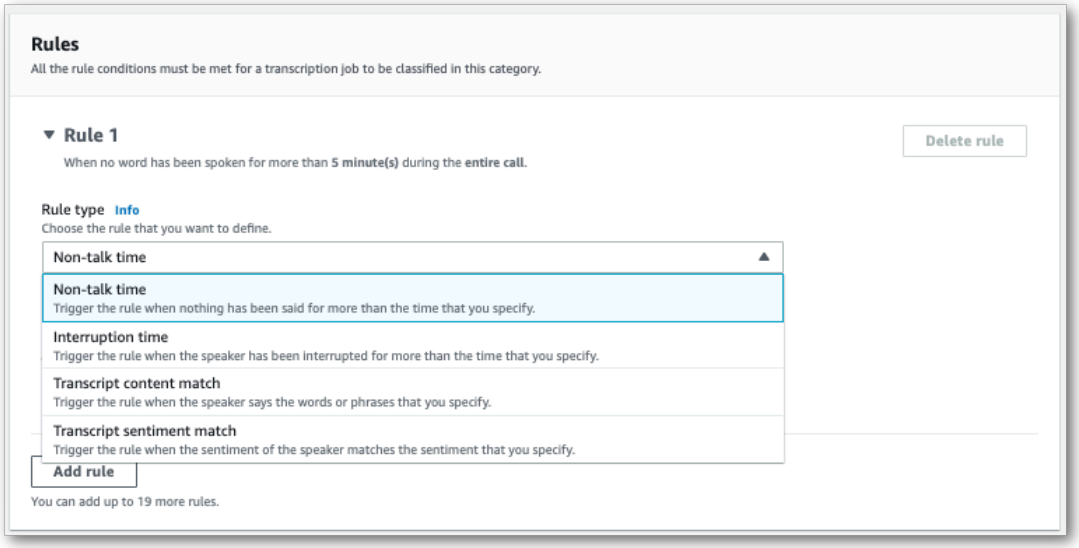
   テンプレートを使用する場合: [テンプレートを使用する (推奨)] を選択し、必要なテンプレートを選択してから [カテゴリの作成] を選択します。



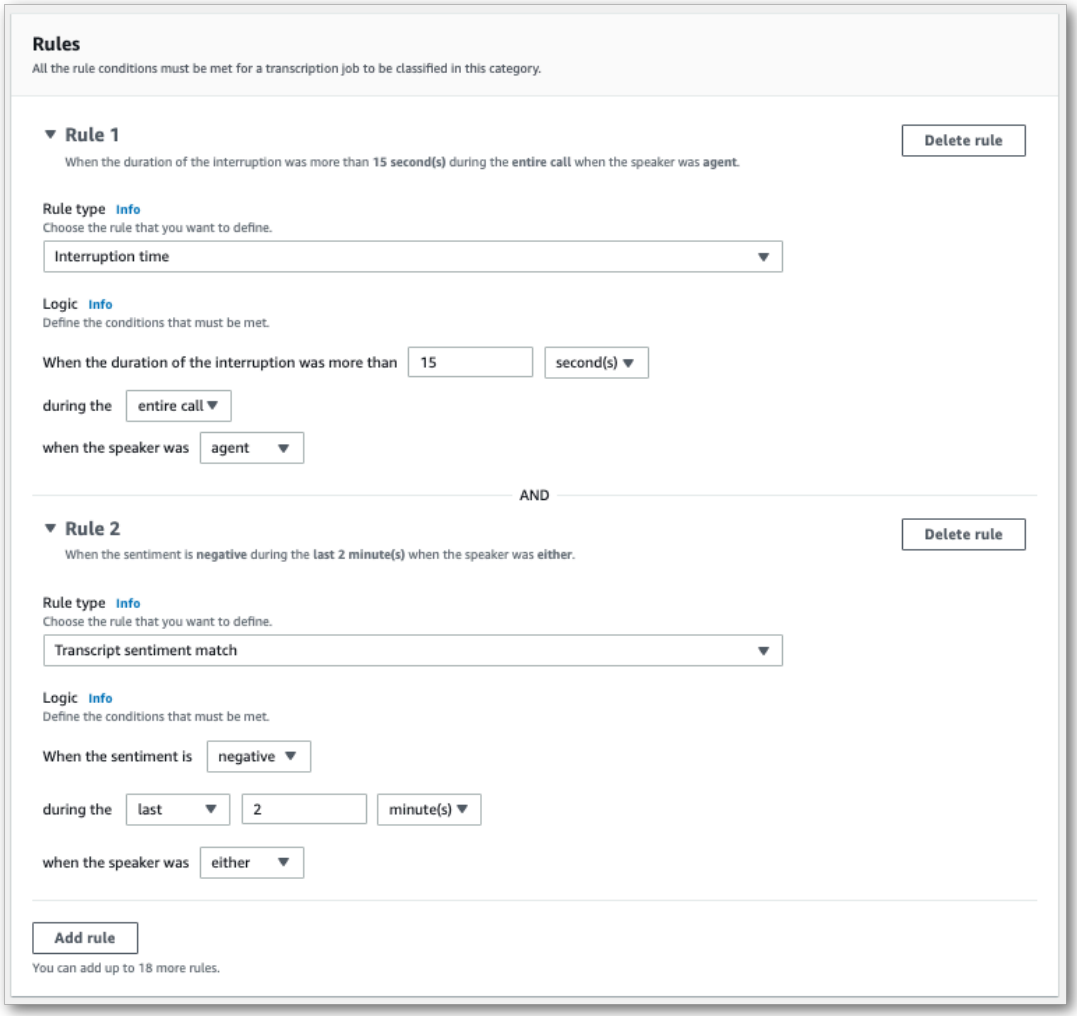
5. カスタムカテゴリを作成する場合: [最初から作成] を選択します。



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ボディを使用しました。

```
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、次のリクエストボディが含まれています。

```
{
  "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 AWS SDK for Python (Boto3) 、 を使用して、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。

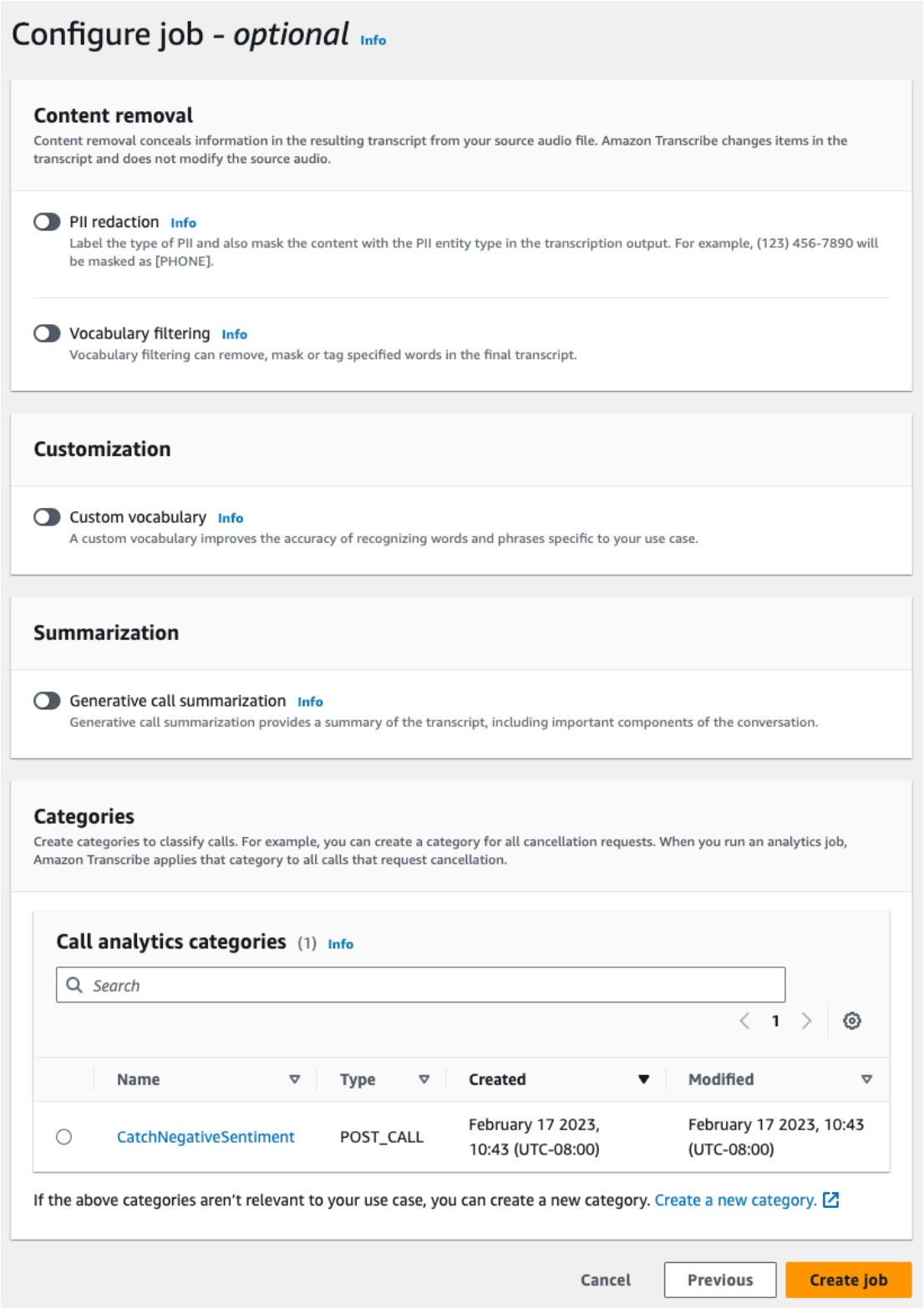
1 つ以上のカテゴリを作成し、音声が少なくとも 1 つのカテゴリですべてのルールに一致する場合、Amazon Transcribe は一致するカテゴリで出力にフラグ付けを行います。カテゴリを使用しないことを選択した場合、または音声がカテゴリで指定したルールに一致しない場合、トランスクリプトにフラグ付けは行われません。

通話後分析文字起こしを開始するには、AWS マネジメントコンソール、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。

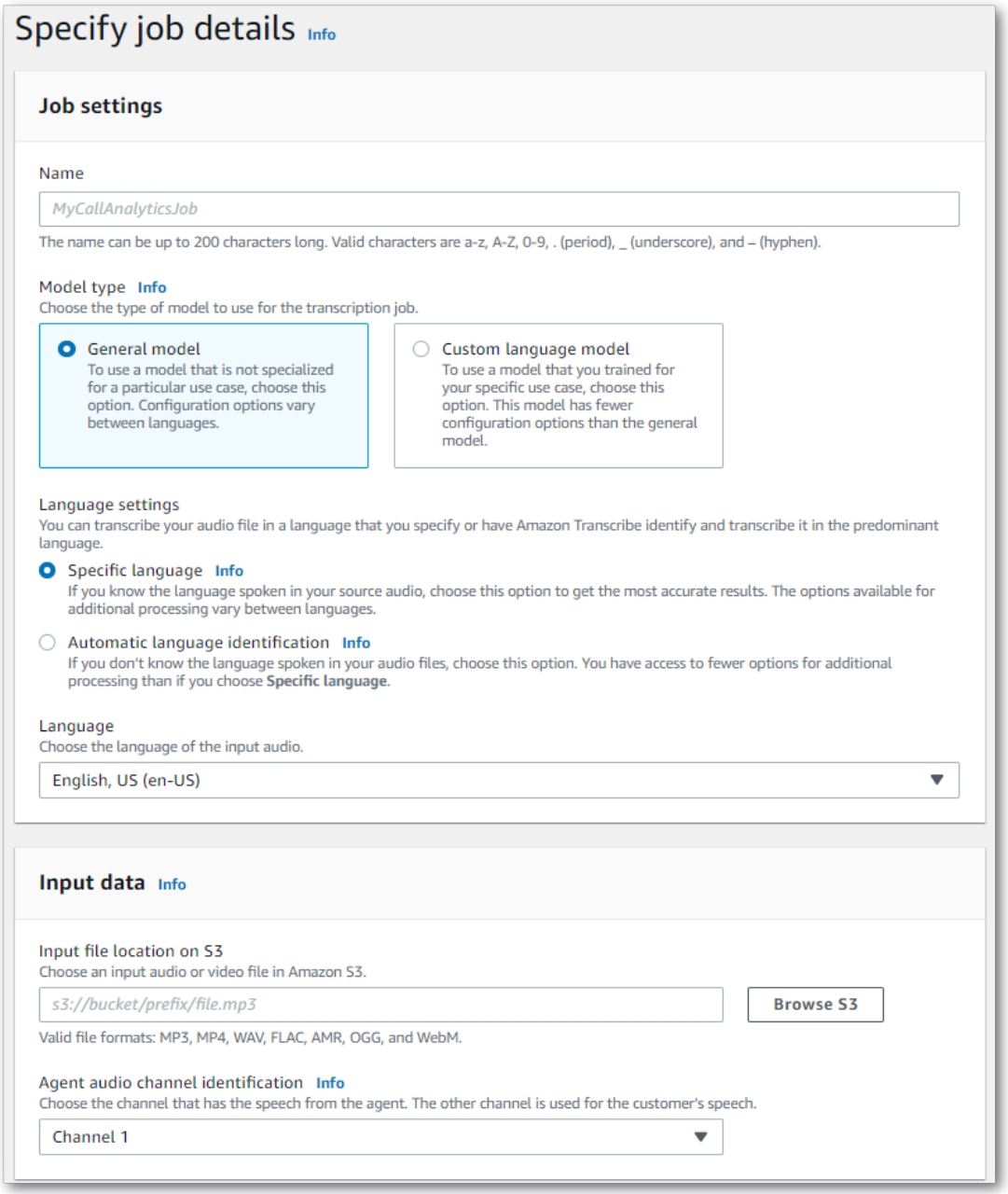
AWS マネジメントコンソール

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。

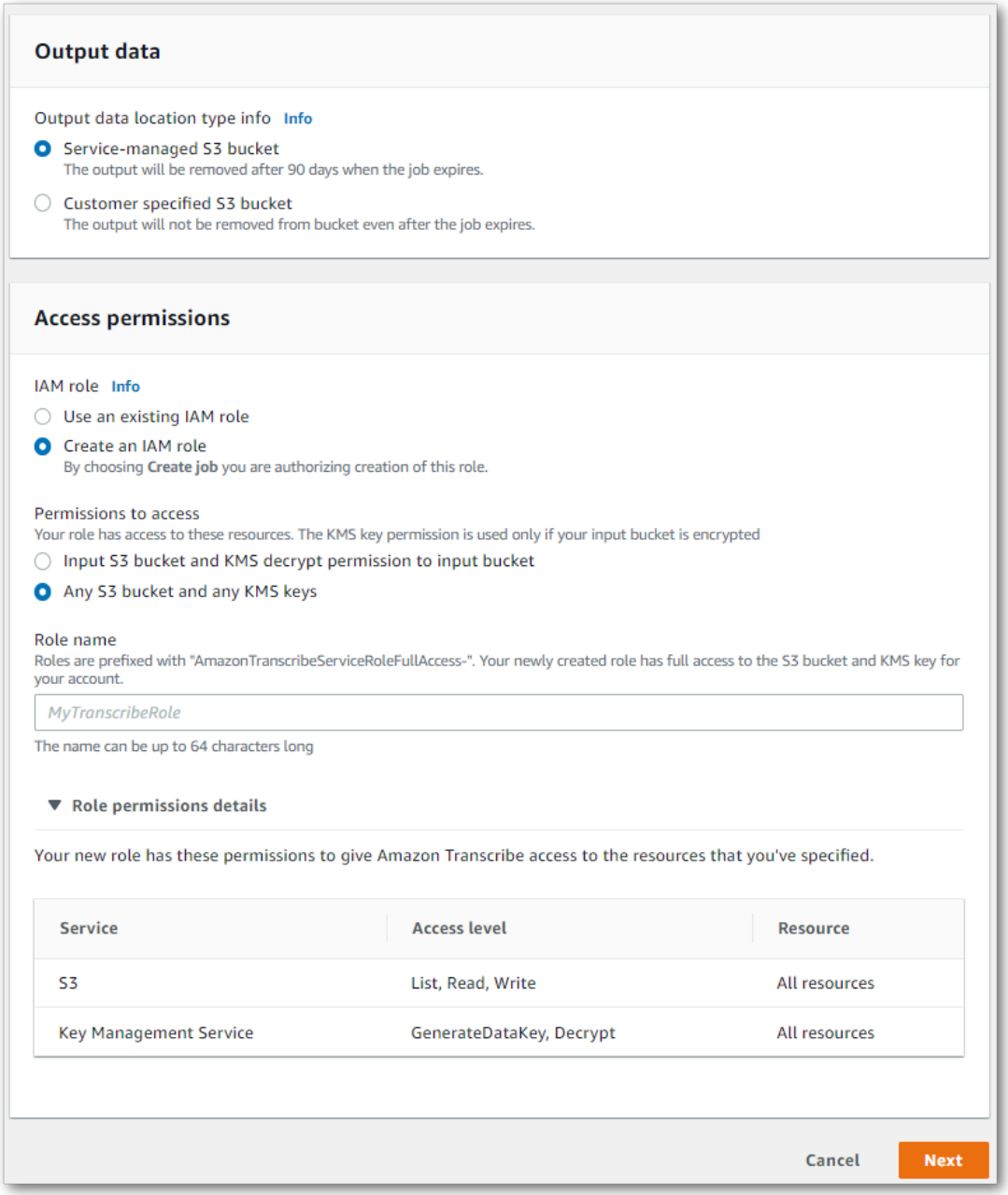
1. ナビゲーションペインの「分析 Amazon Transcribe の呼び出し」で、「分析ジョブの呼び出し」を選択します。
2. [ジョブの作成] を選択します。



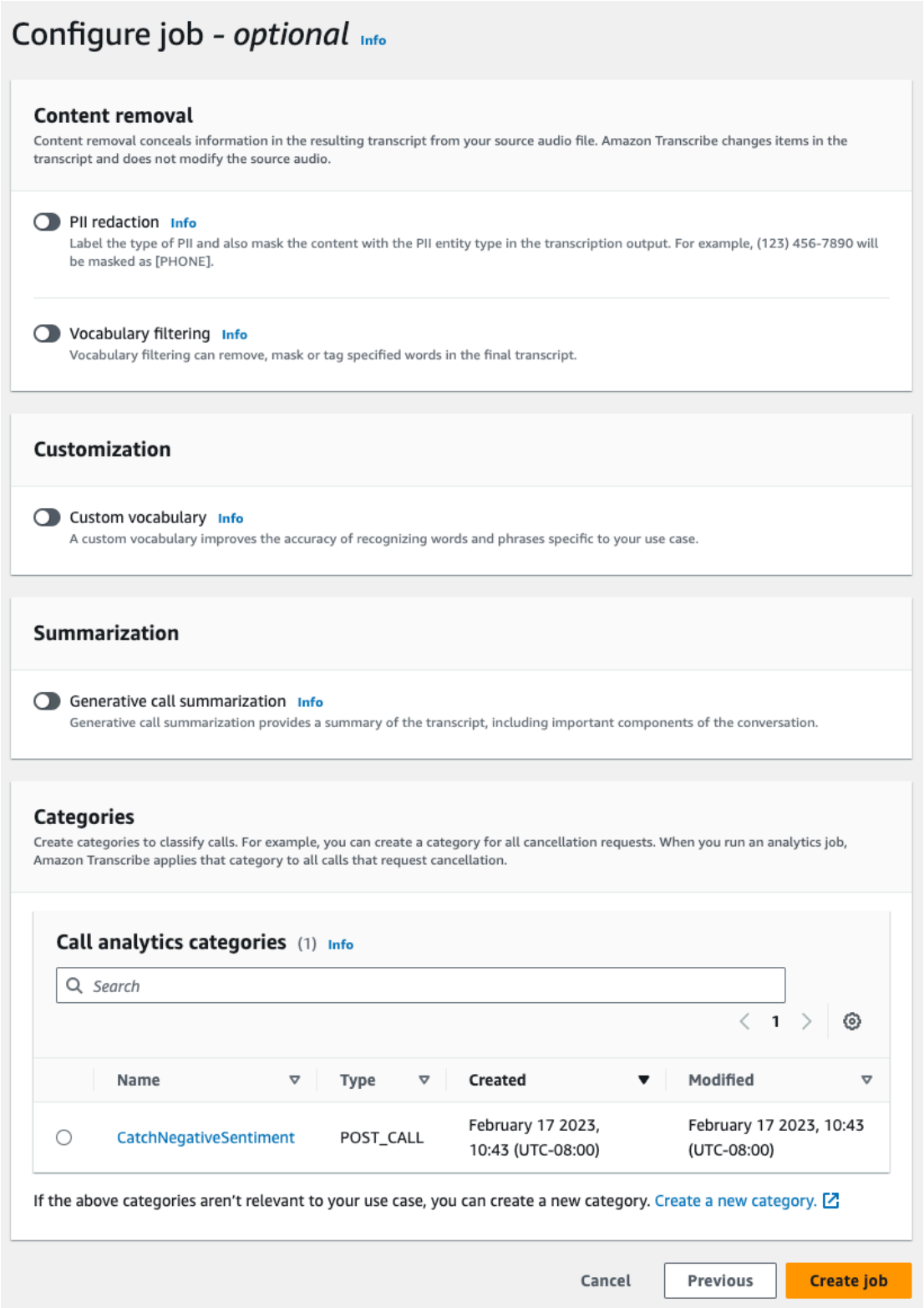
3. [ジョブ詳細の指定] ページで、入力データの場所など、コール分析ジョブに関する情報を提供します。



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] を選択します。
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。



6. [ジョブの作成] を選択します。

AWS CLI

この例では、start-call-analytics-job のコマンドと 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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-location s3://amzn-s3-demo-bucket/my-output-files/ \
--data-access-role-arn arn:aws:iam::111122223333:role/ExampleRole \
--channel-definitions ChannelId=0,ParticipantRole=AGENT
 ChannelId=1,ParticipantRole=CUSTOMER
```

次に、start-call-analytics-job コマンドと、そのジョブのコール分析を有効に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 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、次のリクエストボディが含まれています。

```
{
      "CallAnalyticsJobName": "my-first-call-analytics-job",
      "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole",
      "Media": {
          "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
      },
      "OutputLocation": "s3://amzn-s3-demo-bucket/my-output-files/",
      "ChannelDefinitions": [
          {
              "ChannelId": 0,
              "ParticipantRole": "AGENT"
          },
          {
              "ChannelId": 1,
              "ParticipantRole": "CUSTOMER"
          }
      ]
```

```
}
```

AWS SDK for Python (Boto3)

この例では AWS SDK for Python (Boto3) 、 を使用して 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 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。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
output_location = "s3://amzn-s3-demo-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

通話後分析トランスクリプトは、セグメントごとにターンバイターン形式で表示されます。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、分析ジョブは概要の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_SAFETY_GUIDELINES を返します。

エラーコードは、出力の AnalyticsJobDetails 内の Skipped セクションにあります。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",
    ...
}
```

センチメント分析

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。

- 定性的なターンバイターン感情値

```
"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、オプション機能 - 通話の生成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": []
    },
    ...
}
```

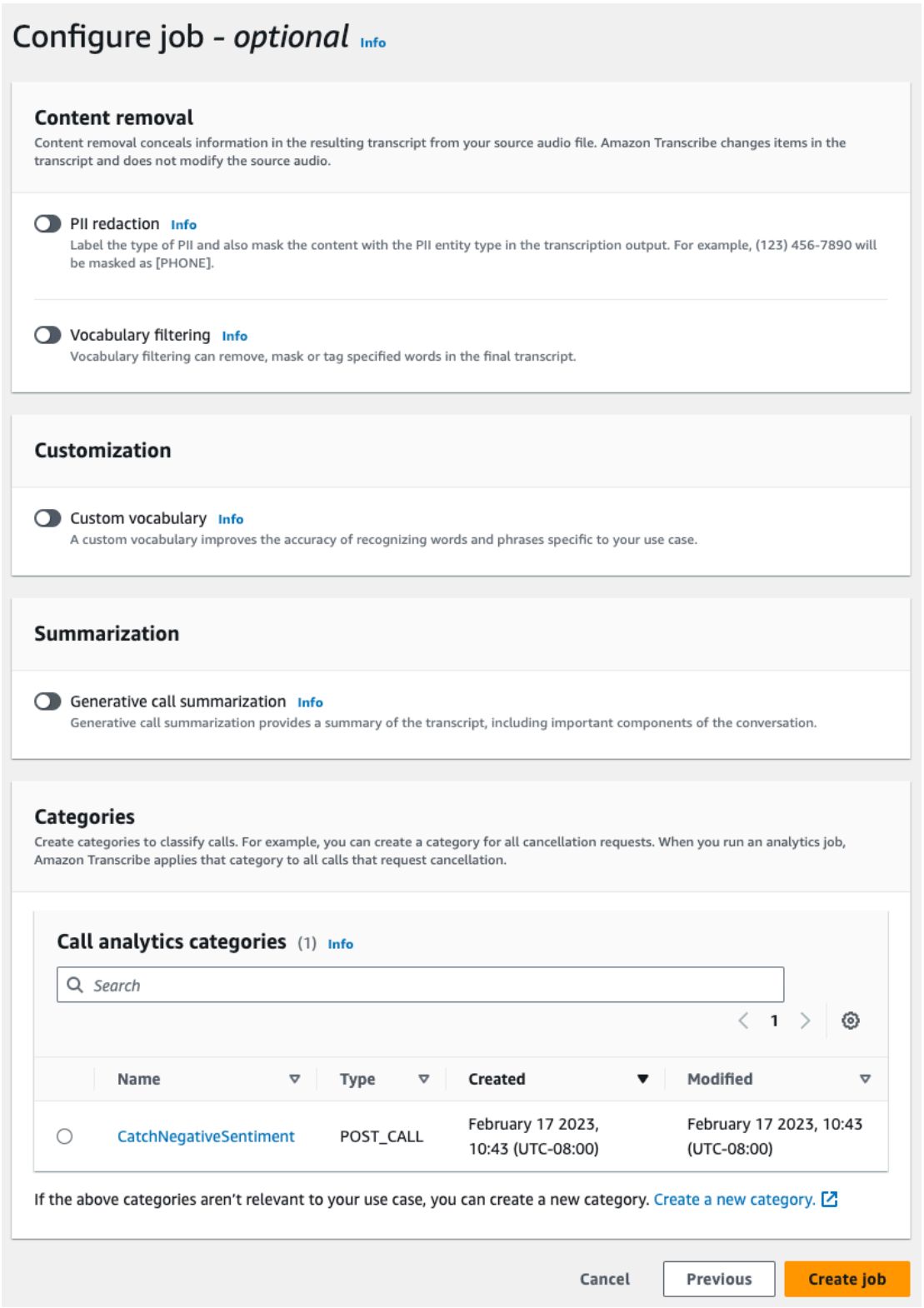
通話の生成要約の有効化

> **Note**
>
> Amazon Bedrock を利用: AWS 自動不正検出を実装します。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 マネジメントコンソール

[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.
Cancel
Previous
Create job

AWS CLI

この例では、start-call-analytics-job コマンドと Settings パラメータを Summarization のサブパラメータで使用しています。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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-location s3://amzn-s3-demo-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}}'
```

次に、start-call-analytics-job コマンドと、ジョブの要約を有効に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 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、次のリクエストボディが含まれています。

```
{
  "CallAnalyticsJobName": "my-first-call-analytics-job",
  "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole",
  "Media": {
    "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputLocation": "s3://amzn-s3-demo-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 AWS SDK for Python (Boto3) 、 を使用して、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 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。

```
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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
output_location = "s3://amzn-s3-demo-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]
- [言語識別]は、各オーディオチャネルで話されているプライマリ言語を検出します。

リアルタイム通話分析に加えて、 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。たとえば、「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 フィルターされます。

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。これは初期状態で機能するため、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には対応していません。

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 (en-GB)、米国 (en-US)、およびスペイン語 (es-US) の英語の方言でサポートされています。

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 のリストを表示したり、 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 (en-GB)、米国 (en-US)、およびスペイン語 (es-US) の英語の方言でサポートされています。

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。

感情分析は初期状態で機能するため、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には対応していません。

以下は出力例です。

言語識別

言語識別は、リアルタイム通話分析中にオーディオストリームの各チャネル内で話される主要言語を自動的に認識して判定します。識別されると、通話分析機能が検出された言語に基づいて最適な文字起こしを処理して返し、この情報をストリームでリアルタイムに配信します。

この機能を使用すると、オーディオストリームの各チャネルで話されている主要言語を自動的に認識して識別できます。言語が検出されると、通話分析機能は識別された言語に適した文字起こしをリアルタイムで処理して配信します。

自動言語識別は、ストリーミング文字起こしで現在サポートされているすべての通話分析ストリーミング対応言語で追加費用なしに利用可能であり、通話分析ストリーミング対応の AWS リージョンで利用できます。

> **Important**
>
> 通話分析は単一言語の識別のみをサポートしており、オーディオチャネルで話されている主要言語を識別します。多言語識別はサポートされておらず、各チャネルは 1 つの言語でのみ文字起こしされます。
> 言語識別を使用するには、サポートされている通話分析ストリーミング言語から、少なくとも 2 つ、最大 5 つの言語コードを指定する必要があります。また、1 つのストリームにつき言語ごとに 1 つの方言のみを選択できます。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。この機能を使用する場合、LanguageCode と IdentifyLanguage は相互に排他的なオプションであるため、LanguageCode パラメータはリクエストで null のままにする必要があります。

Warning

指定した言語コードが実際の言語と一致しない場合、システムはオプションの中から最も類似した言語を選択します。これにより文字起こしの精度が低下する可能性があります。

言語識別機能を使用すると、次のことが可能です。

- 主要言語をリアルタイムで自動的に検出する
- 異なる言語を別々のチャネルで処理する
- 言語検出の信頼スコアを取得する
- 言語固有のカスタム語彙を適用する

言語識別を使用するには、次のパラメータを設定する必要があります。

必須パラメータ:

- `identifyLanguage` - 言語識別を有効にするには、true に設定します。
- `languageOptions` - identifyLanguage が true に設定されているときに使用できる言語コードのリスト。単一言語の選択はサポートされていないため、最低 2 つの言語選択を指定する必要があります。

オプションのパラメータ:

- `preferredLanguage` - 提供された languageOptions から予想される主要言語。優先言語を追加すると、通話分析で言語をより迅速に識別できます。
- `vocabularyNames` - 精度を向上させるためのカスタムボキャブラリー名。ボキャブラリー名では大文字と小文字が区別され、カスタムボキャブラリーの言語が識別されたメディア言語と一致しない場合、文字起こしには適用されないことに注意してください。
- `vocabularyFilterNames` - 文字起こしの出力をカスタマイズするためのボキャブラリーフィルター名。

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。

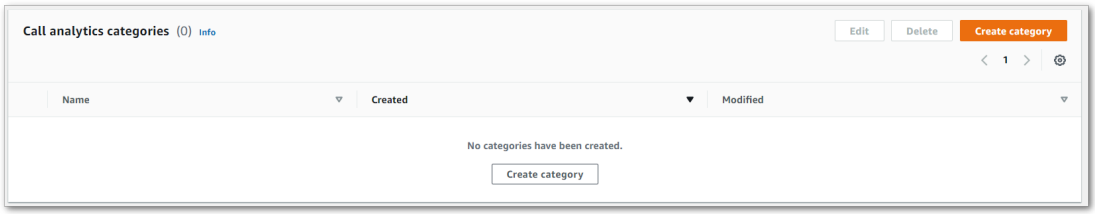
通話後カテゴリとリアルタイムカテゴリ

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。

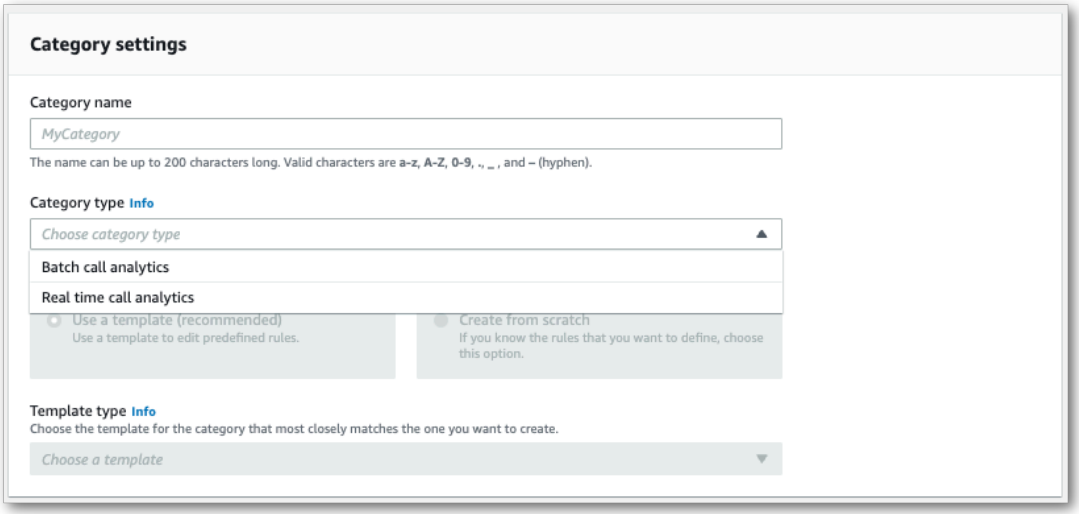
リアルタイム通話分析の新しいカテゴリを作成するには、AWS マネジメントコンソール、AWS CLI、または AWS SDK を使用できます。例については以下を参照してください。

AWS マネジメントコンソール

1. ナビゲーションペインで、Amazon Transcribe 分析の呼び出し Amazon Transcribe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。

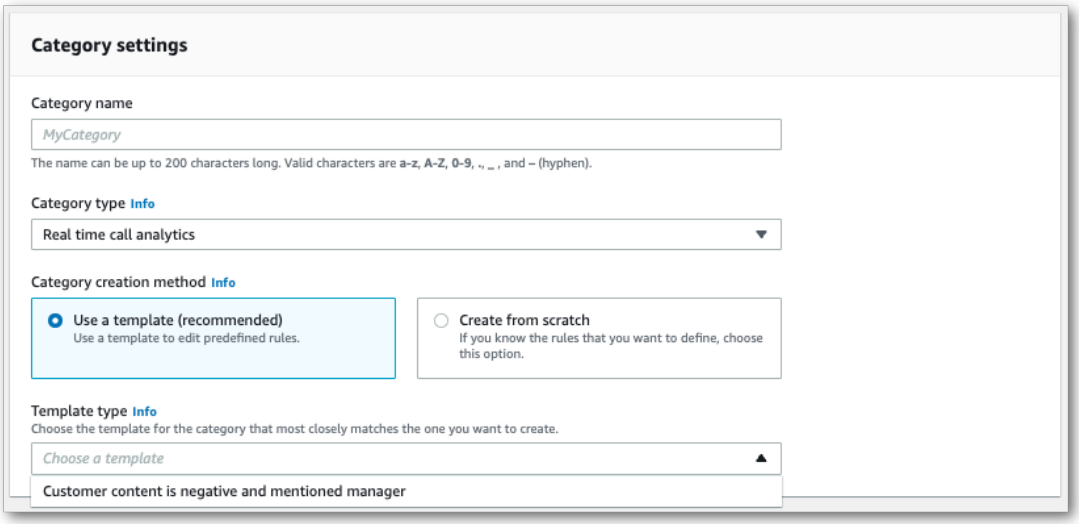


3. [カテゴリの作成ページ] が表示されます。カテゴリの名前を入力し、カテゴリタイプのドロップダウンメニューで [リアルタイムコール分析] を選択します。

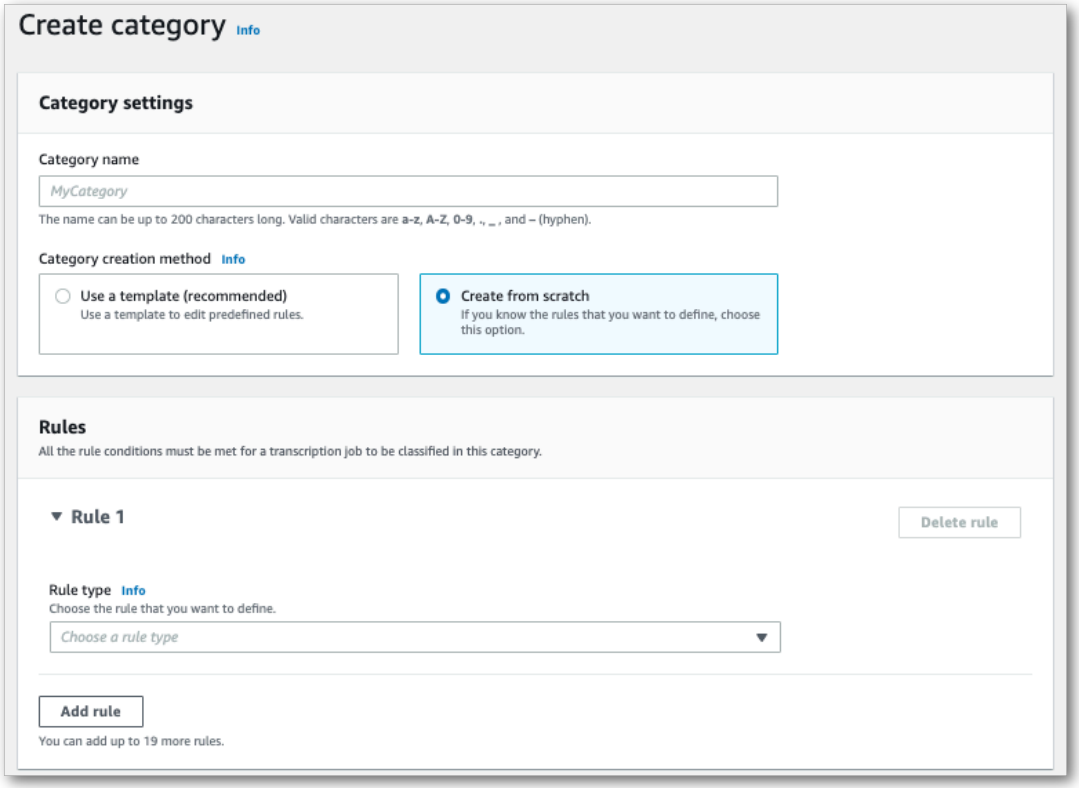


4. テンプレートを選択してカテゴリを作成することも、一から作成することもできます。

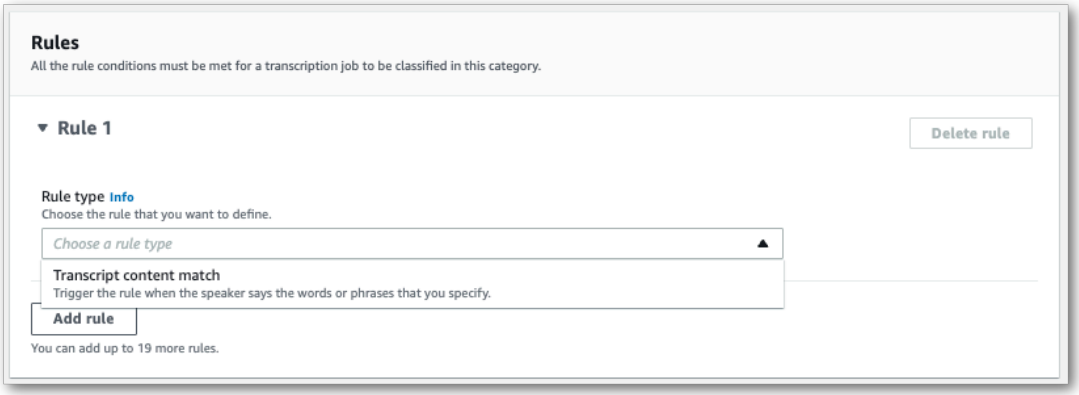
   テンプレートを使用する場合: [テンプレートを使用する (推奨)] を選択し、必要なテンプレートを選択してから [カテゴリの作成] を選択します。



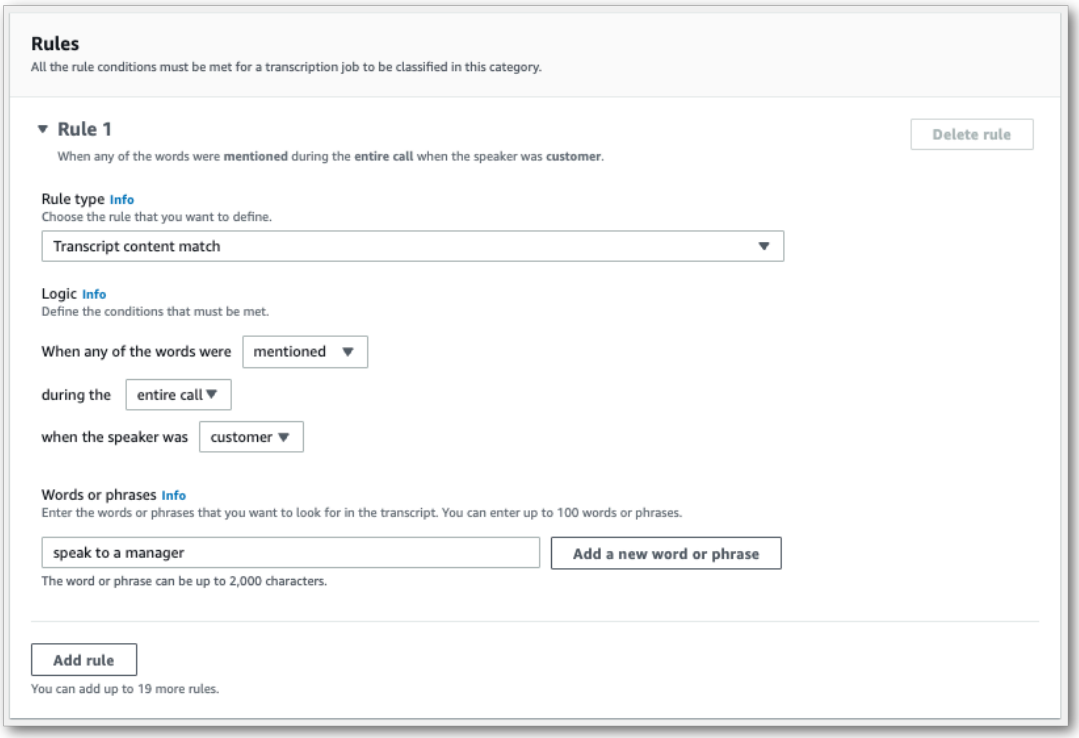
5. カスタムカテゴリを作成する場合: [最初から作成] を選択します。



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ボディを使用しました。

```
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、次のリクエストボディが含まれています。

```
{
  "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 AWS SDK for Python (Boto3) 、 を使用して、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。

- 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 分を超える長時間オーディオストリームに対しては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た状態でリアルタイムコール分析文字起こしを開始するには、AWS マネジメントコンソール (デモのみ)、HTTP/2、またはWebSocketを使用できます。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 マネジメントコンソール のみが、オーディオの例がプリロードされたリアルタイムコール分析のデモを提供しています。独自の音声を使用する場合は、API (HTTP/2、WebSocket、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

通話後のトランスクリプトは、セグメントごとにターンバイターン形式で表示されます。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。

リアルタイム通話分析トランスクリプションを開始するには、AWS マネジメントコンソール、HTTP/2、または WebSocket を使用できます。例については以下を参照してください。

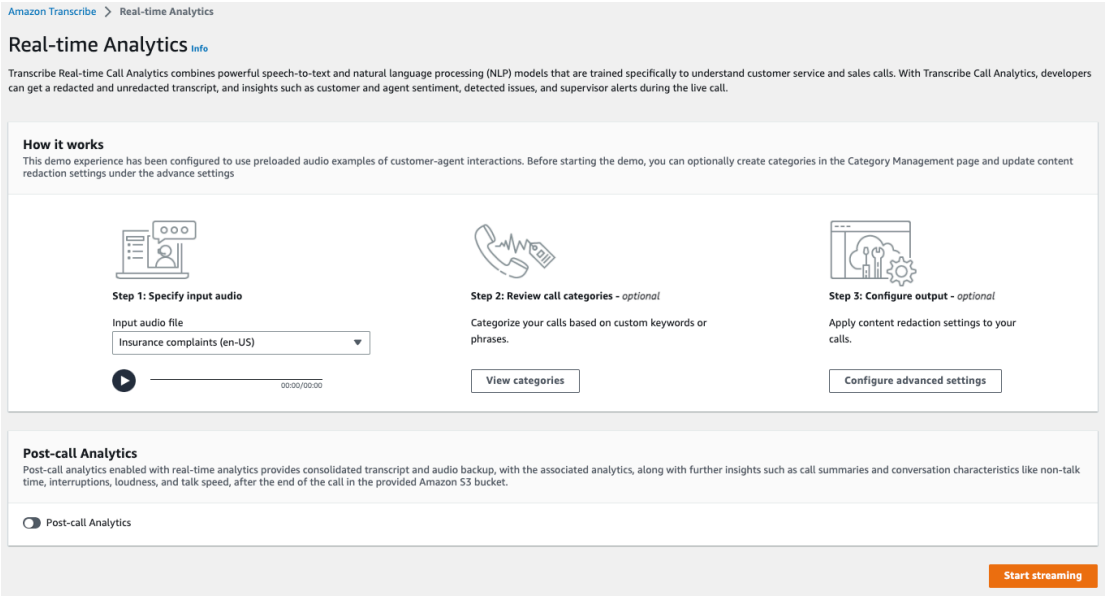
> **Important**
>
> 現在、 AWS マネジメントコンソール のみが、オーディオの例がプリロードされたリアルタイムコール分析のデモを提供しています。独自の音声を使用する場合は、API (HTTP/2、WebSocket、または SDK) を使用する必要があります。

AWS マネジメントコンソール

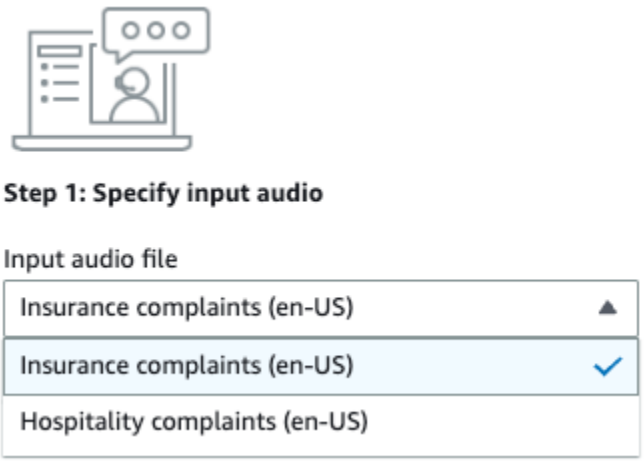
コール分析リクエストをスタートするには、次の手順を実行します。カテゴリで定義されたすべての特性に一致する通話は、該当するカテゴリでラベル付けされます。

> **Note**
>
> AWS マネジメントコンソール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「分析の Amazon Transcribe 呼び出し」で、「リアルタイム呼び出しの分析」を選択します。

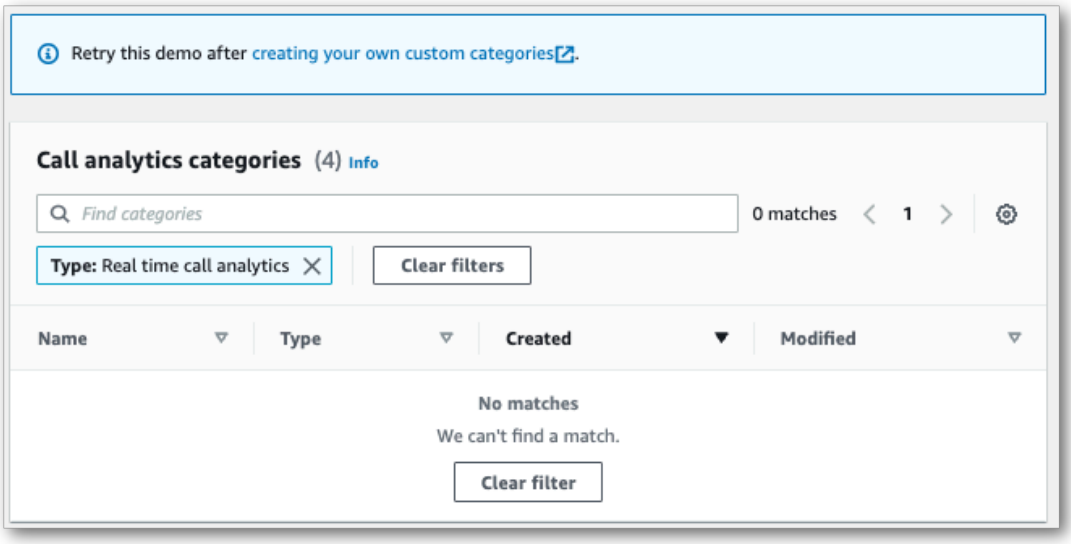


2. ステップ 1: 入力音声の指定では、ドロップダウンメニューから [デモテストファイル] を選択します。



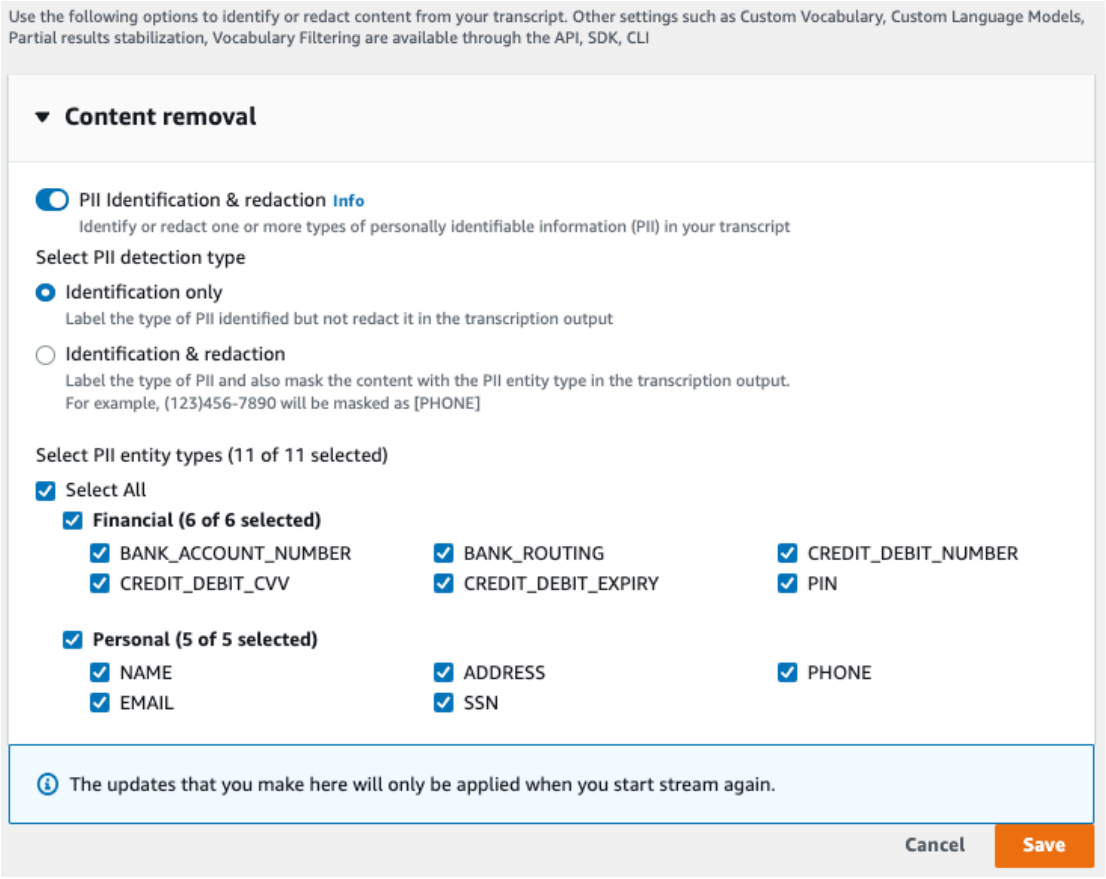
3. ステップ 2: 通話カテゴリの確認では、以前に作成したリアルタイムコール分析カテゴリを確認するオプションがあります。リアルタイムコール分析カテゴリはすべて、文字起こしに適用されます。

[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。



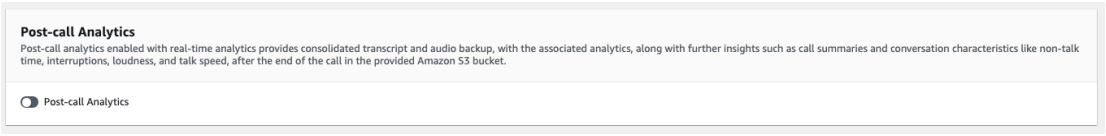
4. ステップ 3: 入力と出力を設定するでは、追加の設定を適用するオプションがあります。

   [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。

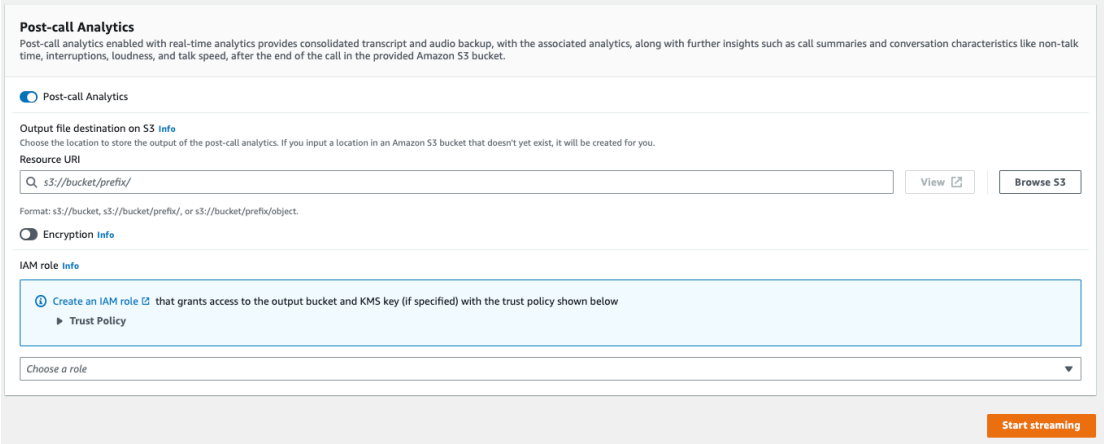


すべての選択を終えたら、[保存] を選択してメインページに戻ります。

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。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://amzn-s3-demo-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 AWS API オペレーションに共通のパラメータは、「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。 Amazon Transcribeでの WebSocket ストリームの使用の詳細については、「WebSocket ストリームの設定」を参照してください。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://amzn-s3-demo-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 AWS API オペレーションに共通のパラメータは、「共通パラメータ」セクションに記載されています。

> **Tip**
>
> 上記の HTTP/2 と WebSocket 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。

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、セグメントごとにターンバイターン形式で表示されます。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
        }
    ]
},
```

言語識別

トランスクリプション出力では、言語識別は次のように表示されます。

```
{
  "LanguageCode": "en-US",
```

```
  "LanguageIdentification": [
    {
      "Code": "en-US",
      "Score": 0.823
    },
    {
      "Code": "de-DE",
      "Score": 0.177
    }
  ]
}
```

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
                }
            }],
            "LanguageCode": "en-US",
            "LanguageIdentification": [
                {
                    "Code": "en-US",
                    "Score": 0.823
                },
                {
                    "Code": "de-DE",
                    "Score": 0.177
                }
            ]
        },
        "CategoryEvent": {
            "MatchedCategories": [
                "network-complaint"
            ],
            "MatchedDetails": {
                "network issues" : {
                    "TimestampRanges": [
                        {
                            "BeginOffsetMillis": 9299375,
                            "EndOffsetMillis": 7899375
                        }
                    ]
                }
            }
        }
    }
}
```

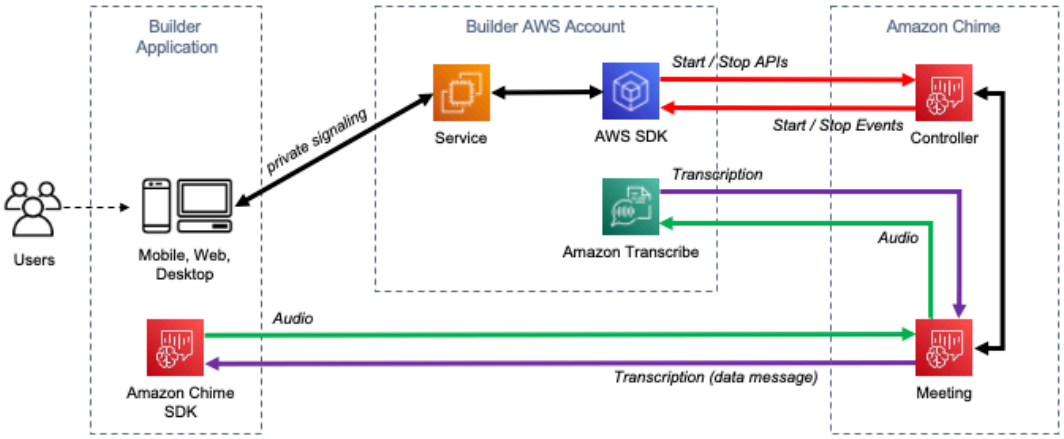
Amazon Chime 通話をリアルタイムで文字起こしする

Amazon Transcribe は Amazon Chime SDK と統合されているため、 Amazon Chime 通話のリアルタイム文字起こしが容易になります。

SDK API を使用して文字起こ Amazon Transcribe しをリクエストすると、 は に音声の Amazon Chime ストリーミング Amazon Chime を開始し、呼び出しの期間中も音声のストリーミングを継続します。

Amazon Chime SDK は「アクティブトーカー」アルゴリズムを使用して上位 2 つのアクティブトーカーを選択し、そのオーディオを 1 つのストリームを介して 2 つの別々のチャネル Amazon Transcribe として に送信します。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の文字起こしを受け取ります。配信例は Amazon Chime SDK デベロッパーガイドで確認できます。

文字 Amazon Chime 起こしのデータフローを次の図に示します。



リアルタイム Amazon Chime 文字起こしの設定方法の詳細については、Amazon Chime 「 SDK デベロッパーガイド」の「 SDK ライブ文字起こしの使用」を参照してください。 Amazon Chime API オペレーションについては、「Amazon Chime SDK API 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

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 Chime SDK 会議が Amazon Transcribe および でのライブ文字起こしをサポートするようになりました Amazon Transcribe Medical

- 遠隔医療ソリューションのAmazon Chime 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、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

AWS SDK 開発者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 でのこのサービスの使用。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

- AWS SDK を使用して Amazon Transcribe カスタム語彙を作成および絞り込む
- 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

SDK を使用した Amazon Transcribe の基本的な例 AWS SDKs

次のコード例は、 AWS SDK で Amazon Transcribe を使用する基本的な方法を説明しています。

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 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。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

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。アクションは SDK を使用した Amazon Transcribe のシナリオ AWS SDKs のコンテキスト内で確認できます。

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 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

次のサンプルコードは、`CreateVocabulary` を使用する方法を説明しています。

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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
/// <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、AWS SDK for .NET API リファレンスの「CreateVocabulary」を参照してください。

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。vocabulary-file-uri として、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://amzn-s3-demo-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、「AWS CLI コマンドリファレンス」の「CreateVocabulary」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、AWS SDK for Python (Boto3) API リファレンスの「CreateVocabulary」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    TRY.
```

```
        IF it_phrases IS NOT INITIAL.
          oo_result = lo_tnb->createvocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            it_phrases = it_phrases ).
        ELSEIF iv_vocab_file_uri IS NOT INITIAL.
          oo_result = lo_tnb->createvocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            iv_vocabularyfileuri = iv_vocab_file_uri ).
        ENDIF.
        MESSAGE 'Custom vocabulary created.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスの「CreateVocabulary」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、DeleteMedicalTranscriptionJob を使用する方法を説明しています。

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    /// <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、「AWS SDK for .NET API リファレンス」の「DeleteMedicalTranscriptionJob」を参照してください。

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」の「DeleteMedicalTranscriptionJob」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「DeleteMedicalTranscriptionJob」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。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、「AWS SDK for JavaScript API リファレンス」の「DeleteMedicalTranscriptionJob」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、DeleteTranscriptionJob を使用する方法を説明しています。

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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    /// <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、「AWS SDK for .NET API リファレンス」の「DeleteTranscriptionJob」を参照してください。

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」の「DeleteTranscriptionJob」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteTranscriptionJob」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

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、AWS SDK for JavaScript API リファレンスの「DeleteTranscriptionJob」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、「AWS SDK for Python (Boto3) API リファレンス」の「DeleteTranscriptionJob」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
TRY.
    lo_tnb->deletetranscriptionjob( iv_job_name ).
    MESSAGE 'Transcription job deleted.' TYPE 'I'.
  CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
    MESSAGE lo_bad_request_ex TYPE 'I'.
    RAISE EXCEPTION lo_bad_request_ex.
  CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
    MESSAGE lo_limit_ex TYPE 'I'.
    RAISE EXCEPTION lo_limit_ex.
  CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
    MESSAGE lo_internal_ex TYPE 'I'.
    RAISE EXCEPTION lo_internal_ex.
ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのDeleteTranscriptionJob」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、DeleteVocabulary を使用する方法を説明しています。

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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
/// <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、AWS SDK for .NET API リファレンスの「DeleteVocabulary」を参照してください。

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、AWS CLI コマンドリファレンスの「DeleteVocabulary」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、AWS SDK for Python (Boto3) API リファレンスの「DeleteVocabulary」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    TRY.
        lo_tnb->deletevocabulary( iv_vocabulary_name ).
        MESSAGE 'Vocabulary deleted.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのDeleteVocabulary」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、GetTranscriptionJob を使用する方法を説明しています。

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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    /// <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、AWS SDK for .NET API リファレンスの「GetTranscriptionJob」を参照してください。

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、このジョブで書き起こした音声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://amzn-s3-demo-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、「AWS CLI コマンドリファレンス」の「GetTranscriptionJob」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、「AWS SDK for Python (Boto3) API リファレンス」の「GetTranscriptionJob」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
TRY.
    oo_result = lo_tnb->gettranscriptionjob( iv_job_name ).
    DATA(lo_job) = oo_result->get_transcriptionjob( ).
    MESSAGE 'Retrieved transcription job details.' TYPE 'I'.
  CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
    MESSAGE lo_bad_request_ex TYPE 'I'.
    RAISE EXCEPTION lo_bad_request_ex.
  CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
    MESSAGE lo_not_found_ex TYPE 'I'.
    RAISE EXCEPTION lo_not_found_ex.
  CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
    MESSAGE lo_internal_ex TYPE 'I'.
    RAISE EXCEPTION lo_internal_ex.
ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのGetTranscriptionJob」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、GetVocabulary を使用する方法を説明しています。

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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
/// <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、AWS SDK for .NET API リファレンスの「GetVocabulary」を参照してください。

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、AWS CLI コマンドリファレンスの「GetVocabulary」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、「AWS SDK for Python (Boto3) API リファレンス」の「GetVocabulary」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    TRY.
        oo_result = lo_tnb->getvocabulary( iv_vocabulary_name ).
        MESSAGE 'Retrieved vocabulary details.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
```

```
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのGetVocabulary」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、ListMedicalTranscriptionJobs を使用する方法を説明しています。

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    /// <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、「AWS SDK for .NET API リファレンス」の「ListMedicalTranscriptionJobs」を参照してください。

CLI

AWS CLI

医療文字起こしジョブを一覧表示するには

次の 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、文字起こし出力の 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。さらに他の文字起こしジョブを表示するには、NextToken パラメータの値をコピーし、再度 list-medical-transcription-jobs コマンドを実行して、その値を --next-token オプションに指定します。

```
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、「AWS CLI コマンドリファレンス」の「ListMedicalTranscriptionJobs」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。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、「AWS SDK for JavaScript API リファレンス」の「ListMedicalTranscriptionJobs」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、ListTranscriptionJobs を使用する方法を説明しています。

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
   /// <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、「AWS SDK for .NET API リファレンス」の「ListTranscriptionJobs」を参照してください。

CLI

AWS CLI

文字起こしジョブを一覧表示するには

次の 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、AWS CLI コマンドリファレンスの「ListTranscriptionJobs」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、AWS SDK for Java 2.x API リファレンスの「ListTranscriptionJobs」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

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、AWS SDK for JavaScript API リファレンスの「ListTranscriptionJobs」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、「AWS SDK for Python (Boto3) API リファレンス」の「ListTranscriptionJobs」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    TRY.
        IF iv_job_filter IS NOT INITIAL.
          oo_result = lo_tnb->listtranscriptionjobs( iv_jobnamecontains =
iv_job_filter ).
        ELSE.
          oo_result = lo_tnb->listtranscriptionjobs( ).
        ENDIF.
        MESSAGE 'Retrieved transcription jobs list.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのListTranscriptionJobs」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、ListVocabularies を使用する方法を説明しています。

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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    /// <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、AWS SDK for .NET API リファレンスの「ListVocabularies」を参照してください。

CLI

AWS CLI

カスタム語彙を一覧表示するには

次の 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た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、AWS CLI コマンドリファレンスの「ListVocabularies」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、「AWS SDK for Python (Boto3) API リファレンス」の「ListVocabularies」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    TRY.
        IF iv_vocab_filter IS NOT INITIAL.
          oo_result = lo_tnb->listvocabularies( iv_namecontains =
iv_vocab_filter ).
        ELSE.
          oo_result = lo_tnb->listvocabularies( ).
        ENDIF.
        MESSAGE 'Retrieved vocabularies list.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
```

```
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのListVocabularies」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、StartMedicalTranscriptionJob を使用する方法を説明しています。

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    /// <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、「AWS SDK for .NET API リファレンス」の「StartMedicalTranscriptionJob」を参照してください。

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":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    }
}
```

出力:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-
job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-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。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://mysecondfile.json
```

mysecondfile.json の内容:

```
{
    "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    }
}
```

出力:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-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":"amzn-s3-demo-bucket",
        "Media": {
          "MediaFileUri": "s3://amzn-s3-demo-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://amzn-s3-demo-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":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-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://amzn-s3-demo-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":"amzn-s3-demo-bucket",
    "Media": {
```

```
        "MediaFileUri": "s3://amzn-s3-demo-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://amzn-s3-demo-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。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://mysixthfile.json
```

mysixthfile.json の内容:

```
{
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-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://amzn-s3-demo-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":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-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://amzn-s3-demo-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、AWS CLI コマンドリファレンスの「StartMedicalTranscriptionJob」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。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、「AWS SDK for JavaScript API リファレンス」の「StartMedicalTranscriptionJob」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、StartTranscriptionJob を使用する方法を説明しています。

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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    /// <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、「AWS SDK for .NET API リファレンス」の「StartTranscriptionJob」を参照してください。

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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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://amzn-s3-demo-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、「AWS CLI コマンドリファレンス」の「StartTranscriptionJob」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

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、「AWS SDK for JavaScript API リファレンス」の「StartTranscriptionJob」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、「AWS SDK for Python (Boto3) API リファレンス」の「StartTranscriptionJob」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    TRY.
        DATA(lo_media) = NEW /aws1/cl_tnbmedia( iv_mediafileuri = iv_media_uri ).
        DATA(lo_settings) = NEW /aws1/cl_tnbsettings( ).
        IF iv_vocabulary_name IS NOT INITIAL.
          lo_settings = NEW /aws1/cl_tnbsettings( iv_vocabularyname =
iv_vocabulary_name ).
        ENDIF.

        oo_result = lo_tnb->starttranscriptionjob(
          iv_transcriptionjobname = iv_job_name
          io_media = lo_media
          iv_mediaformat = iv_media_format
```

```
          iv_languagecode = iv_language_code
          io_settings = lo_settings ).

        MESSAGE 'Transcription job started.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのStartTranscriptionJob」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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

次のサンプルコードは、UpdateVocabulary を使用する方法を説明しています。

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

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
    /// <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、「AWS SDK for .NET API リファレンス」の「UpdateVocabulary」を参照してください。

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://amzn-s3-demo-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、AWS CLI コマンドリファレンスの「UpdateVocabulary」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
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、「AWS SDK for Python (Boto3) API リファレンス」の「UpdateVocabulary」を参照してください。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> GitHub には、その他のリソースもあります。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。

```
TRY.
    IF it_phrases IS NOT INITIAL.
      oo_result = lo_tnb->updatevocabulary(
        iv_vocabularyname = iv_vocabulary_name
        iv_languagecode = iv_language_code
        it_phrases = it_phrases ).
    ELSEIF iv_vocab_file_uri IS NOT INITIAL.
      oo_result = lo_tnb->updatevocabulary(
        iv_vocabularyname = iv_vocabulary_name
        iv_languagecode = iv_language_code
        iv_vocabularyfileuri = iv_vocab_file_uri ).
    ENDIF.
    MESSAGE 'Vocabulary updated.' TYPE 'I'.
  CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
    MESSAGE lo_bad_request_ex TYPE 'I'.
  CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
    MESSAGE lo_limit_ex TYPE 'I'.
    RAISE EXCEPTION lo_limit_ex.
  CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
    MESSAGE lo_not_found_ex TYPE 'I'.
  CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
    MESSAGE lo_internal_ex TYPE 'I'.
    RAISE EXCEPTION lo_internal_ex.
  CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
    MESSAGE lo_conflict_ex TYPE 'I'.
    RAISE EXCEPTION lo_conflict_ex.
ENDTRY.
```

- API の詳細については、 AWS SDK for SAP ABAP API リファレンスのUpdateVocabulary」を参照してください。

AWS SDK 開発者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 でのこのサービスの使用。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。これらのシナリオでは、Amazon Transcribe 内で複数の関数を呼び出すか、その他の AWS のサービスと組み合わせて、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、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

例

- 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
- AWS SDK を使用して Amazon Transcribe カスタム語彙を作成および絞り込む
- 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

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および実行の手順については、GitHub で完全な例を参照してください。

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 開発者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 でのこのサービスの使用。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

次のコード例は、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および実行の手順については、GitHub で完全な例を参照してください。

この例で使用されているサービス

- Amazon Polly
- Amazon S3
- Amazon Transcribe

AWS SDK 開発者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 でのこのサービスの使用。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 カスタム語彙を作成および絞り込む

次のコード例は、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**
>
> GitHub には、その他のリソースもあります。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。

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 開発者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 でのこのサービスの使用。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。
- トランスクリプトが保存されている 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**
>
> GitHub には、その他のリソースもあります。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。

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);
                })
                .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 static Subscription currentSubscription;
        private final InputStream inputStream;

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

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

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

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final 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**
>
> GitHub には、その他のリソースもあります。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。

```
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 開発者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 でのこのサービスの使用。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。

のセキュリティ Amazon Transcribe

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、最もセキュリティの影響を受けやすい組織の要件を満たすように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活用できます。

セキュリティは、 AWS とお客様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説明しています。

- クラウドのセキュリティ: AWS は、 で AWS サービスを実行するインフラストラクチャを保護する責任を担います AWS クラウド。 は、安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。が適用されるコンプライアンスプログラムの詳細については Amazon Transcribe、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ: お客様の責任は、使用する AWS のサービスに応じて判断されます。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。以下のトピックでは、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 Amazon Transcribe ように を設定する方法について説明します。また、他の AWS サービスを使用して Amazon Transcribe リソースをモニタリングおよび保護する方法についても説明します。

トピック

- の Identity and Access Management Amazon Transcribe
- でのデータ保護 Amazon Transcribe
- モニタリング Amazon Transcribe
- のコンプライアンス検証 Amazon Transcribe
- の耐障害性 Amazon Transcribe
- のインフラストラクチャセキュリティ Amazon Transcribe
- での脆弱性の分析と管理 Amazon Transcribe
- のセキュリティのベストプラクティス Amazon Transcribe

の Identity and Access Management Amazon Transcribe

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 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に Amazon Transcribe リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「Amazon Transcribe ID とアクセスのトラブルシューティング」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「が IAM と Amazon Transcribe 連携する方法」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「Amazon Transcribe アイデンティティベースのポリシーの例」を参照)

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

(AWS IAM アイデンティティセンター IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサイ

ンインできます。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。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、まず、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント root ユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。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。ルートユーザー認証情報を必要とするタスクについては、「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。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーティッド ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID ソースの認証情報 AWS のサービス を使用して Directory Service 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は、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「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、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの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。詳細については、IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用して にアクセスすることを人間 AWS のユーザーに要求する」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、ID またはリソースに関連付けられたときにアクセス許可を定義します。 は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、「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、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

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のポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーと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「管理ポリシーとインラインポリシーのいずれかを選択する」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。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。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。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。

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を使用できません。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の最大数を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。
- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。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」を参照してください。

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アクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

が IAM と Amazon Transcribe 連携する方法

IAM を使用して へのアクセスを管理する前に Amazon Transcribe、 で使用できる IAM 機能を確認してください Amazon Transcribe。

IAM で使用できる機能 Amazon Transcribe

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	なし
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
一時認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	あり
サービスリンクロール	いいえ

Amazon Transcribe および他の AWS のサービスがほとんどの IAM 機能とどのように連携するかの概要を把握するには、IAM 「 ユーザーガイド」のAWS 「 と連携する のサービス IAM」を参照してください。

のアイデンティティベースのポリシー Amazon Transcribe

アイデンティティベースのポリシーのサポート: 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 許可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、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」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。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。

のアイデンティティベースのポリシーの例 Amazon Transcribe

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、「」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。

内のリソースベースのポリシー Amazon Transcribe

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には例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。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コントロールできます。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、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、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

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。詳細については、IAM ユーザーガイドのIAM でのクロスアカウントリソースアクセスを参照してください。

のポリシーアクション Amazon Transcribe

ポリシーアクションのサポート: あり

管理者は JSON AWS 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。

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。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アクセス許可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「 で定義されるアクション Amazon Transcribe」を参照してください。

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 Amazon Transcribe 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、「」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。

のポリシーリソース Amazon Transcribe

ポリシーリソースのサポート: あり

管理者は JSON AWS 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。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。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。リソースレベルのアクセス許可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 リソースタイプとその ARNs「 で定義されるリソース Amazon Transcribe」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、「」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。

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: あり

管理者は JSON AWS 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。

`Condition` 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、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。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 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「 の条件キー Amazon Transcribe」を参照してください。条件キーを使用できるアクションとリソースについては、「 で定義されるアクション Amazon Transcribe」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、「」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。

ACLs Amazon Transcribe

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アクセス許可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。

を使用した ABAC Amazon Transcribe

ABAC (ポリシー内のタグ) のサポート: 一部

属性ベースのアクセスコントロール (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグ

がリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

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、ポリシーの条件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 つの条件キーすべてをサポートする場合、そのサービスの値はありです。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 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「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、「」を参照してくださいリソースのタグ付け。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。

での一時的な認証情報の使用 Amazon Transcribe

一時的な認証情報のサポート: あり

一時的な認証情報は、 AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。 AWS では、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 ユーザーガイド」の「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

のクロスサービスプリンシパルのアクセス許可 Amazon Transcribe

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。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。

のサービスロール Amazon Transcribe

サービスロールのサポート: 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 のサービスに許可を委任するロールを作成する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 機能が破損する可能性があります。 Amazon Transcribe が指示する場合にのみ、サービスロールを編集します。

のサービスにリンクされたロール Amazon Transcribe

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、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アクセス許可を表示できますが、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 サービスの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、AWS 「 と連携する のサービス IAM」を参照してください。表の「サービスリンクロール」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 (サービスやアカウント) のことです。この種のなりすましは、クロスアカウントおよびクロスサービスで発生するおそれがあります。

混乱した代理を防ぐために、 は、 のリソースへのアクセス権が付与されたサービスプリンシパルを使用して、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 AWS を提供します AWS アカウント。このセクションでは、 固有のサービス間の混乱した代理の防止に焦点を当てています Amazon Transcribeが、このトピックの詳細については、IAM 「 ユーザーガイド」の混乱した代理の問題セクションを参照してください。

が リソースにアクセス IAM Amazon Transcribe するためのアクセス許可を制限するには、リソースポリシーaws:SourceAccountで グローバル条件コンテキストキー aws:SourceArnと 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の両方を使用し、 aws:SourceArn値に AWS アカウント ID が含まれている場合、同じポリシーステートメントで使用する場合、 aws:SourceAccountの値と AWS アカウント で同じ AWS アカウント ID aws: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。その 内のリソースを AWS アカウント クロスサービスアクセスに関連付ける場合は、 を使用します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 を指定しながら、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*:*。

混乱した代理問題を防ぐ方法を示すロールの継承ポリシーの例については、「混乱した代理の防止ポリシー」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

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。

これらのサンプルの 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 ポリシーを作成する (コンソール)」を参照してください。

Amazon Transcribe が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認可リファレンス」の「Amazon Transcribe のアクション、リソース、条件キー」を参照してください。

トピック

- ポリシーに関するベストプラクティス

- の使用 AWS マネジメントコンソール
- IAM ロールに必要なアクセス許可
- Amazon S3 暗号化キーに必要なアクセス許可
- 自分の権限の表示をユーザーに許可する
- AWS KMS 暗号化コンテキストポリシー
- 混乱した代理の防止ポリシー
- タグに基づく文字起こしジョブの表示

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内の Amazon Transcribe リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。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 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス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にアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス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 ジョブ機能のAWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、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。たとえば、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できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、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 CloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。
- IAM アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー 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 アクセスアナライ

ザーは 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の作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。

- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。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 を使用した安全な API アクセス 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。

の使用 AWS マネジメントコンソール

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、 の Amazon Transcribe リソースの詳細を一覧表示および表示できます AWS アカウント。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アクセス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 を使用できるようにするにはAWS マネジメントコンソール、次のいずれか AWSの管理ポリシーをアタッチします。

- `AmazonTranscribeFullAccess`: すべての Amazon Transcribe リソースを作成、読み取り、更新、削除、実行するフルアクセスを許可します。また、バケット名に `transcribe` を含む Amazon S3 バケットへのアクセスも許可します。
- `AmazonTranscribeReadOnlyAccess`: 文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**
>
> 管理ポリシーは、 IAM AWS マネジメントコンソール にサインインしてポリシー名で検索することで確認できます。「transcribe(文字起こし)」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 アクションのアクセス許可を許可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

呼び出す IAM ロールを作成する場合は Amazon Transcribe、 Amazon S3 バケットへのアクセス許可が必要です。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、以下のセクションを参照してください。

信頼ポリシー

文字起こしリクエストを行うために使用する IAM エンティティには、 がそのロール Amazon Transcribe を引き受けることができる信頼ポリシーが必要です。次の Amazon Transcribe 信頼ポリシーを使用します。通話後分析を有効にした状態でリアルタイムコール分析リクエストを行う場合は、「リアルタイムコール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

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "transcribe.amazonaws.com"
        ]
      },
      "Action": [
        "sts:AssumeRole"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:transcribe:us-west-2:111122223333:*"
        }
```

```
      }
    }
  ]
}
```

リアルタイムコール分析の信頼ポリシー

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "transcribe.streaming.amazonaws.com"
        ]
      },
      "Action": [
        "sts:AssumeRole"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
          "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、指定された入力バケットからファイルにアクセスするアクセス許可を IAM ロールに付与します。

JSON

```
{
    "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

次のポリシーは、指定された出力バケットにファイルを書き込むアクセス許可を IAM ロールに付与します。

JSON

```
{
    "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アクセス許可

を使用して Amazon S3 バケットを KMS key 暗号化する場合は、 KMS key ポリシーに以下を含めます。これにより、 はバケットの内容 Amazon Transcribe にアクセスできます。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 デベロッパーガイド」の「 AWS アカウントへのアクセスを外部に許可 KMS keyする」を参照してください。

JSON

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

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチ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の作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行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

次のポリシーは、この特定の AWS KMS の Decrypt および Encrypt オペレーションを使用するアクセス`ExampleRole`許可を IAM ロールに付与します KMS key。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 暗号化コンテキスト。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Principal": {
              "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
```

```
            },
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey*",
                "kms:ReEncrypt*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:color": "indigoBlue"
                }
             }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

混乱した代理の防止ポリシー

以下は、混乱した代理問題aws:SourceAccount Amazon Transcribe を防ぐために aws:SourceArnと を使用する方法を示すロールの継承ポリシーの例です。混乱した代理の防止に関する詳細については、「サービス間の混乱した代理の防止」を参照してください。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "transcribe.amazonaws.com"
            },
```

```
            "Action": [
                "sts:AssumeRole"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:transcribe:us-
west-2:111122223333:*"
                }
            }
        }
    ]
}
```

タグに基づく文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、文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、アクセス許可は、文字起こしジョブタグ Owner にそのユーザーのユーザー名の値がある場合のみ、付与されます。このポリシーでは、 AWS マネジメントコンソール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。

このポリシーは、アカウントの IAM エンティティにアタッチできます。test-role という名前のロールが文字起こしジョブを表示しようとする場合、その文字起こしジョブには Owner=test-role または owner=test-role というタグを付ける必要があります(条件キー名では大文字と小文字は区別されません)。そうしないと、アクセスが拒否されます。詳細については、IAM ユーザーガイドの「IAM JSON ポリシー要素: 条件」を参照してください。

での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。

JSON

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

```
            "Action": "transcribe:ListTranscriptionJobs",
            "Resource": "*"
        },
        {
            "Sid": "ViewTranscriptionJobsIfOwner",
            "Effect": "Allow",
            "Action": "transcribe:GetTranscriptionJob",
            "Resource": "arn:aws:transcribe:us-west-2:111122223333:transcription-
job/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

Amazon Transcribe ID とアクセスのトラブルシューティング

次の情報を使用して、 および AWS Identity and Access Management () の使用時に発生する可能性がある一般的な問題を診断 Amazon Transcribe および修正しますIAM。

トピック

- でアクションを実行する権限がありません Amazon Transcribe
- iam:PassRole を実行する権限がありません
- 自分の 以外のユーザーに自分の Amazon Transcribe リソース AWS アカウント へのアクセスを許可したい

でアクションを実行する権限がありません Amazon Transcribe

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、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 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

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またはサービスにリンクされたロールを作成する代わりに、既存のロールをその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権限が必要です。

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と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権限が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary 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 管理者にお問い合わせください。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 以外のユーザーに自分の Amazon Transcribe リソース AWS アカウント 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人が、リソースにアクセスするため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できます。リソースベースのポリシーまたはアクセスコントロールリスト (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、以下を参照してください:

- がこれらの機能 Amazon Transcribe をサポートしているかどうかを確認するには、「」を参照してくださいが IAM と Amazon Transcribe 連携する方法。

- 所有 AWS アカウント している のリソースへの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「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」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におけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

でのデータ保護 Amazon Transcribe

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 はすべての 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責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされ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、「Data Privacy FAQChina」を参照してください。 欧州におけるデータ保護に関する情報については、General Data Protection Regulation (GDPR) Center を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定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が各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 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 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。

- Amazon Macie などの高度な管理された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 AWS を介して にアクセスするときに FIPS 140-3 検証済みの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) 140-3」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を、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含めないことを強くお勧めします。これは、コンソール Amazon Transcribe 、API、または SDK を使用して AWS CLIまたは他の AWS のサービス を操作する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、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

の Amazon Virtual Private Cloud (Amazon VPC) エンドポイント Amazon Transcribe は、接続のみを許可する VPC 内の論理エンティティです Amazon Transcribe。 Amazon VPC はリクエストを にルーティング Amazon Transcribe し、レスポンスを VPC に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。で Amazon VPC Amazon Transcribe エンドポイントを使用する方法については、「」を参照してくださいAmazon Transcribe およびインターフェイス VPC エンドポイント (AWS PrivateLink)。

データ暗号化

データ暗号化とは、転送中と保管中のデータをすることです。で管理 Amazon S3されたキーを使用するか、転送中に標準の Transport Layer Security (TLS) とともに KMS keys 保管することで、データを保護できます。

保管中の暗号化

Amazon Transcribe は、 Amazon S3 バケットに配置されたトランスクリプトのサーバー側の暗号化にデフォルト Amazon S3 キー (SSE-S3) を使用します。

`StartTranscriptionJob` オペレーションを使用すると、独自の を指定 KMS key 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 は、デフォルトキーで暗号化された Amazon EBS ボリュームを使用します。

転送中の暗号化

Amazon Transcribe は、TLS 1.2 と AWS 証明書を使用して、転送中のデータを暗号化します。ストリーミング文字起こしも対象になります。

キー管理

Amazon Transcribe は と連携して KMS keys 、データの暗号化を強化します。を使用すると Amazon S3、文字起こしジョブの作成時に入力メディアを暗号化できます。との統合 AWS KMS 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、文字起こしジョブの出力はデフォルトの Amazon S3 キー (SSE-S3) で暗号化されます。

詳細については AWS KMS、「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 マネジメントコンソール

文字起こしジョブの出力を暗号化するには、リクエスト AWS アカウント を行っている KMS key にを使用するか、別の KMS key から を使用するかを選択できます AWS アカウント。

を指定しない場合 KMS key、文字起こしジョブの出力はデフォルトの Amazon S3 キー (SSE-S3) で暗号化されます。

出力の暗号化を有効にする

1. [出力データ] で、[暗号化] を選択します。

2. KMS key が現在使用している AWS アカウント からか、別の からかを選択します AWS アカウント。現在の のキーを使用する場合は AWS アカウント、KMS key ID からキーを選択します。別の のキーを使用している場合は AWS アカウント、キーの ARN を入力する必要があります。別の のキーを使用するには AWS アカウント、呼び出し元に のkms:Encryptアクセス許可が必要です 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、StartCallAnalyticsJob、、StartMedicalTranscriptionJobまたはStartTranscriptionJobオペレーションの OutputEncryptionKMSKeyIdパラメータ KMS key を使用して を指定する必要があります。

現在の AWS アカウントにあるキーを使用する場合は、次の 4 つの方法のいずれか KMS key で を指定できます。

1. KMS key ID 自体を使用します。例えば、1234abcd-12ab-34cd-56ef-1234567890ab。
2. KMS key ID にエイリアスを使用します。例えば、alias/ExampleAlias。
3. KMS key ID に Amazon リソースネーム (ARN) を使用します。例えば、arn:aws:kms:region:account-ID:key/1234abcd-12ab-34cd-56ef-1234567890ab。
4. エイリアスに ARN KMS key を使用します。例えば、arn:aws:kms:region:account-ID:alias/ExampleAlias。

現在の とは異なる AWS アカウント にあるキーを使用する場合は AWS アカウント、次の 2 つの方法のいずれか KMS key で を指定できます。

1. KMS key ID に ARN を使用します。例えば、arn:aws:kms:region:account-ID:key/1234abcd-12ab-34cd-56ef-1234567890ab。
2. エイリアスに ARN KMS key を使用します。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 KMS keyを使用するためのアクセス許可が必要です。

AWS KMS 暗号化コンテキスト

AWS KMS 暗号化コンテキストは、プレーンテキストの非シークレットキーと値のペアのマップです。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済みデータを表し、データにセキュリティレイヤーを追加します。 では、お客様が指定した Amazon S3 バケットへの文字起こし出力を暗号化するために対称暗号化キー Amazon Transcribe 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 ログ内のプレーンテキストで表示されます (これを使用して暗号化オペレーションを識別および分類できます) 。

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。出力暗号化オペレーション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 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、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 KMS key に基づいて、対称暗号化へのアクセスを制御できます。 https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#cryptographic-operations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。

サービス改善のためのデータ使用をオプトアウトする

デフォルトでは、 は、サービスを開発し、エクスペリエンスを継続的に改善するために処理した音声入力 Amazon Transcribe を保存して使用します。オプトアウト AWS Organizations ポリシー Amazon Transcribe を使用して、コンテンツの開発と改善に使用されないように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、 およびその他の Amazon Transcribe AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 AWS には、監視 Amazon Transcribe、問題発生時の報告、必要に応じて自動アクションを実行するための以下のモニタリングツールが用意されています。

- 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。たとえば、 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。呼び出し元のユーザーとアカウント AWS、呼び出し元の送信元 IP アドレス、呼び出しの発生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す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
- Amazon Transcribe によるモニタリング AWS CloudTrail
- Amazon EventBridge での の使用 Amazon Transcribe

Amazon Transcribe によるモニタリング Amazon CloudWatch

Amazon Transcribe を使用して をモニタリングし CloudWatch、未加工データを収集して読み取り可能なほぼリアルタイムのメトリクスに加工できます。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、Amazon CloudWatch 「メトリクスの使用」を参照してください。

Amazon Transcribe によるモニタリング AWS 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 AWS のサービス Amazon Transcribe によって実行されたアクションを記録するサービスである と統合されています。 は、 のすべての API コール CloudTrail をキャプチャします Amazon Transcribe。これには、 からの呼び出し AWS マネジメントコンソール 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、 バケット Amazon Transcribeへの CloudTrail Amazon S3 イベントを含むイベントの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 マネジメントコンソール 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。

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、 は で有効になります。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 CloudTrail イベントに記録されます。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、 が指定された Amazon S3 バケットにイベント CloudTrail をログファイルとして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、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 マネジメントコンソール、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て対応できます。詳細については、以下を参照してください。

- 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 アクションをログに記録します。たとえば、`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。CloudTrail が Amazon Transcribe API オペレーションをログに記録すると、CloudTrail ログエントリは Amazon S3 URI 値などのリクエストパラメータとレスポンスパラメータの機密情報に空の文字列を使用します。

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、指定された Amazon S3 bucket へのログファイルとしてのイベントの配信を可能にする設定です。 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 parameters. CloudTrail log 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。

> **Note**
>
> CloudTrail が Amazon Transcribe API オペレーションをログに記録すると、CloudTrail ログエントリは Amazon S3 URI 値などのリクエストパラメータとレスポンスパラメータの機密情報に空の文字列を使用します。

```
{
    "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": ""
            }
        },
        "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": ""
                }
            }
        },
        "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": ""
                    },
                    "transcript": {
                        "transcriptFileUri": ""
                    }
                }
            },
            "requestID": "BD8798EACDD16751",
            "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        }
    ]
}
```


Amazon EventBridge での の使用 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、 はイベントをイベントストリーム EventBridge 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。たとえば、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、チャットアプリケーション内の Amazon Q Developerのチャット通知、AWS Console Mobile Applicationのプッシュ通知などの複数のチャネルで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 集計を使用できます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 マネジメントコンソール。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 ルールを作成する方法の例については、Amazon EventBridge 「 ルール」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。たとえば、 AWS Lambda または Amazon Simple Notification Service () ですAmazon SNS。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出力されます。でのイベントの作成と管理の詳細については EventBridge、「 Amazon EventBridge ユーザーガイド」のAmazon EventBridge 「 イベント」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が `COMPLETED`または に変わったときに開始 Amazon Transcribe 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`。
- `source`: イベントのソース。これは常に Amazon Transcribe 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: リクエストが行われた AWS リージョン 。
- 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"),
        "FailureReason": "failure reason", // only present when JobStatus is 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。
- FailureReason: このフィールドは JobStatus が FAILED の場合にのみ存在し、失敗の理由を示します。
- 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://amzn-s3-demo-bucket/my-output-files/my-
redacted-file.JSON",
            "TranscriptFileUri": "s3://amzn-s3-demo-bucket/my-output-files/my-
file.JSON"
        },
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/my-output-files/my-redacted-
file.WAV",
            "RedactedMediaFileUri": "s3://amzn-s3-demo-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。

AWS HealthScribe ストリーム後分析イベント

ClinicalNoteGenerationResult が から IN_PROGRESSに変わるなど、a AWS HealthScribe ストリーム後分析オペレーションの状態が変わるとCOMPLETED、 AWS HealthScribe は次の情報を含むイベントを生成します。

```
{
   "version":"0",
   "id":"event ID",
   "detail-type":"MedicalScribe Post Stream Analytics Update",
   "source":"aws.transcribe",
   "account":"111122223333",
   "time":"timestamp",
   "region":"us-east-1",
   "resources":[],
   "detail":{
      "SessionId": <SessionID>,
      "UpdateType": "ClinicalNoteGenerationResult",
      "ClinicalNoteGenerationResult": {
          "ClinicalNoteOutputLocation": s3://amzn-s3-demo-bucket/clinical-note-output-files/clinical-notes.JSON,
          "TranscriptOutputLocation": s3://amzn-s3-demo-bucket/my-output-files/my-file.JSON,
```

```
            "Status": <IN_PROGRESS | COMPLETED | FAILED>,
            "FailureReason": <failure_reason>
        }
       }
}
```

- UpdateType: イベントを生成したストリーム後分析オペレーションのタイプ。結果オブジェクトの内容は、UpdateType によって異なります。
- SessionId: AWS HealthScribe ストリームの識別番号。この ID を使用して、元のストリーミングセッションを識別し、イベントを生成したストリーム後分析を検索します。
- Status: ストリーム後分析オペレーションのステータス。これは、IN_PROGRESS、COMPLETED、または FAILED です。
- ClinicalNoteOutputLocation: ClinicalNoteGenerationResult の出力 Amazon S3 バケットの URI。
- TranscriptOutputLocation: トランスクリプトの 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。

> **Note**
>
> このイベントは CreateVocabulary API オペレーションにのみ適用されます。

```
{
    "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AWS のサービス プログラムによるスコープ」の「コンプライアンス」を参照して、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、「Downloading Reports in AWS Artifact」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。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

の耐障害性 Amazon Transcribe

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。 は、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェー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することができます。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、フォールトトレランス、および拡張性が優れています。

AWS リージョン および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。

のインフラストラクチャセキュリティ Amazon Transcribe

マネージドサービスである Amazon Transcribe は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS 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環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS が発行した API コールを使用して、ネットワーク Amazon Transcribe 経由で にアクセスします。クライアントは次をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。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。

での脆弱性の分析と管理 Amazon Transcribe

設定と IT コントロールは、 AWS とお客様の間の責任共有です。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 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するには、インターフェイス VPC エンドポイント Amazon Transcribe を作成します。インターフェイスエンドポイントは、インターネットゲートウェイAWS PrivateLink、NAT デバイス、VPN 接続、または Direct Connect 接続を使用せずに Amazon Transcribe APIs にプライベートにアクセスできるテクノロジーである を利用しています。VPC 内のインスタンスは、 Amazon Transcribe APIs と通信するためにパブリック IP アドレスを必要としません。VPC と の間のトラフィック Amazon Transcribe 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、Amazon VPC 「 ユーザーガイド」の「インターフェイスエンドポイントのプロパティと制限」を確認してください。

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。

のインターフェイス VPC エンドポイントの作成 Amazon Transcribe

AWS マネジメントコンソール または を使用して、 Amazon Transcribe サービスの VPC Amazon VPC エンドポイントを作成できます AWS CLI。詳細については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントの作成」を参照してください。

でバッチ文字起こしを行う場合は 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.*us-west-2*.transcribe

で文字起こしをストリーミングするには 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.*us-west-2*.transcribestreaming

エンドポイントのプライベート DNS を有効にすると、 などの のデフォルト DNS 名 Amazon Transcribe を使用して AWS リージョンに API リクエストを行うことができますtranscribestreaming.us-east-2.amazonaws.com。

詳細については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントを介したサービスへアクセスする」を参照してください。

の VPC エンドポイントポリシーの作成 Amazon Transcribe

ストリーミングサービスまたはバッチ文字起こし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 VPC エンドポイントにアタッチできます 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。

- 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 エンドポイントによるサービスのアクセスコントロール」を参照してください。

例: Amazon Transcribe バッチ文字起こしアクションの VPC エンドポイントポリシー

以下は、 Amazon Transcribeでバッチ文字起こしを行うためのエンドポイントポリシーの例です。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: 文字起こしアクションを Amazon Transcribe ストリーミングするための VPC エンドポイントポリシー

以下は、 Amazon Transcribeでストリーミング文字起こしを行うためのエンドポイントポリシーの例です。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

自分と共有されているサブネットで VPC エンドポイントを作成、説明、変更、または削除することはできません。ただし、VPC エンドポイントを共有するサブネットで使用することはできます。VPC 共有の詳細については、「 Amazon Virtual Private Cloud ガイド」の「他のアカウントと VPC を共有する」を参照してください。

のセキュリティのベストプラクティス Amazon Transcribe

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必ずしも適切または十分でない可能性があるので、処方箋ではなく、あくまで有用な検討事項として使用します。

- 暗号化コンテキストなどのデータ AWS KMS 暗号化を使用する

  AWS KMS 暗号化コンテキストは、プレーンテキストの非シークレットキーと値のペアのマップです。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 暗号化コンテキスト」を参照してください。
- 可能な限り、一時的な認証情報を使用する

  長期的に使用できるアクセスキーの代わりに、可能な限り一時的な認証情報を使用します。プログラムによるアクセスと長期的な認証情報を持つ IAM 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すすめします。

  詳細については、「アクセスキーの更新」および「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。
- Amazon Transcribe アクセスを必要とするアプリケーションと AWS サービスに IAM ロールを使用する

IAM ロールを使用して、 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します Amazon Transcribe。ロールを使用する場合、パスワードやアクセスキーなどの長期的な認証情報を Amazon EC2 インスタンスや AWS サービスに配布する必要はありません。 IAM ロールは、アプリケーションが AWS 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アクセス許可を提供できます。

詳細については、「IAM ロール」および「ロールの一般的なシナリオ: ユーザー、アプリケーション、およびサービス」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の使用

  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 Transcribe.では、タグは、文字起こしジョブ、カスタム語彙、カスタム語彙フィルター、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制御 を参照してください。

- AWS モニタリングツールを使用する

  モニタリングは、 および Amazon Transcribe 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な部分です。 Amazon Transcribe を使用してモニタリングできます CloudTrail。

  詳細については、Amazon Transcribe によるモニタリング AWS CloudTrail を参照してください。

- 有効化 AWS Config

  AWS Config は、 AWS リソースの設定を評価、監査、評価できます。を使用すると AWS Config、 AWS リソース間の設定や関係の変更を確認できます。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することもできます。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、「 とは」を参照してください AWS Config。

Amazon Transcribe 医療

Amazon Transcribe Medical は、医師が指示したメモ、薬物安全性モニタリング、遠隔医療の予約、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 Transcribe Medical 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 (バッチ) の文字起こしを通じて利用できます。

> **Important**
>
> Amazon Transcribe Medical は、専門的な医療アドバイス、診断、または治療に代わるものではありません。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、結果は適切なトレーニングを受けた人間によるレビュー担当者によってレビューおよび検証される必要があります。 Amazon Transcribe 医療文字起こしは、トレーニングを受けた医療専門家による正確性と健全な医療判断のレビュー後に、患者ケアシナリオでのみ使用してください。

Amazon Transcribe Medical は責任共有モデルに基づいて運営されており、 AWS は Amazon Transcribe Medical を実行するインフラストラクチャの保護に責任を負い、お客様はデータの管理に責任を負います。詳細については、「 責任共有モデル」を参照してください。

Amazon Transcribe Medical は米国英語 (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 などの可逆音声形式を使用してください。 Amazon Transcribe Medical 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。

トランスクリプトの分析には、 AWS のサービスなどの他の 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	音声入力
心臓病学	なし	ストリーミングのみ
神経学	なし	ストリーミングのみ
オンコロジー	なし	ストリーミングのみ

専門分野	サブ専門分野	音声入力
プライマリケア	家庭医療	バッチ、ストリーミング
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	なし	ストリーミングのみ
泌尿器科	なし	ストリーミングのみ

利用可能なリージョンとクォータ

コール分析は、以下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
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、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

リージョン	文字起こしタイプ
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
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、 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。

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

音声タイプに応じて、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、既に作成した 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、「 の開始方法 Amazon Simple Storage Service」を参照してください。

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 Medical では、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、スピーカーパーティショニングを有効にしたり、オーディオ内のチャネルを特定したりすることもできます。詳細については、「スピーカーパーティショニングを有効にする」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 Medical は、医学用語と測定値を文字起こしできます。 Amazon Transcribe Medical は、発話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。Amazon Transcribe Medical が医学用語と測定値に使用する規則のリストについては、このページ

の表を参照してください。[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。[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	出力の例
摂氏	C	患者の体温は 37.4℃ である。
摂氏	C	患者の体温は 37.4℃ 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。

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	出力の例
X オーバー Y	X/Y	患者の BP は 120/80 であった。
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 Medical は、数字を単語ではなく数字として文字起こしします。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します。	• 「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) します。	• 「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」で示します。	• 「three hundred and three dot five」 > 303.5 • 「three point twenty three」 > 3.23 • 「zero point four」 > 0.4

ルール	説明
	• 「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%
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
時刻は数字に変換します。	• 「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 桁に変換します。 20 世紀、21 世紀、および 22 世紀にのみ有効です。	• 「nineteen sixty two」 > 1962 • 「the year is twenty twelve」 > the year is 2012 • 「twenty nineteen」 > 2019 • 「twenty one thirty」 > 2130
日付は数字に変換します。	• 「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」で区切ります。	• 「twenty three to thirty seven」 > 23 to 37

医療会話の文字起こし

Amazon Transcribe Medical を使用すると、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、臨床医と患者の医療会話を文字起こしできます。バッチ文字起こしジョブを使用すると、音声ファイルを変換することができます。 Amazon Transcribe 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、文字起こしジョブまたはストリームで臨床医の医療専門分野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: ストリーミング文字起こしのみ利用可能
- 神経学: ストリーミング文字起こしでのみ利用可能
- 腫瘍学: ストリーミング文字起こしでのみ利用可能
- 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: ストリーミング文字起こしでのみ利用可能

医療用カスタム語彙を使用すると、文字起こしの精度を向上することができます。医療用カスタム語彙の詳細については、「医療用カスタム語彙による文字起こしの精度の向上」を参照してください。

デフォルトでは、 Amazon Transcribe Medical は信頼度が最も高い文字起こしを返します。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。

文字起こし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

- 医療に関する会話の音声ファイルの文字起こし
- リアルタイムストリームで医療に関する会話の文字起こし
- スピーカーパーティショニングを有効にする
- マルチチャネルの音声文字起こし

医療に関する会話の音声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、医療に関する会話の音声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。StartMedicalTranscriptionJob API または AWS マネジメントコンソールで、バッチ文字起こしジョブを開始できます。

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 マネジメントコンソール

臨床医と患者の対話の文字起こし (AWS マネジメントコンソール)

を使用して臨床医と患者の対話を AWS マネジメントコンソール 文字起こしするには、文字起こしジョブを作成し、音声入力タイプの会話を選択します。

1. AWS マネジメントコンソール 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. [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. 残りのフィールドでは、音声ファイル Amazon S3 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. [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) を使用した医療に関する会話の文字起こしをするには

- 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、文字起こしを行う音声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 の場合、CONVERSATION を指定します。

f. OutputBucketName の場合、文字起こし結果を保存する Amazon S3 バケットを指定します。

以下は、 を使用してPRIMARYCARE専門分野の臨床医と患者の医療会話を AWS SDK for Python (Boto3) 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://amzn-s3-demo-bucket/my-input-files/my-audio-file.flac"
transcribe.start_medical_transcription_job(
      MedicalTranscriptionJobName = job_name,
      Media = {
        'MediaFileUri': job_uri
      },
      OutputBucketName = 'amzn-s3-demo-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://amzn-s3-demo-bucket/my-input-files/my-audio-
file.flac"
      },
      "OutputBucketName": "amzn-s3-demo-bucket",
      "OutputKey": "my-output-files/",
      "LanguageCode": "en-US",
      "Specialty": "PRIMARYCARE",
      "Type": "CONVERSATION"
  }
```

リアルタイムストリームで医療に関する会話の文字起こし

医療に関する会話の音声ストリームは、HTTP/2 または WebSocket プロトコルで文字起こしを行うことができます。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、WebSocket ストリームの設定 を参照してください。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。次の例をガイダンスとして使用して、WebSocket リクエストの specialty URL パラメータまたは Specialty APIの StartMedicalStreamTranscription パラメータの値を指定します。

- 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 マネジメントコンソール

ストリーミングの医療に関する会話を文字起こしする (AWS マネジメントコンソール)

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ムで AWS マネジメントコンソール 文字起こしするには、医療会話を文字起こししし、ストリームを開始し、マイクで話し始めるオプションを選択します。

1. AWS マネジメントコンソール 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. [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、カスタム語彙を使用します。カスタム語彙を有効にするには、使用するカスタム語彙の名前に VocabularyName パラメータの値を設定します。詳細については、「医療用カスタム語彙による文字起こしの精度の向上」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、「スピーカーパーティショニングを有効にする」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 URL を作成します。この URL には、アプリケーションと Amazon Transcribe 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、「WebSocket ストリームの設定」を参照してください。

署名付き URL 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、カスタム語彙を使用します。カスタム語彙を有効にするには、使用するカスタム語彙の名前に vocabulary-name の値を設定します。詳細については、「医療用カスタム語彙による文字起こしの精度の向上」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、「スピーカーパーティショニングを有効にする」を参照してください。

署名付き URL の作成方法の詳細については、「WebSocket ストリームの設定」を参照してください。

スピーカーパーティショニングを有効にする

Amazon Transcribe Medical でスピーカーパーティショニングを有効にするには、スピーカー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。

スピーカーダイアライゼーションを有効にすると、 Amazon Transcribe Medical は各スピーカーの発話に各スピーカーの一意の識別子をラベル付けします。発話とは発言の単位であり、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。入力音声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ム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、スピーカー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ムでスピーカーパーティショニングを有効にする

バッチ文字起こしで、スピーカーパーティショニングを有効にする

StartMedicalTranscriptionJob API または AWS マネジメントコンソールでバッチ文字起こしジョブをスピーカーパーティショニングを有効にできます。これにより、臨床医と患者の会話で話者ごとに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 マネジメントコンソール

を使用して文字起こ AWS マネジメントコンソール しジョブでスピーカーダイアライゼーションを有効にするには、音声識別を有効にしてから、スピーカーパーティショニングを有効にします。

1. AWS マネジメントコンソール 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. [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. [次へ] を選択します。
6. [音声識別] を有効にします。
7. 音声識別タイプ では、[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、音声ファイル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、スピーカーパーティショニングを有効にする

- 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、文字起こしを行う音声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、CONVERSATION を指定します。

f. にはOutputBucketName、文字起こし結果を保存する Amazon S3 バケットを指定します。

g. Settings オブジェクトの場合、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、 を使用して AWS SDK for Python (Boto3) 、スピーカーパーティショニングを有効にしたプライマリケア臨床医患者ダイアログ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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media={
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-bucket',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    OutputBucketName = 'amzn-s3-demo-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ニングを有効にした文字起こしジョブの文字起こし結果を示しています。

```
{
    "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を文字起こしする (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://amzn-s3-demo-bucket/my-input-files/my-audio-
file.flac"
      },
      "OutputBucketName": "amzn-s3-demo-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ムでスピーカー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ムでスピーカーをパーティション化し、音声にラベルを付けるには、 AWS マネジメントコンソール または ストリーミングリクエストを使用します。スピーカーパーティショニングは、ストリーミング内のスピーカーが 2～5 人の場合に最も効果的です。 Amazon Transcribe Medical はストリーム内で 5 人以上のスピーカーをパーティション分割できますが、その数を超えるとパーティションの精度が低下します。

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 URL を使用します。URL には、アプリケーションと Amazon Transcribe 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。

マイクで話されている音声のスピーカーパーティショニングを有効にする (AWS マネジメントコンソール)

を使用して AWS マネジメントコンソール 、臨床医と患者の会話のリアルタイムストリーム、または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できます。

1. AWS マネジメントコンソール にサインインします。
2. ナビゲーションペインで、 Amazon Transcribe Medical でリアルタイム文字起こしを選択します。
3. 音声入力タイプの場合、文字起こしする医療音声の種類を選択します。
4. [追加設定] では、[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 ストリーム内のスピーカーパーティショニングを有効にする

医療に関する関係の会話の HTTP/2 ストリーム内のスピーカーパーティショニングを有効にする場合、`StartMedicalStreamTranscription` API 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を指定します。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`

医療に関する会話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 リクエスト内のスピーカーパーティショニングを有効にする

API によって WebSocket ストリーミング内のスピーカーをパーティション化する場合、次の形式を使用して WebSocket リクエストをスタートするための署名付き URL を作成し、`show-speaker-label` を `true` 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 Transcribe Medical は、スピーカーの変更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ムを切断します。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、ストリーミング文字起こし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。スピーカーパーティショニングを有効にすると、各単語には完全に文字起こしされた音声セグメントのSpeakerラベルが付けられます。Amazon Transcribe Medical はこのラベルを使用して、ストリーム内の各スピーカーに一意の整数を割り当てます。speaker-change の値を持つ Type 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、WebSocket ストリームの設定 を参照してください。

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オーディオファイルまたはストリームがある場合は、チャネル識別を使用して、それらの各チャネルから音声を文字起こしできます。 Amazon Transcribe Medical は、各チャネルから音声を個別に文字起こししします。各チャ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ム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。

- バッチ文字起こし – AWS マネジメントコンソール および StartMedicalTranscriptionJob API
- ストリーミング文字起こし - WebSocket ストリーミングと StartMedicalStreamTranscription API

マルチチャネルの音声ファイルの文字起こし

オーディオファイルを文字起こしすると、 Amazon Transcribe Medical は各チャネルの項目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。あるチャネルの人が別のチャネルで話しかけると、各チャ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。

バッチ文字起こしジョブでマルチチャネルオーディオを文字起こしするには、 AWS マネジメントコンソール または `StartMedicalTranscriptionJob` API を使用します。

AWS マネジメントコンソール

を使用してバッチ文字起こ AWS マネジメントコンソール しジョブでチャネル識別を有効にするには、音声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、 のオーディオ識別のサブセットです AWS マネジメントコンソール。

1. AWS マネジメントコンソール 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. [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. [次へ] を選択します。
6. [音声識別] を有効にします。
7. 音声識別タイプ 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` 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を指定します。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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_medical_transcription_job(
      MedicalTranscriptionJobName = job_name,
      Media = {
        'MediaFileUri': job_uri
      },
      OutputBucketName = 'amzn-s3-demo-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

バッチ文字起こしジョブを使用して、マルチチャネル音声ファイルの文字起こしをする (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://amzn-s3-demo-bucket/my-input-files/my-audio-
file.flac"
      },
      "OutputBucketName": "amzn-s3-demo-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ムの文字起こし

HTTP/2 ストリーミングまたは WebSocket 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、StartMedicalStreamTranscription API 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 Service Quotas」を参照してください。

HTTP/2 ストリームでのマルチチャネル音声の文字起こし

HTTP/2 ストリームでのマルチチャネル音声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` - 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 ストリーム内のマルチチャネル音声の文字起こし

WebSocket ストリーム内でスピーカーをパーティション化する場合、次の形式を使用して署名付き URL を作成し、WebSocket リクエストを開始します。`enable-channel-identification` を `true` に、ストリームのチャネル数を `number-of-channels` に指定します。署名付き URI には、アプリケーションと Amazon Transcribe 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のパラメータは、「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、「WebSocket ストリームの設定」を参照してください。

マルチチャネルストリーミング出力

ストリーミング文字起こし出力は、HTTP/2 リクエストと WebSocket 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 Transcribe Medical を使用すると、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、臨床医が作成したメディカルノートを文字起こしできます。バッチ文字起こしジョブを使用すると、音声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 Amazon Transcribe を生成するように、文字起こしジョブまたはストリームで臨床医の医療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 : ストリーミング文字起こしのみ利用可能
- 神経学: ストリーミング文字起こしでのみ利用可能
- 腫瘍学: ストリーミング文字起こしでのみ利用可能
- 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: ストリーミング文字起こしでのみ利用可能
- 泌尿器学 : ストリーミング文字起こしでのみ利用可能

カスタム語彙を使用すると、文字起こしの精度を向上することができます。医療用カスタム語彙の詳細については、「医療用カスタム語彙による文字起こしの精度の向上」を参照してください。

デフォルトでは、 Amazon Transcribe Medical は信頼度が最も高い文字起こしを返します。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。

文字起こし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音声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、医療に関する会話の音声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。`StartMedicalTranscriptionJob` API または AWS マネジメントコンソールで、バッチ文字起こしジョブを開始できます。

`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 マネジメントコンソール

臨床医と患者の対話の文字起こし (AWS マネジメントコンソール)

を使用して臨床医と患者の対話を AWS マネジメントコンソール 文字起こしするには、文字起こしジョブを作成し、音声入力タイプの会話を選択します。

1. AWS マネジメントコンソール 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. [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 – ディクテーション
5. 残りのフィールドでは、音声ファイル Amazon S3 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. [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) を使用した医療に関する会話の文字起こしをするには

- 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 パラメータに、文字起こしを行う音声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、DICTATION を指定します。

  f. OutputBucketName の場合、文字起こし結果を保存する Amazon S3 バケットを指定します。

  以下は、 を使用してPRIMARYCARE専門分野の臨床医のメディカルディクテーションを AWS SDK for Python (Boto3) 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://amzn-s3-demo-bucket/my-input-files/my-audio-file.flac"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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文字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 (AWS CLI) で、スピーカーパーティショニングを有効にする

- 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://amzn-s3-demo-bucket/my-input-files/my-audio-file.flac"
      },
      "OutputBucketName": "amzn-s3-demo-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 ストリームを使用して、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。また、 を使用して AWS マネジメントコンソール 、自分や他のユーザーが直接話す音声をマイクに書き起こすこともできます。

HTTP/2 ストリームまたは WebSocket ストリームでは、次の医療分野の音声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。次の例をガイダンスとして使用して、WebSocket リクエストの `specialty` URL パラメータまたは `Specialty` APIの `StartMedicalStreamTranscription` パラメータの値を指定します。

- 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、カスタム語彙を使用します。カスタム語彙を有効にするには、使用するカスタム語彙の名前に `vocabulary-name` の値を設定します。

を使用してマイクに話されたディクテーションを文字起こしする AWS マネジメントコンソール

を使用してメディカルディクテーションのストリーミングオーディオを AWS マネジメントコンソール 文字起こしするには、メディカルディクテーションを文字起こししし、ストリームを開始し、マイクで話し始めます。

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 マネジメントコンソール)

1. AWS マネジメントコンソール 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. [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 URL を作成します。この URI には、アプリケーションと Amazon Transcribe Medical 間のオーディオ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、「WebSocket ストリームの設定」を参照してください。

署名付き URL 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 URL の作成方法」の詳細については、「WebSocket ストリームの設定」を参照してください。

医療用カスタム語彙による文字起こしの精度の向上

Amazon Transcribe Medical の文字起こしの精度を向上させるには、1 つ以上の医療カスタム語彙を作成して使用します。カスタム語彙 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、これらの単語やフレーズを書き起こす際の Amazon Transcribe Medical のパフォーマンスを向上させるのに役立ちます。

Medical を使用する際、お客様は自身のデータの整合性について責任を負います Amazon Transcribe 。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語彙を作成し、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語彙を 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、 に最大 100 個のカスタム語彙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 を超えることはできません。に用意できるカスタム語彙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。

カスタム語彙は、アメリカ英語 (en-US) で利用できます。

トピック

医療用カスタム語彙のテキストファイルを作成する

カスタム語彙を作成する場合、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。各フィールドは、ドメイン固有の用語がどのように発音されるか、または文字起こしにこれらの用語を表示する方法を Amazon Transcribe Medical に伝えます。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

医療用カスタム語彙を作成するには、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。

- 各列を 1 つの Tab character Amazon Transcribe で区切ります。列をスペースまたは複数の Tab 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 Medical の文字セット」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 Medical の文字セット」を参照してください。DisplayAs 列を指定しない場合、 Amazon Transcribe Medical は出力ファイルのPhrase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 Medical の文字セット」を参照してください。米国英語は Amazon Transcribe Medical で使用できる唯一の言語です。
- **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 Medical の文字セット」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 Medical は出力の入力ファイルのPhrase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 Transcribe Medical を使用してカスタム語彙を作成する AWS リージョン のと同じ .txt の Amazon S3 バケットに 拡張子でファイルを保存します。

カスタム語彙ファイルは、 LFと の両方のCRLF行末をサポートしています。

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		ALL
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
ALL		eɪ ɛ l ɛ l	A.L.L。
	ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia		atrioventricular-nodal-reentrant-tachycardia

テキストファイルを使用して医療用カスタム語彙を作成する

カスタム語彙を作成するには、単語またはフレーズのコレクションを含むテキストファイルを準備しておく必要があります。 Amazon Transcribe Medical はこのテキストファイルを使用して、それらの単語またはフレーズの文字起こし精度を向上させるために使用できるカスタム語彙を作成します。`CreateMedicalVocabulary` API または Amazon Transcribe Medical コンソールを使用してカスタム語彙を作成できます。

AWS マネジメントコンソール

を使用してカスタム語 AWS マネジメントコンソール 彙を作成するには、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 マネジメントコンソール 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、カスタム語彙を選択します。

3. 名前 を使用する場合、語彙の設定で、カスタム語彙の名前を選択します。
4. Amazon S3で音声ファイルまたはビデオファイルの場所を指定します。
   - 語彙の設定 の S3 の語彙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 の語彙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。

カスタム語彙の処理ステータスが AWS マネジメントコンソールで確認できます。

API

医療用カスタム語彙を作成 (API)するには

- StartTranscriptionJob API では、以下のものを指定します。
  a. LanguageCode の場合、en-US を指定します。
  b. ではVocabularyFileUri、カスタム語彙の定義に使用するテキストファイル Amazon S3 の場所を指定します。
  c. VocabularyName の場合、カスタム語彙の名前を指定します。指定する名前は、 内で一意である必要があります AWS アカウント。

カスタム語彙の処理状況を表示する場合、GetMedicalVocabulary API を使用します。

以下は、 AWS SDK for Python (Boto3) を使用してカスタム語彙を作成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')
vocab_name = "my-first-vocabulary"
response = transcribe.create_medical_vocabulary(
    VocabularyName = job_name,
    VocabularyFileUri = 's3://amzn-s3-demo-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、スピーカーパーティショニングを有効にする

- 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://amzn-s3-demo-bucket/my-vocabularies/my-vocabulary-
file.txt \
--language-code en-US
```

医療用カスタム語彙を使用した音声ファイルの文字起こし

StartMedicalTranscriptionJob または を使用して AWS マネジメントコンソール 、文字起こしの精度を向上させるためにカスタム語彙を使用する文字起こしジョブを開始します。

AWS マネジメントコンソール

1. AWS マネジメントコンソール 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. [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. [次へ] を選択します。
6. カスタマイズで、カスタム語彙を有効にします。
7. 語彙選択で、カスタム語彙を選択します。
8. [作成] を選択します。

API

バッチ文字起こしジョブ (API) を使用して音声ファイル内のスピーカーパーティショニングを有効にするには

- 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、文字起こしを行う音声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. OutputBucketName の場合、文字起こし結果を保存する Amazon S3 バケットを指定します。
  g. Settings オブジェクトの場合、以下を指定します。
     - VocabularyName – カスタム語彙の名前です。

次のリクエストでは AWS SDK for Python (Boto3) 、 を使用して、カスタム語彙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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_medical_transcription_job(
   MedicalTranscriptionJobName = job_name,
   Media = {
       'MediaFileUri': job_uri
   },
   OutputBucketName = 'amzn-s3-demo-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)
```

医療用カスタム語彙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文字起こしの精度を向上させるために、HTTP/2 ストリームまたは WebSocket ストリームを使用してカスタム語彙を使用できます。HTTP/2 リクエストを開始する場合、`StartMedicalStreamTranscription` API を使用します。カスタム語彙は、、`StartMedicalStreamTranscription` API AWS マネジメントコンソール、または 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文字起こし (AWS マネジメントコンソール)

を使用してメディカルディクテーションのストリーミングオーディオを AWS マネジメントコンソール 文字起こしするには、メディカルディクテーションを文字起こししし、ストリームを開始し、マイクで話し始めます。

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 マネジメントコンソール)

1. AWS マネジメントコンソール 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. 医療専門分野の場合、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、カスタム語彙 を選択します。
   - 語彙選択で、カスタム語彙を選択します。
6. [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 ストリームでスピーカーパーティションを有効にする

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

- host: (前の例の AWS リージョン 「us-west-2」) を呼び出している で更新 AWS リージョン します。有効な のリストについては AWS リージョン、AWS リージョン 「」および「エンドポイント」を参照してください。
- authorization: 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 を使用した AWS リクエストの署名」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。前の例で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 を使用した AWS リクエストの署名」を参照してください。

- x-amz-date: 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。
- x-amzn-transcribe-session-id: ストリーミングセッションの名前。
- x-amzn-transcribe-media-encoding: 入力音声に使用されるエンコード。有効な値のリストについては、「StartMedicalStreamTranscription」または「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。
- 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: 入力オーディオのサンプルレート (ヘルツ単位) 。 は 8,000 Hz ~ 48,000 Hz の範囲 Amazon Transcribe をサポートします。電話音声などの低品質音声は、通常 8,000 Hz 前後です。高品質の音声は、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音声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: 文字起こしに使用したいボキャブラリーの名前。
- x-amzn-transcribe-specialty: 文字起こしの対象となる医療専門分野。
- x-amzn-transcribe-type: ディクテーションにするか会話にするかを選択します。
- x-amzn-transcribe-show-speaker-label: ダイアライゼーションを有効にするには、この値が true でなければなりません。
- 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 による WebSocket ストリーム内のスピーカーをパーティション化する場合、次の形式を使用して WebSocket リクエストをスタートするための署名付き URL を作成し、vocabulary-name を カスタム語彙の名前に特定します。

```
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 Medical の文字セット

Amazon Transcribe Medical でカスタム語彙を使用するには、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。

文字	コード	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

文字	コード	文字	コード
g	0067	ɔ	0254
h	0068	ɔɪ	0254 026A
i	0069	ə	0259
j	006A	ɛ	025B
k	006B	ɝ	025D
l	006C	g	0261
l̩	006C 0329	ɪ	026A
m	006D	ɹ	0279
n	006E	ʃ	0283
n̩	006E 0329	ʊ	028A
oʊ	006F 028A	ʌ	028C
p	0070	ʍ	028D
s	0073	ʒ	0292
t	0074	ʤ	02A4
u	0075	ʧ	02A7
v	0076	θ	03B8

文字起こし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、文字起こし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ムまたはバッチ文字起こしジョブを使用して PHI を識別できます。

独自の後処理を使用して、文字起こし出力で識別された 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
  - 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 Transcribe Medical は、1996 年の医療保険の相互運用性と説明責任に関する法律 (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。音声ファイル内の PHI の識別については、「音声ファイル内の 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ムでの PHI の識別 」を参照してください。

トピック

音声ファイル内の PHI の識別

バッチ文字起こしジョブを使用して、音声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。個人健康情報 (PHI) 識別を有効にすると、 Amazon Transcribe Medical は文字起こし結果で識別した PHI にラベルを付けます。 Amazon Transcribe Medical が識別できる PHI の詳細については、「」を参照してください文字起こしにおける個人の健康情報 (PHI) の特定。

`StartMedicalTranscriptionJob` API または AWS マネジメントコンソールでバッチ文字起こしジョブを開始できます。

AWS マネジメントコンソール

を使用して臨床医と患者の対話を AWS マネジメントコンソール 文字起こしするには、文字起こしジョブを作成し、音声入力タイプの会話を選択します。

音声ファイルを書き起こし、PHI (AWS マネジメントコンソール) を識別するには

1. AWS マネジメントコンソール 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. [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 – に固有の文字起こしジョブの名前 AWS アカウント。
    b. 音声入力タイプ – [会話] または [ディクテーション]。
5. 残りのフィールドでは、音声ファイル Amazon S3 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. [次へ] を選択します。
7. [音声設定]で、[PHI 識別] を選択します。
8. [作成] を選択します。

API

バッチ文字起こしジョブ (API) を使用して音声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. MedicalTranscriptionJobName の場合、 AWS アカウントに一意の名前を指定します。

  b. LanguageCode の場合、音声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、文字起こしを行う音声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 の場合、CONVERSATION または DICTATION のいずれかを指定します。

  f. OutputBucketName の場合、文字起こし結果を保存する Amazon S3 バケットを指定します。

  以下は、 を使用してオーディオファイルを文字起こししし AWS SDK for Python (Boto3) 、患者の PHI を識別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-transcription-job"
job_uri = "s3://amzn-s3-demo-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 = 'amzn-s3-demo-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 を識別した場合の文字起こし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": [{
                "id": 0,
            "start_time": "0.0",
            "end_time": "0.37",
            "alternatives": [{
                "confidence": "0.9993",
                "content": "The"
            }],
            "type": "pronunciation"
        }, {
                "id": 1,
            "start_time": "0.37",
            "end_time": "0.44",
            "alternatives": [{
                "confidence": "0.9981",
                "content": "patient's"
            }],
            "type": "pronunciation"
        }, {
                "id": 2,
            "start_time": "0.44",
            "end_time": "0.52",
```

```
            "alternatives": [{
                "confidence": "1.0",
                "content": "name"
            }],
            "type": "pronunciation"
        }, {
                "id": 3,
            "start_time": "0.52",
            "end_time": "0.92",
            "alternatives": [{
                "confidence": "1.0",
                "content": "is"
            }],
            "type": "pronunciation"
        }, {
                "id": 4,
            "start_time": "0.92",
            "end_time": "0.9989",
            "alternatives": [{
                "confidence": "1.0",
                "content": "Bertrand"
            }],
            "type": "pronunciation"
        }, {
                "id": 5,
            "alternatives": [{
                "confidence": "0.0",
                "content": "."
            }],
            "type": "punctuation"
        }],
        "entities": [{
            "content": "Bertrand",
            "category": "PHI*-Personal*",
            "startTime": 0.92,
            "endTime": 1.2,
            "confidence": 0.9989
        }],
        "audio_segments": [
            {
                "id": 0,
                "transcript": "The patient's name is Bertrand.",
                "start_time": "0.0",
                "end_time": "0.9989",
```

```
                "items": [
                    0,
                    1,
                    2,
                    3,
                    4,
                    5
                ]
            }
        ]
    },
    "status": "COMPLETED"
}
```

AWS CLI

バッチ文字起こしジョブ (AWS CLI) を使用して音声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://amzn-s3-demo-bucket/my-input-files/my-audio-file.flac" \
--output-bucket-name amzn-s3-demo-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ムでの PHI の識別

HTTP/2 ストリーミングまたは WebSocket ストリーミングのいずれかで、個人の健康情報 (PHI) を識別できます。PHI 識別をアクティブ化すると、 Amazon Transcribe Medical は文字起こし結果で識別した PHI にラベルを付けます。 Amazon Transcribe Medical が識別できる PHI の詳細については、「」を参照してください[文字起こし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

を使用してマイクによって取得された音声 AWS マネジメントコンソール を文字起こししし、PHI を特定するには、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、ストリームを開始して、コンピュータのマイクで話し始めます。

を使用してディクテーション内の PHI を識別するには AWS マネジメントコンソール

1. AWS マネジメントコンソール 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. [音声入力タイプ] で、[ディクテーション] を選択します。
4. 追加設定の場合、[PHI の識別] 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、音声のサンプルレートを指定します。
- content-identification-type の場合、PHI を指定します。

WebSocket ストリーミング内の PHI の識別

PHI 識別を有効にした状態で WebSocket 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のパラメータは、「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 Transcribe Medical を使用すると、信頼度が最も高い文字起こしが得られます。ただし、信頼性レベルが低い追加の文字起こしを返すように Amazon Transcribe Medical を設定できます。

代替文字起こしを使用して、変換された音声のさまざまな解釈を確認します。たとえば、ユーザーが文字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 マネジメントコンソール または `StartMedicalTranscriptionJob` API を使用して代替文字起こしを生成できます。

AWS マネジメントコンソール

を使用して代替文字起こし AWS マネジメントコンソール を生成するには、ジョブを設定するときに代替結果を有効にします。

1. AWS マネジメントコンソール 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. [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. [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、音声ファイル内のスピーカーごとにテキストを分離する

- 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 パラメータで、文字起こしする音声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. OutputBucketName の場合、トランスクリプション結果を保存する Amazon S3 バケットを指定します。

  g. Settings オブジェクトの場合、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、 を使用して AWS SDK for Python (Boto3) 、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://amzn-s3-demo-bucket/my-input-files/my-audio-file.flac
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-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

プライマリケアを実践している臨床医と患者との間の会話を音声ファイルに文字起こしする (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":"amzn-s3-demo-bucket",
      "Media": {
          "MediaFileUri": "s3://amzn-s3-demo-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)

インターフェイス VPC エンドポイントを作成することで、VPC と Amazon Transcribe Medical 間のプライベート接続を確立できます。インターフェイスエンドポイントは、インターネットゲートウェイAWS PrivateLink、NAT デバイス、VPN 接続、または Direct Connect 接続なしで Amazon Transcribe Medical APIs にプライベートにアクセスできるテクノロジーである を利用しています。VPC 内のインスタンスは、パブリック IP アドレスがなくても Amazon Transcribe Medical APIs と通信できます。VPC と Amazon Transcribe Medical 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、「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 Medical VPC エンドポイントに関する考慮事項

Amazon Transcribe Medical 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 VPC 「 ユーザーガイド」の「インターフェイスエンドポイントのプロパティと制限」を確認してください。

Amazon Transcribe Medical 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 Medical 用のインターフェイス VPC エンドポイントの作成

Amazon Transcribe Medical サービスの VPC エンドポイントは、 AWS マネジメントコンソール または を使用して作成できます AWS CLI。詳細については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントの作成」を参照してください。

Amazon Transcribe Medical でバッチ文字起こしを行う場合は、次のサービス名を使用して VPC エンドポイントを作成します。

- com.amazonaws.*us-west-2*.transcribe

Amazon Transcribe Medical で文字起こしをストリーミングするには、次のサービス名を使用して VPC エンドポイントを作成します。

- com.amazonaws.*us-west-2*.transcribestreaming

エンドポイントのプライベート DNS を有効にすると、 AWS リージョンなどの のデフォルトの DNS 名を使用して Amazon Transcribe Medical に API リクエストを行うことができますtranscribestreaming.us-east-2.amazonaws.com。

詳細については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントを介したサービスへアクセスする」を参照してください。

Amazon Transcribe Medical ストリーミング用の VPC エンドポイントポリシーの作成

Amazon Transcribe Medical への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。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。
- 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: Amazon Transcribe Medical ストリーミング文字起こしアクションの VPC エンドポイントポリシー

以下は、 Amazon Transcribe Medical で文字起こしをストリーミングするためのエンドポイントポリシーの例です。エンドポイントにアタッチすると、このポリシーは、すべてのリソースのすべてのプリンシパルに対して、リストされている Amazon Transcribe Medical アクションへのアクセスを許可します。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartMedicalStreamTranscription",
         ],
         "Resource":"*"
      }
   ]
}
```

例: Amazon Transcribe Medical バッチ文字起こしアクションの VPC エンドポイントポリシー

以下は、 Amazon Transcribe Medical でのバッチ文字起こしのエンドポイントポリシーの例です。エンドポイントにアタッチすると、このポリシーは、すべてのリソースのすべてのプリンシパルに対して、リストされている Amazon Transcribe Medical アクションへのアクセスを許可します。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartMedicalTranscriptionJob"
         ],
         "Resource":"*"
      }
   ]
}
```

共有サブネット

自分と共有されているサブネットで VPC エンドポイントを作成、説明、変更、または削除することはできません。ただし、VPC エンドポイントを共有するサブネットで使用することはできます。VPC 共有の詳細については、「 Amazon Virtual Private Cloud ガイド」の「他のアカウントと VPC を共有する」を参照してください。

AWS HealthScribe

> **Amazon Connect Health Ambient エージェントの概要**
>
> アンビエントエージェントを使用して、ヘルスケアワークフローを強化できるようになりました。詳細については、Amazon Connect Health Ambient ドキュメント」を参照してください。

AWS HealthScribe は、HIPAA 対応の機械学習 (ML) 機能です。音声認識と生成 AI を組み合わせて、患者と臨床医の会話を文字起こしし、easy-to-review臨床ノートを生成します。 AWS HealthScribe は、医療ソフトウェアベンダーがドキュメントの負担を軽減し、コンサルティング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するのに役立ちます。このサービスは、豊富な会話トランスクリプトを自動的に提供し、話者の役割を識別し、会話を分類し、医学用語を抽出して、予備的な臨床ノートを生成します。 AWS HealthScribe はこれらの機能を組み合わせて、個別の AI サービスを統合および最適化する必要をなくし、実装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 によって生成される結果は確率的であり、音質、背景ノイズ、話者の明確性、医学用語の複雑さ、文脈固有の言語のニュアンス、機械学習や生成 AI の性質など、さまざまな要因によって常に正確であるとは限りません。 AWS HealthScribe 出力は、臨床医や医療書記の補助的な役割で使用されるように設計されています。 AWS HealthScribe 出力は、電子医療記録の一部として、トレーニングを受けた医療専門家による健全な医療判断の正確性と帰属を確認した後、患者ケアシナリオでのみ使用する必要があります。 AWS HealthScribe 出力は、専門的な医療アドバイス、診断、または治療に代わるものではなく、

また、疾患や病状を治療、緩和、予防、または診断することを意図したものではありません。

トピック

セキュリティ

AWS HealthScribe AWS は責任共有モデルの下で動作し、 AWS HealthScribe を実行するインフラストラクチャを保護する責任は が負い、データの管理はユーザーが負います。詳細については、「責任共有モデル」を参照してください。

デフォルトでは、 AWS HealthScribe は保管時の暗号化を提供し、 Amazon S3マネージドキーを使用して機密データを保護します。 AWS HealthScribe 文字起こしジョブを作成するとき、またはストリームを開始するときに、カスタマーマネージドキーを指定できます。これにより、暗号化に 2 番目のレイヤーが追加されます。詳細については、「AWS HealthScribe の保管時のデータ暗号化」を参照してください。

サービスの可用性

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。

技術要件

- サポートされている言語: 米国英語 (en-US)

- 推奨されるオーディオ形式: 可逆オーディオ (FLAC や WAV など)
- エンコード: PCM 16 ビット
- サンプルレート: 16,000 Hz 以上

サポートされている医療専門分野

AWS HealthScribe は現在、以下の専門分野をサポートしています。

- アレルギー免疫科
- 心臓病学
- 皮膚科
- 内分泌科
- 消化器科
- 血液/腫瘍科
- 感染性疾患
- 腎臓病
- 神経学
- 産婦人科
- オンコロジー
- 眼科
- 整形外科
- 耳鼻科
- 疼痛治療
- 小児科
- プライマリケア
- 精神医学
- 呼吸器内科
- リウマチ
- 外科手術
- 泌尿器科

ワークフロー

AWS HealthScribe ワークフローには、文字起こしジョブとストリーミングが含まれます。文字起こしジョブの実行またはストリームを完了すると、 AWS HealthScribe はトランスクリプトファイルを生成し、各会話ターンのターンごとの文字起こし出力とインサイトを提供します。また、概要とエビデンスのリンクを含む臨床ドキュメントファイルを生成します。詳細については、「AWS HealthScribe トランスクリプトファイル」および「AWS HealthScribe 臨床ドキュメントファイル」を参照してください。

- 文字起こしジョブ – 文字起こしジョブを使用すると、 AWS HealthScribe は Amazon S3 バケットから完了した医療相談メディアファイルを分析します。以下は、 AWS HealthScribe 文字起こしジョブに固有の API オペレーションです。
  - StartMedicalScribeJob
  - ListMedicalScribeJobs
  - GetMedicalScribeJob
  - DeleteMedicalScribeJob

  詳細 (コード例を含む) については、「AWS HealthScribe 文字起こしジョブ」を参照してください。
- ストリーミング – AWS HealthScribe ストリーミングは、1 つのチャネルでオーディオストリームを受け入れ、もう 1 つのチャネルでオーディオ文字起こしを提供するリアルタイムの HTTP2 ベースの双方向サービスです。

  以下は、 AWS HealthScribe ストリーミングに固有の API オペレーションです。
  - StartMedicalScribeStream
  - GetMedicalScribeStream

  詳細 (コード例を含む) については、「AWS HealthScribe ストリーミング」を参照してください。

AWS HealthScribe トランスクリプトファイル

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を含む標準のturn-by-turn文字起こし出力に加えて、 AWS HealthScribe には以下が用意されています。

- 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。

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_0`、`CLINICIAN_1`、および `PATIENT_0`、`PATIENT_1` 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 — 会話に存在する臨床的な関連エンティティ (`ClinicalEntity`) を抽出します。AWS HealthScribe はAmazon Comprehend Medical でサポートされているすべての臨床エンティティを検出します。

文字起こしジョブからの文字起こしの例については、「文字起こしジョブの出力例」の文字起こし出力を参照してください。ストリーミングからのトランスクリプトの例については、「ストリーミング文字起こしの出力例」のトランスクリプト出力を参照してください。

AWS HealthScribe 臨床ドキュメントファイル

AWS HealthScribe は、臨床メモの概要に次のいずれかのテンプレートを使用できます。デフォルトは `HISTORY_AND_PHYSICAL` です。

- HISTORY_AND_PHYSICAL: 臨床ドキュメントの主要なセクションの概要を提供します。セクションの例には、主訴、現在の病歴、システムのレビュー、過去の医療履歴、評価、計画などがあります。
- GIRPP: 目標に向けた患者の進捗状況に基づいて概要を提供します。セクションの例には、目標、介入、対応、進捗状況、計画などがあります。

- BIRP: 患者の行動パターンと反応に焦点を当てます。セクションの例には、動作、介入、反応、計画などがあります。
- SIRP: 治療の状況コンテキストを強調します。セクションの例には、状況、介入、反応、計画などがあります。
- DAP: 臨床ドキュメントの簡略化された形式を提供します。セクションの例には、データ、評価、計画などがあります。
- BH_SOAP: 動作ヘルスに焦点を当てたドキュメント形式。セクションの例には、主観、目的、評価、計画などがあります。
- PH_SOAP: 物理ヘルスに焦点を当てたドキュメント形式。セクションの例には、主観、目的、評価、計画などがあります。

使用するテンプレートを指定するには、以下を実行します。

- 文字起こしジョブの場合は、StartMedicalScribeJob API オペレーションの `Settings` の ClinicalNoteGenerationSettings の `NoteTemplate` で使用するテンプレートを指定します。
- ストリーミングの場合は、MedicalScribeConfigurationEvent の `PostStreamAnalyticsSettings` の ClinicalNoteGenerationSettings の `NoteTemplate` で使用するテンプレートを指定します。

Medical Scribe のコンテキスト

`MedicalScribeContext` オブジェクトには、患者固有の詳細を使用して臨床メモの生成をカスタマイズするために使われるコンテキスト情報が含まれています。

`MedicalScribeContext` オブジェクトには以下のコンポーネントが含まれます。

- PatientContext - 臨床メモの生成をカスタマイズするために使用される患者固有の情報が含まれます。これには、以下が含まれます。
  - Pronouns - 臨床メモ生成のコンテキストとして提供する患者の優先代名詞。これらの代名詞は、生成された臨床出力で患者を参照するときに使用されます。

Medical Scribe のコンテキストを設定する方法については、「MedicalScribeContext」を参照してください。

トピック

- HISTORY_AND_PHYSICAL テンプレートセクション

- GIRPP テンプレートセクション
- BIRP テンプレートセクション
- SIRP テンプレートセクション
- DAP テンプレートセクション
- BH_SOAP テンプレートセクション
- PH_SOAP テンプレートセクション

HISTORY_AND_PHYSICAL テンプレートセクション

HISTORY_AND_PHYSICAL インサイトテンプレートには、以下のセクションが含まれています。

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。
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。
PHYSICAL_EXAMINATION	患者の身体システムおよびバイタルの身体検査から得た臨床医の検出結果のドキュメント。
PAST_FAMILY_HISTORY	患者の家族に見られる病状に関する情報。
PAST_SOCIAL_HISTORY	患者の社会生活、習慣、職業、健康に影響する環境要因に関する詳細。

セクション	説明
DIAGNOSTIC_TESTING	検査、画像検査、その他の診断手順の結果と解釈。

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。これにより、ユーザーはアプリケーションで概要の精度を検証できます。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共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。

文字起こしジョブの臨床ドキュメントファイルの例については、「文字起こしジョブの出力例」の臨床ドキュメントファイルの例を参照してください。ストリーミングからの臨床ドキュメントファイルの例については、「ストリーミング文字起こしの出力例」の臨床ドキュメントファイルの例を参照してください。

GIRPP テンプレートセクション

GIRPP インサイトテンプレートには、以下のセクションが含まれています。

セクション	説明
目標	治療を通じて対処する必要がある特定された問題、課題、または動作。
介入	特定された目標を達成するために臨床医が使用する特定の治療、方法、または手法。
[応答]	参加レベル、反応、フィードバックなど、患者が介入にどのように反応したか。
進行状況	患者の改善や障壁の観察など、治療目標に向けた動きに関する臨床医の評価。
プラン	今後の介入、宿題の割り当て、紹介など、治療の次のステップ。

BIRP テンプレートセクション

BIRP インサイトテンプレートには、以下のセクションが含まれています。

セクション	説明
行動	患者が提示する問題と、治療に対する反応。
介入	特定された目標を達成するために臨床医が使用する特定の治療、方法、または手法。
[応答]	参加レベル、反応、フィードバックなど、患者が介入にどのように反応したか。
プラン	今後の介入、宿題の割り当て、紹介など、治療の次のステップ。

SIRP テンプレートセクション

SIRP インサイトテンプレートには、以下のセクションが含まれています。

セクション	説明
状況	患者が提示する問題と、治療を求める目標。
介入	特定された目標を達成するために臨床医が使用する特定の治療、方法、または手法。
[応答]	参加レベル、反応、フィードバックなど、患者が介入にどのように反応したか。
プラン	今後の介入、宿題の割り当て、紹介など、治療の次のステップ。

DAP テンプレートセクション

DAP インサイトテンプレートには、以下のセクションが含まれています。

セクション	説明
データ	治療を求める患者の理由と患者に関する情報。
評価	患者の状態に関する臨床医の診断。
プラン	今後の介入、宿題の割り当て、紹介など、治療の次のステップ。

BH_SOAP テンプレートセクション

BH_SOAP インサイトテンプレートには、以下のセクションが含まれています。

セクション	説明
主観	クライアントの治療目標や経験、現在および過去の課題。
目的	クライアントに関するデータと事実。
評価	患者の状態に関する臨床医の診断。
プラン	今後の介入、宿題の割り当て、紹介など、治療の次のステップ。

PH_SOAP テンプレートセクション

PH_SOAP インサイトテンプレートには、以下のセクションが含まれています。

セクション	説明
主観	クライアントの治療目標や経験、現在および過去の課題。
目的	クライアントに関するデータと事実。
評価	患者の状態に関する臨床医の診断。

セクション	説明
プラン	今後の介入、宿題の割り当て、紹介など、治療の次のステップ。

AWS HealthScribe 文字起こしジョブ

An AWS HealthScribe 文字起こしジョブは、Amazon S3 バケットからのメディアファイルを処理します。メディアファイルを処理すると、患者と臨床医の会話を書き起こし、医療相談を分析して、トランスクリプトファイルと臨床ドキュメントファイルの 2 つの JSON 出力ファイルを生成します。

以下は、 AWS HealthScribe 文字起こしジョブに固有の API オペレーションです。

- StartMedicalScribeJob
- ListMedicalScribeJobs
- GetMedicalScribeJob
- DeleteMedicalScribeJob

AWS HealthScribe 文字起こしジョブの開始

CLI または AWS SDKs を使用して AWS HealthScribe AWS ジョブを開始できます。

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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-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 は以下のリクエストボディを含みます。

```
{
  "MedicalScribeJobName": "my-first-medical-scribe-job",
  "Media": {
    "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
   },
  "OutputBucketName": "amzn-s3-demo-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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_medical_scribe_job(
    MedicalScribeJobName = job_name,
    Media = {
      'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-bucket',
    DataAccessRoleArn = 'arn:aws:iam::111122223333:role/ExampleRole',
    Settings = {
      'ShowSpeakerLabels': false,
      'ChannelIdentification': true
    },
    ChannelDefinitions = [
      {
        'ChannelId': 0,
        'ParticipantRole': 'CLINICIAN'
      }, {
        'ChannelId': 1,
        'ParticipantRole': 'PATIENT'
      }
    ]
)
while True:
    status = transcribe.get_medical_scribe_job(MedicalScribeJobName = job_name)
    if status['MedicalScribeJob']['MedicalScribeJobStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

Note

AWS マネジメントコンソールは現在、 AWS HealthScribe ジョブをサポートしていません。

文字起こしジョブの出力例

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

An AWS HealthScribe トランスクリプトファイル (StartMedicalScribeJobリクエストから) 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": 0.17,
        "EndAudioTime": 0.9,
        "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://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
   },
  "OutputBucketName": "amzn-s3-demo-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 ストリーミング

AWS HealthScribe ストリーミングを使用すると、医療会話をリアルタイムで文字起こしできます。AWS HealthScribe ストリーミングは、1 つのチャネルでオーディオストリームを受け入れ、他のチャネルでオーディオ文字起こしを提供するリアルタイムの HTTP2 ベースの双方向サービスです。ストリーミングが完了すると、 AWS HealthScribe はストリームの内容を分析し、トランスクリプト JSON ファイルと臨床ノート JSON ファイルを生成します。

ストリーミングを開始するには、StartMedicalScribeStream API オペレーションを使用します。この API は、オーディオイベントのストリーミングに使用する HTTP2 ベースの双方向チャネルを開始します。

ストリームを開始するときは、まず `MedicalScribeConfigurationEvent` でストリーム設定を指定します。このイベントには、チャネル定義、暗号化設定、および集約されたトランスクリプトや臨床メモ生成の出力設定などのストリーム後分析設定が含まれます。

オーディオのストリーミングを開始したら、次のようにストリーミングを管理します。

- 完了したら、ストリーム後分析で結果の処理を開始するには、 Typeを MedicalScribeSessionControlEventとして送信END_OF_SESSIONし、 AWS HealthScribe が分析を開始します。
- ストリーミングを一時停止するには、MedicalScribeSessionControlEvent を送信せずに入力ストリームを完了します。
- 一時停止したストリームを再開するには、StartMedicalScribeStream API オペレーションを使用して同じ SessionId を指定します。これは、最初にストリーミングを開始したときに使用した SessionId です。

トピック

ガイドラインと要件

以下は、 AWS HealthScribe ストリーミングのガイドラインと要件です。

- オーディオイベントを送信する前に、まず MedicalScribeConfigurationEvent でストリーミング設定を指定する必要があります。
- ストリーム後分析を実行するには、MedicalScribeConfigurationEvent の ResourceAccessRoleArn に正しいアクセス許可が必要です。詳細については、「ResourceAccessRoleArn ロールのアクセス許可」を参照してください。
- セッションは、最初のストリーム作成から 5 時間以内に何度でも再開できます。
- すべてのストリーミングリクエストで、セッション中に最長 2 時間のオーディオをストリーミングできます。
- デフォルトでは、 AWS HealthScribe は保管時の暗号化を提供し、 Amazon S3マネージドキーを使用して機密データを保護します。ストリームを開始するときに、2 番目の暗号化レイヤーの AWS KMS キーを指定できます。には、 AWS KMS キーを使用するためのアクセス許可ResourceAccessRoleArnが必要です。詳細については、「AWS HealthScribe の保管時のデータ暗号化」を参照してください。
- AWS SDKs、SDK で AWS HealthScribe ストリーミングを使用できます。
- ストリーミングを終了した後に LimitExceededException 例外が発生した場合は、セッションを再起動しても、ストリーム後分析を生成できます。ストリーミングを再起動する

には、StartMedicalScribeStream API を使用し、同じ SessionID を使用します。次に、MedicalScribeSessionControlEventを Type として送信END_OF_SESSIONし、 AWS HealthScribe が分析を開始します。

ResourceAccessRoleArn ロールのアクセス許可

ストリーム後分析を実行するには、 ResourceAccessRoleArnの が Amazon S3 出力バケットにアクセスでき、指定した場合は AWS KMS キーにアクセスできるMedicalScribeConfigurationEvent必要があります。また、ロールの信頼ポリシーは、transcribe.streaming.amazonaws.com サービスがロールを引き受けるアクセス許可を付与する必要があります。

以下は、Amazon S3 バケットのアクセス許可と AWS KMS キーのアクセス許可を付与する IAM ポリシーの例です。詳細については、「AWS HealthScribe の保管時のデータ暗号化」を参照してください。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:DescribeKey",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey*"
            ],
            "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
```

```
            "Effect": "Allow"
        }
    ]
}
```

以下に示しているのは、信頼ポリシーの例です。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "transcribe.streaming.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Starting AWS HealthScribe ストリーミング文字起こし

次のコード例は、 AWS SDKs を使用して AWS HealthScribe ストリーミング文字起こしを設定する方法を示しています。

トピック

- SDK for Java 2.x
- ストリーミング文字起こしの出力例

SDK for Java 2.x

次の例では、SDK for Java 2.x を使用してストリーミングを設定し、StartMedicalScribeStream リクエストを行います。

```
package org.example;

import io.reactivex.rxjava3.core.BackpressureStrategy;
import io.reactivex.rxjava3.core.Flowable;
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;
import
 software.amazon.awssdk.services.transcribestreaming.model.ClinicalNoteGenerationSettings;
import software.amazon.awssdk.services.transcribestreaming.model.LanguageCode;
import software.amazon.awssdk.services.transcribestreaming.model.MediaEncoding;

import
 software.amazon.awssdk.services.transcribestreaming.model.MedicalScribeInputStream;
import
 software.amazon.awssdk.services.transcribestreaming.model.MedicalScribePostStreamAnalyticsSett
import
 software.amazon.awssdk.services.transcribestreaming.model.MedicalScribeSessionControlEventType
import
 software.amazon.awssdk.services.transcribestreaming.model.MedicalScribeTranscriptEvent;
import
 software.amazon.awssdk.services.transcribestreaming.model.MedicalScribeTranscriptSegment;
import
 software.amazon.awssdk.services.transcribestreaming.model.StartMedicalScribeStreamRequest;
import
 software.amazon.awssdk.services.transcribestreaming.model.StartMedicalScribeStreamResponseHand
import
 software.amazon.awssdk.services.transcribestreaming.model.medicalscribeinputstream.DefaultConf

import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.TargetDataLine;
import java.io.BufferedInputStream;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.util.Arrays;
import java.util.concurrent.CompletableFuture;


public class HealthScribeStreamingDemoApp {
    private static final int CHUNK_SIZE_IN_BYTES = 6400;
    private static final int SAMPLE_RATE = 16000;
    private static final Region REGION = Region.US_EAST_1;
    private static final String sessionId = "1234abcd-12ab-34cd-56ef-123456SAMPLE";
    private static final String bucketName = "amzn-s3-demo-bucket";
    private static final String resourceAccessRoleArn =
 "arn:aws:iam::123456789012:role/resource-access-role";
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .httpClientBuilder(NettyNioAsyncHttpClient.builder())
                .region(REGION)
                .build();
        try {
            StartMedicalScribeStreamRequest request =
 StartMedicalScribeStreamRequest.builder()
                    .languageCode(LanguageCode.EN_US.toString())
                    .mediaSampleRateHertz(SAMPLE_RATE)
                    .mediaEncoding(MediaEncoding.PCM.toString())
                    .sessionId(sessionId)
                    .build();

            MedicalScribeInputStream endSessionEvent =
 MedicalScribeInputStream.sessionControlEventBuilder()
                    .type(MedicalScribeSessionControlEventType.END_OF_SESSION)
                    .build();

            CompletableFuture<Void> result = client.startMedicalScribeStream(
                    request,
                    new AudioStreamPublisher(getStreamFromMic(),
 getConfigurationEvent(),endSessionEvent),
                    getMedicalScribeResponseHandler());
            result.get();
```

```
            client.close();
        } catch (Exception e) {
            System.err.println("Error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }

    private static AudioInputStream getStreamFromMic() throws LineUnavailableException
{
        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        AudioFormat format = new AudioFormat(SAMPLE_RATE, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            throw new LineUnavailableException("The audio system microphone line is not
supported.");
        }
        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        int bufferSize = (CHUNK_SIZE_IN_BYTES / format.getFrameSize()) *
format.getFrameSize();
        line.open(format);
        line.start();

        // Create a wrapper class that can be closed when Enter is pressed
        AudioInputStream audioStream = new AudioInputStream(line);

        // Start a thread to monitor for Enter key
        System.out.println("Recording... Press Enter to stop");
        Thread monitorThread = new Thread(() -> {
            try {
                System.in.read();
                line.stop();
                line.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        });
        monitorThread.setDaemon(true);  // Set as daemon thread so it doesn't prevent
JVM shutdown
        monitorThread.start();

        return new AudioInputStream(
            new BufferedInputStream(new AudioInputStream(line)),
```

```
            format,
            AudioSystem.NOT_SPECIFIED
        );
    }

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

    private static StartMedicalScribeStreamResponseHandler
 getMedicalScribeResponseHandler() {

        return StartMedicalScribeStreamResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(Throwable::printStackTrace)
            .onComplete(() -> {
                System.out.println("=== All records streamed successfully ===");
            })
            .subscriber(event -> {
                if (event instanceof MedicalScribeTranscriptEvent) {
                    MedicalScribeTranscriptSegment segment =
 ((MedicalScribeTranscriptEvent) event).transcriptSegment();
                    if (segment != null && segment.content() != null && !
segment.content().isEmpty()) {
                        System.out.println(segment.content());
                    }
                }
            })
            .build();
    }

    private static DefaultConfigurationEvent getConfigurationEvent() {
        MedicalScribePostStreamAnalyticsSettings postStreamSettings =
 MedicalScribePostStreamAnalyticsSettings
                .builder()
                .clinicalNoteGenerationSettings(
                        ClinicalNoteGenerationSettings.builder()
                                .outputBucketName(bucketName)
                                .build()
                )
                .build();
```

```
        return (DefaultConfigurationEvent)
MedicalScribeInputStream.configurationEventBuilder()
                .resourceAccessRoleArn(resourceAccessRoleArn)
                .postStreamAnalyticsSettings(postStreamSettings)
                .build();
    }

    private static class AudioStreamPublisher implements
Publisher<MedicalScribeInputStream> {
        private final InputStream audioInputStream;
        private final MedicalScribeInputStream configEvent;
        private final MedicalScribeInputStream endSessionEvent;

        private AudioStreamPublisher(AudioInputStream audioInputStream,
                                     MedicalScribeInputStream configEvent,
                                     MedicalScribeInputStream endSessionEvent) {
            this.audioInputStream = audioInputStream;
            this.configEvent = configEvent;
            this.endSessionEvent = endSessionEvent;
        }

        @Override
        public void subscribe(Subscriber<? super MedicalScribeInputStream> subscriber)
{
            createAudioFlowable()
                    .doOnComplete(() -> {
                        try {
                            audioInputStream.close();
                        } catch (IOException e) {
                            throw new UncheckedIOException(e);
                        }
                    })
                    .subscribe(subscriber);
        }

        private Flowable<MedicalScribeInputStream> createAudioFlowable() {
            // Start with config event
            Flowable<MedicalScribeInputStream> configFlow = Flowable.just(configEvent);

            // Create audio chunk flowable
            Flowable<MedicalScribeInputStream> audioFlow = Flowable.create(emitter -> {
                byte[] buffer = new byte[CHUNK_SIZE_IN_BYTES];
                int bytesRead;
```

```
                try {
                    while (!emitter.isCancelled() && (bytesRead =
 audioInputStream.read(buffer)) > 0) {
                        byte[] audioData = bytesRead < buffer.length
                                ? Arrays.copyOfRange(buffer, 0, bytesRead)
                                : buffer;

                        MedicalScribeInputStream audioEvent =
 MedicalScribeInputStream.audioEventBuilder()
                                .audioChunk(SdkBytes.fromByteArray(audioData))
                                .build();

                        emitter.onNext(audioEvent);
                    }
                    emitter.onComplete();
                } catch (IOException e) {
                    emitter.onError(e);
                }
            }, BackpressureStrategy.BUFFER);

            // End with session end event
            Flowable<MedicalScribeInputStream> endFlow =
 Flowable.just(endSessionEvent);

            // Concatenate all flows
            return Flowable.concat(configFlow, audioFlow, endFlow);
        }
    }
}
```

ストリーミング文字起こしの出力例

ストリーミングが完了すると、 AWS HealthScribe はストリームの内容を分析し、トランスクリプト JSON ファイルと臨床ノート JSON ファイルを生成します。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

以下は、ストリーミングセッションからの a AWS HealthScribe トランスクリプトファイルの例です。

```
{
    "Conversation": {
        "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"
        }],
        "ConversationId": "sampleConversationUUID",
        "LanguageCode": "en-US",
        "SessionId": "sampleSessionUUID",
        "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": "uuid4"
        }, {
            "BeginAudioTime": 6.96,
            "Content": "Yes.",
            "EndAudioTime": 7.88,
            "ParticipantDetails": {
                "ParticipantRole": "CLINICIAN_0"
            },
            "SectionDetails": {
                "SectionName": "SUBJECTIVE"
            },
```

```
            "SegmentId": "uuid5"
        }, {
            "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": "uuid6"
        }]
    }
}
```

臨床ドキュメント出力の例

以下は、ストリーミングセッションからの a AWS HealthScribe 臨床ドキュメントインサイトファイルの例です。

```
{
  "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."
```

```
          }
        ]
      }
    ],
    "SessionId": "sampleSessionUUID"
  }
}
```

AWS HealthScribe の保管時のデータ暗号化

デフォルトでは、 AWS HealthScribe は保管時の暗号化を提供し、 AWS HealthScribe managed AWS Key Management Service (AWS KMS) キーを使用して顧客の機密データを保護します。デフォルトでは、保管中のデータを暗号化することで、機密データの保護に伴う運用のオーバーヘッドと複雑な作業を軽減できます。また、セキュリティを重視したアプリケーションを構築することで、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。 AWS HealthScribe 文字起こしジョブを作成するとき、またはストリームを開始するときに、カスタマーマネージドキーを指定できます。これにより、暗号化に 2 番目のレイヤーが追加されます。

- AWS HealthScribe マネージド AWS KMS キー — AWS HealthScribe はデフォルトで AWS HealthScribe managed AWS Key Management Service (AWS KMS) キーを使用して、中間ファイルを自動的に暗号化します。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。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のアクションの実施やプログラムの変更を行う必要はありません。
- カスタマーマネージドキー — AWS HealthScribe は、作成、所有、管理する対称カスタマーマネージドキーの使用をサポートし、既存の AWS 所有の暗号化に 2 番目の暗号化レイヤーを追加します。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 ポリシーと権限の確立と維持
  - キーポリシーの有効化と無効化
  - キー暗号化マテリアルのローテーション
  - タグを追加する
  - キーエイリアスの作成
  - 削除のためのキースケジューリング

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 は、個人を特定できるデータを無償で保護するために AWS、所有キーを使用して保管時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。
> 詳細については AWS KMS、「 とは AWS Key Management Service」を参照してください。

トピック

- カスタマーマネージドキーの作成
- AWS HealthScribe のカスタマーマネージドキーの指定
- AWS KMS 暗号化コンテキスト
- AWS HealthScribe の暗号化キーのモニタリング

カスタマーマネージドキーの作成

対称カスタマーマネージドキーは AWS マネジメントコンソール、、または AWS KMS APIs を使用して作成できます。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。キーポリシーは、カスタマーマネージドキーの作成時に指定できます。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。

AWS KMS AWS HealthScribe のキーポリシー

StartMedicalScribeJob または `DataAccessRole` `ResourceAccessRole`StartMedicalScribeStream リクエストで 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。 StartMedicalScribeJob DataAccessRole (文字起こしジョブ用) または ResourceAccessRole (ストリーミング用) として別のアカウントでカスタマーマネージドキーを使用するには、キーポリシーのそれぞれのロールを以下のアクションで信頼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 がキーを検証できるようにします。

以下は、 AWS HealthScribe ストリーミング用のカスタマーマネージドキーを使用するためのアクセス許可を ResourceAccessRole クロスアカウントに付与するために使用できるキーポリシーの例です。このポリシーを文字起こしジョブに使用するには、DataAccessRole ARN を使用するように Principal を更新し、暗号化コンテキストを削除または変更します。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid": "AllowAccessForKeyAdministrators",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
         },
         "Action": [
           "kms:*"
         ],
         "Resource": "*"
      },
      {
         "Sid": "AllowAccessToResourceAccessRoleForMedicalScribe",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/ResourceAccessRole"
         },
         "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
```

```
            "kms:GenerateDataKey*"
         ],
         "Resource": "*",
         "Condition": {
            "StringEquals": {
                "kms:EncryptionContext:aws:us-east-1:transcribe:medical-
scribe:session-id": "1234abcd-12ab-34cd-56ef-123456SAMPLE"
            }
         }
      },
      {
         "Sid": "AllowAccessToResourceAccessRoleForDescribeKey",
         "Effect": "Allow",
         "Principal": {
             "AWS": "arn:aws:iam::111122223333:role/ResourceAccessRole"
         },
         "Action": "kms:DescribeKey",
         "Resource": "*"
      }
   ]
}
```

アクセスロールの IAM ポリシーアクセス許可

DataAccessRole または ResourceAccessRole にアタッチされた IAM ポリシーは、カスタマー管理のキーとロール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関係なく、必要な AWS KMS アクションを実行するためのアクセス許可を付与する必要があります。また、ロールの信頼ポリシーは、ロールを引き受ける権限を AWS HealthScribe に付与する必要があります。

次の IAM ポリシーの例は、 AWS HealthScribe ストリーミングの ResourceAccessRole アクセス許可を付与する方法を示しています。このポリシーを文字起こしジョブに使用するには、`transcribe.streaming.amazonaws.com` を `transcribe.amazonaws.com` に置き換え、暗号化コンテキストを削除または変更します。

JSON

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

```
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey*"
            ],
            "Resource": "arn:aws:kms:us-west-2:111122223333:key/KMS-Example-
KeyId",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "transcribe.streaming.amazonaws.com",
                    "kms:EncryptionContext:aws:us-east-1:transcribe:medical-
scribe:session-id": "1234abcd-12ab-34cd-56ef-123456SAMPLE"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-west-2:111122223333:key/KMS-Example-
KeyId",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "transcribe.streaming.amazonaws.com"
                }
            }
        }
    ]
}
```

ResourceAccessRole の信頼ポリシーの例を次に示します。DataAccessRole の場合は、`transcribe.streaming.amazonaws.com` を `transcribe.amazonaws.com` に置き換えます。

JSON

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

```
            "Effect": "Allow",
            "Principal": {
                "Service": "transcribe.streaming.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:transcribe:us-
west-2:111122223333:*"
                }
            }
        }
    ]
}
```

ポリシーでのアクセス許可の指定またはキーアクセスのトラブルシューティングの詳細については、「 AWS Key Management Service デベロッパーガイド」を参照してください。 https://docs.aws.amazon.com//kms/latest/developerguide/policy-evaluation.html#example-no-iam

AWS HealthScribe 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は、文字起こしジョブまたはストリーミングの 2 番目のレイヤー暗号化として指定できます。

- 文字起こしジョブの場合は、StartMedicalScribeJob API オペレーションの OutputEncryptionKMSKeyId でキーを指定します。
- ストリーミングの場合は、MedicalScribeConfigurationEvent の MedicalScribeEncryptionSettings でキーを指定します。

AWS KMS 暗号化コンテキスト

AWS KMS 暗号化コンテキストは、プレーンテキストの非シークレットキーと値のペアのマップです。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済みデータを表し、データにセキュリティレイヤーを追加します。 AWS HealthScribe では、お客様が指定した Amazon S3 バケットへの AWS HealthScribe 出力を暗号化するために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 ログ内のプレーンテキストで表示されます (これを使用して暗号化オペレーションを識別および分類できます) 。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の暗号化コンテキストを提供するには、OutputEncryptionKMSKeyId パラメータが対称 AWS KMS キー ID を参照する必要があります。
> ストリーミングでは、MedicalScribeConfigurationEvent の MedicalScribeEncryptionSettings で `KmsEncryptionContext` のキーと値のペアを指定します。
> IAM ポリシーでAWS KMS 条件キーを使用して、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。

AWS HealthScribe 暗号化コンテキスト

AWS HealthScribe は、すべての暗号化オペレーションで同じ AWS Key Management Service 暗号化コンテキストを使用します。暗号化コンテキストは、任意のものにカスタマイズできる文字列から文字列へのマップです。

```
"encryptionContext": {
   "ECKey": "ECValue"
   ...
}
```

AWS HealthScribe ストリームの場合、デフォルトのサービス生成の暗号化コンテキストは次のとおりです。このコンテキストは、指定した暗号化コンテキストの上に適用されます。

```
"encryptionContext": {
  "aws:<region>:transcribe:medical-scribe:session-id":
 "1234abcd-12ab-34cd-56ef-123456SAMPLE"
}
```

AWS HealthScribe 文字起こしジョブの場合、デフォルトのサービス生成の暗号化コンテキストは次のとおりです。このコンテキストは、指定した暗号化コンテキストの上に適用されます。

```
"encryptionContext": {
  "aws:<region>:transcribe:medical-scribe:job-name": "<job-name>",
  "aws:<region>:transcribe:medical-scribe:start-time-epoch-ms": "<job-start-time>"
}
```

暗号化コンテキストを指定しない場合、サービス生成の暗号化コンテキストのみがすべての AWS KMS 暗号化オペレーションに使用されます。

暗号化コンテキストを使用した AWS HealthScribe のモニタリング

対称カスタマーマネージドキーを使用して AWS HealthScribe で保管中のデータを暗号化する場合、監査レコードとログの暗号化コンテキストを使用して、カスタマーマネージドキーがどのように使用されているかを特定することもできます。暗号化コンテキストは、 AWS CloudTrail または CloudWatch Logs によって生成されたログにも表示されます。

暗号化コンテキストを使用してカスタマーマネージドキーへのアクセスを制御する

対称カスタマーマネージドキー (CMK) へのアクセスを制御するための条件として、キーポリシーと IAM ポリシー内の暗号化コンテキストを使用することもできます。

次に、特定の暗号化コンテキストのカスタマーマネージドキーへのアクセスを付与するキーポリシーステートメントの例を示します。このポリシーステートメントの条件では、暗号化コンテキストを指定する暗号化コンテキスト制約がグラントに必要です。

JSON

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

```
                "AWS": "arn:aws:iam::111122223333:role/ResourceAccessRole"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey*"
            ],
            "Resource": "arn:aws:kms:us-west-2:111122223333:key/KMS-Example-
KeyId",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:aws:us-east-1:transcribe:medical-
scribe:session-id": "1234abcd-12ab-34cd-56ef-123456SAMPLE",
                    "kms:EncryptionContext:ECKey": "ECValue"
                }
            }
        },
        {
            "Sid": "AllowAccessToResourceAccessRoleForDescribeKey",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ResourceAccessRole"
            },
            "Action": "kms:DescribeKey",
            "Resource": "arn:aws:kms:us-west-2:111122223333:key/KMS-Example-
KeyId"
        }
    ]
}
```

AWS HealthScribe の暗号化キーのモニタリング

AWS HealthScribe で AWS Key Management Service カスタマーマネージドキーを使用する場合、AWS CloudTrail または CloudWatch ログを使用して AWS HealthScribe が送信するリクエストを追跡できます AWS KMS。

次の例は、 AWS HealthScribe がカスタマーマネージドキーをどのように使用するかをモニタリングするために使用できる CloudTrail Encrypt および Decrypt イベントです。

暗号化

```
{
```

```
   "eventVersion":"1.09",
   "userIdentity":{
      "type":"AssumedRole",
      "principalId":"AROAIGDTESTANDEXAMPLE:Sampleuser01",
      "arn":"arn:aws:sts::123456789012:assumed-role/Admin/Sampleuser01",
      "accountId":"123456789012",
      "accessKeyId":"AKIAIOSFODNN7EXAMPLE3",
      "sessionContext":{
         "sessionIssuer":{
            "type":"Role",
            "principalId":"AROAIGDTESTANDEXAMPLE:Sampleuser01",
            "arn":"arn:aws:sts::123456789012:assumed-role/Admin/Sampleuser01",
            "accountId":"123456789012",
            "userName":"Admin"
         },
         "attributes":{
            "creationDate":"2024-08-16T01:10:05Z",
            "mfaAuthenticated":"false"
         }
      },
      "invokedBy":"transcribe.streaming.amazonaws.com"
   },
   "eventTime":"2024-08-16T01:10:05Z",
   "eventSource":"kms.amazonaws.com",
   "eventName":"Encrypt",
   "awsRegion":"us-east-1",
   "sourceIPAddress":"transcribe.streaming.amazonaws.com",
   "userAgent":"transcribe.streaming.amazonaws.com",
   "requestParameters":{
      "encryptionContext":{
         "aws:us-east-1:transcribe:medical-scribe:session-
id":"1234abcd-12ab-34cd-56ef-123456SAMPLE"
      },
      "encryptionAlgorithm":"SYMMETRIC_DEFAULT",
      "keyId":"1234abcd-12ab-34cd-56ef-1234567890ab"
   },
   "responseElements":null,
   "requestID":"cbe0ac33-8cca-49e5-9bb5-dc2b8dfcb389",
   "eventID":"1b9fedde-aa96-48cc-9dd9-a2cce2964b3c",
   "readOnly":true,
   "resources":[
      {
         "accountId":"123456789012",
         "type":"AWS::KMS::Key",
```

```
         "ARN":"arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
      }
   ],
   "eventType":"AwsApiCall",
   "managementEvent":true,
   "recipientAccountId":"123456789012",
   "eventCategory":"Management"
}
```

Decrypt

```
{
   "eventVersion":"1.09",
   "userIdentity":{
      "type":"AssumedRole",
      "principalId":"AROAIGDTESTANDEXAMPLE:Sampleuser01",
      "arn":"arn:aws:sts::123456789012:assumed-role/Admin/Sampleuser01",
      "accountId":"123456789012",
      "accessKeyId":"AKIAIOSFODNN7EXAMPLE3",
      "sessionContext":{
         "sessionIssuer":{
            "type":"Role",
            "principalId":"AROAIGDTESTANDEXAMPLE:Sampleuser01",
            "arn":"arn:aws:sts::123456789012:assumed-role/Admin/Sampleuser01",
            "accountId":"123456789012",
            "userName":"Admin"
         },
         "attributes":{
            "creationDate":"2024-08-16T20:47:04Z",
            "mfaAuthenticated":"false"
         }
      },
      "invokedBy":"transcribe.streaming.amazonaws.com"
   },
   "eventTime":"2024-08-16T20:47:04Z",
   "eventSource":"kms.amazonaws.com",
   "eventName":"Decrypt",
   "awsRegion":"us-east-1",
   "sourceIPAddress":"transcribe.streaming.amazonaws.com",
   "userAgent":"transcribe.streaming.amazonaws.com",
```

```
   "requestParameters":{
      "keyId":"mrk-de27f019178f4fbf86512ab03ba860be",
      "encryptionAlgorithm":"SYMMETRIC_DEFAULT",
      "encryptionContext":{
         "aws:us-east-1:transcribe:medical-scribe:session-
id":"1234abcd-12ab-34cd-56ef-123456SAMPLE"
      }
   },
   "responseElements":null,
   "requestID":"8b7fb865-48be-4e03-ac3d-e7bee3ba30a1",
   "eventID":"68b7a263-d410-4701-9e2b-20c196628966",
   "readOnly":true,
   "resources":[
      {
         "accountId":"123456789012",
         "type":"AWS::KMS::Key",
         "ARN":"arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
      }
   ],
   "eventType":"AwsApiCall",
   "managementEvent":true,
   "recipientAccountId":"123456789012",
   "eventCategory":"Management"
}
```

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	バッチ文字起こしジョブの PII リダクションではAGE、、、AWS_ACCESS_KEY 、AWS_SECRET_KEY 、CA_HEALTH_NUMBER CA_SOCIAL_INSURANCE_NUMBER 、、DATE_TIME 、、DRIVER_ID INTERNATIONAL_BANK_ACCOUNT_NUMBER 、IP_ADDRESS LICENSE_PLATE 、MAC_ADDRESS 、、、PASSPORT_NUMBER PASSWORDSWIFT_CODE URLUS_INDIVIDUAL_TAX_IDENTIFICATION_NUMBER 、、USERNAME、、、、、の各エンティティタイプがサポートされるようになりましたVEHICLE_IDENTIFICATION_NUMBER 。	2026 年 5 月 22 日
機能更新	バッチ文字起こしジョブの PII リダクションでは、フランス	2026 年 5 月 1 日

	語 (fr-FR)、カナダフランス語 ()、ドイツ語 (fr-CA)de-DE、スイスドイツ語 (de-CH)、イタリア語 (it-IT)、ポルトガル語 ()、ブラジルポルトガル語 (pt-PT) の追加言語がサポートされるようになりましたpt-BR。	
新機能	AWS HealthScribe は、臨床メモの概要の新しいテンプレートをサポートするようになりました。BIRP、SIRP、DAP、BH_SOAP、PH_SOAP。詳細については、「AWS HealthScribe 臨床ドキュメントファイル」を参照してください。	2025 年 5 月 30 日
新機能	AWS HealthScribe は、臨床メモの概要の GIRPP テンプレートをサポートするようになりました。詳細については、「AWS HealthScribe 臨床ドキュメントファイル」を参照してください。	2025 年 2 月 4 日
新機能	AWS HealthScribe は、医療会話をリアルタイムで文字起こしするためのストリーミングをサポートするようになりました。	2025 年 1月 29 日
セクション更新	8000 周波数の ar-SA 言語サポートの更新。	2025 年 1 月 16 日
セクション更新	8000 周波数の ar-SA 言語サポートの更新。	2025 年 1 月 16 日

セクション更新	AWS GovCloud (US) (米国西部、us-gov-west-1)、AWS GovCloud (US) (米国東部、us-gov-east-1)、またはアフリカ (ケープタウン、af-south-1) リージョンでサポートされていない非サポート言語コードを、サポート対象言語セクションに追加して更新しました。	2025 年 1 月 14 日
セクション更新	「audio_segments」という新しいフィールドを使用して Transcribe Batch JSON 出力を拡張しました。	2024 年 7 月 15 日
機能更新	ダイアライゼーションの最大話者数を 10 ではなく 30 に更新しました。	2024 年 5 月 10 日
セクション更新	通話の生成要約を更新し、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 IPA と SoundsLike を更新しました。	2024 年 4 月 30 日
機能更新	Amazon Transcribe Call Analytics が通話の生成要約をサポートす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 リファレンスが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を使用した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 秘匿化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 は、 AWS アカウント リソースのセキュリティレベルを高めるために AWS KMS 暗号化コンテキストをサポートするようになりました。	2021 年 9 月 10 日
新しい言語	Amazon Transcribe は、アフリカーン語、デンマーク語、標準中国語 (繁体字) 、タイ語、ニュージーランド英語、南アフリカ英語をサポートす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 は、バッチ文字起こしジョブのコール分析をサポートするようになりました。	2021 年 8 月 4 日
新機能	Amazon Transcribe では、バッチカスタム言語モデルでのカスタム語彙の使用がサポートされ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で部分的な結果の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 は、オーディオ文字起こしをストリーミングするための OGG/OPUS コーデックと FLAC コーデックをサポートす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関する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 は、バッチ文字起こしでのインターフェイス VPC エンドポイントのサポートを追加します。	2020 年 10 月 9 日
新機能	Amazon Transcribe は、ストリーミングでのチャネル識別のサポートを追加します。	2020 年 9 月 17 日
新機能	Amazon Transcribe では、バッチ文字起こしにおける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 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を追加します。	2020 年 8 月 19 日
新機能	Amazon Transcribe は、カスタム言語モデルのサポートを追加します。	2020 年 8 月 5 日
新機能	Amazon Transcribe は、ストリーミングのインターフェイス VPC エンドポイントのサポートを追加します。	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 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を追加します。	2020 年 2 月 26 日
新機能	Amazon Transcribe 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を追加します。	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、アラビア湾、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、独自の を指定 KMS key して文字起こし出力ファイルを暗号化できます。詳細については、[文字起こしのストリーミングをスタートする] API の「KMSKeyId の暗号化を出力する」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、口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オーディオ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) (us-west-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 では、モダンスタンダードアラビア語のサポートが追加されました。	2019 年 5 月 28 日
新機能	Amazon Transcribe は、数値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 に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 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を追加します。	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 Amazon Transcribe 認識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

最新の AWS 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。