

---

# Amazon Transcribe

## Developer Guide



## Amazon Transcribe: Developer Guide

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

## Table of Contents

What is Amazon Transcribe .....	1
Amazon Transcribe use cases .....	1
Use cases in action .....	2
Pricing .....	2
Supported languages .....	3
Feature summary .....	5
Guidelines and quotas .....	7
Supported AWS Regions .....	7
Guidelines .....	7
Quotas .....	7
How it works .....	10
Data input and output .....	10
Media formats .....	10
Sample rates .....	11
Output .....	11
Transcribing digits .....	11
Rules for transcribing numbers in English .....	12
Rules for transcribing numbers in German .....	13
Alternative transcriptions .....	14
Requesting alternative transcriptions .....	15
Job queuing .....	18
IAM policies for job queuing .....	19
Getting started .....	22
Signing up for an AWS account .....	22
Installing the AWS CLI and SDK .....	22
Creating an IAM user .....	23
Creating an Amazon S3 bucket .....	23
Transcribing with the AWS Management Console .....	23
Transcribing with the AWS CLI .....	31
Starting a new transcription job .....	32
Getting the status of a transcription job .....	33
Listing your transcription jobs .....	33
Deleting your transcription job .....	34
Transcribing with the AWS SDKs .....	34
Working with AWS SDKs .....	42
Transcribing with HTTP or WebSockets .....	42
Streaming transcriptions .....	44
Streaming and partial results .....	44
Partial-result stabilization .....	45
Partial-result stabilization example output .....	47
Setting up a streaming transcription .....	48
Event stream encoding .....	48
Data frames .....	50
Setting up an HTTP/2 stream .....	50
Setting up a WebSocket stream .....	54
Tagging resources .....	60
Tag-based access control .....	60
Tagging your Amazon Transcribe resources .....	61
Improving transcription accuracy .....	64
Custom vocabularies .....	64
Creating a custom vocabulary using a table .....	65
Creating a custom vocabulary using a list .....	69
Character sets .....	72
Custom language models .....	107

Creating a custom language model .....	108
Vocabulary filtering .....	118
Step 1: Creating a list of unwanted words .....	118
Step 2: Creating a vocabulary filter .....	119
Step 3: Filtering transcriptions .....	120
Filtering batch transcriptions .....	121
Filtering streaming transcriptions .....	125
Transcribing multi-channel audio .....	129
Transcribing multi-channel audio .....	129
Transcribing multi-channel audio streams .....	132
Transcribing multi-channel audio in an HTTP/2 stream .....	133
Transcribing multi-channel audio in a WebSocket stream .....	134
Multi-channel streaming output .....	134
Identifying languages .....	136
Batch language identification .....	136
Streaming language identification .....	141
Identifying speakers (diarization) .....	145
Identifying speakers in audio .....	145
Identifying speakers in real-time streams .....	148
HTTP/2 streaming .....	149
WebSocket streaming .....	149
Streaming transcription output .....	150
Analyzing call center audio .....	153
Call Analytics use cases .....	153
Considerations and additional information .....	154
Call Analytics insights .....	154
Call categorization .....	154
Call characteristics .....	155
Call summarization .....	155
Sensitive data redaction .....	156
Sentiment analysis .....	156
Creating categories .....	156
Rule criteria .....	161
Starting a Call Analytics job .....	163
Example output .....	168
Call categorization .....	168
Call characteristics .....	168
Call summarization .....	169
Sensitive data redaction .....	170
Sentiment analysis .....	170
Compiled output .....	172
Redacting transcripts .....	176
Redacting PII in your batch job .....	177
Redacting or identifying PII in a real-time stream .....	180
Example PII redaction and identification output .....	183
Example redacted batch output .....	183
Example redacted streaming output .....	185
Example PII identification output .....	186
Creating subtitles .....	188
Adding subtitles to your Amazon Transcribe video files .....	188
Code examples .....	191
Actions .....	192
Create a custom vocabulary .....	192
Delete a custom vocabulary .....	193
Delete a medical transcription job .....	194
Delete a transcription job .....	195
Get a custom vocabulary .....	196

Get a transcription job .....	197
List custom vocabularies .....	197
List medical transcription jobs .....	198
List transcription jobs .....	199
Start a medical transcription job .....	201
Start a transcription job .....	202
Update a custom vocabulary .....	203
Scenarios .....	204
Create and refine a custom vocabulary .....	204
Transcribe audio and get job data .....	210
Cross-service examples .....	211
Build an Amazon Transcribe app .....	211
Build an Amazon Transcribe streaming app .....	211
Convert text to speech and back to text .....	212
Security .....	213
Data protection .....	213
Encryption at rest .....	214
Encryption in transit .....	214
Key management .....	214
Opting out of using your data for service improvement .....	216
VPC endpoints (AWS PrivateLink) .....	217
Identity and access management .....	219
Audience .....	219
Authenticating with identities .....	219
Managing access using policies .....	221
How Amazon Transcribe works with IAM .....	223
Identity-based policy examples .....	226
Confused deputy prevention .....	230
Troubleshooting .....	231
Monitoring in Amazon Transcribe .....	233
Monitoring Amazon Transcribe with CloudTrail .....	233
Using Amazon EventBridge with Amazon Transcribe .....	235
CloudWatch metrics .....	239
Compliance validation .....	239
Resilience .....	239
Infrastructure security .....	240
Amazon Transcribe Medical .....	241
How it works .....	241
Speech input .....	242
Streaming transcription overview .....	243
Batch transcription overview .....	243
Transcribing numbers .....	244
Transcribing medical terms and measurements .....	245
Getting started .....	247
Set up an account .....	247
Getting started with the AWS Management Console (streaming) .....	248
Getting started with batch transcription .....	249
Streaming transcription .....	249
Event stream encoding .....	254
Using HTTP/2 streaming .....	255
Using WebSocket streaming .....	267
Transcribing a medical conversation .....	274
Transcribing an audio file .....	275
Transcribing a real-time stream .....	279
Identifying speakers .....	280
Transcribing multi-channel audio .....	287
Transcribing a medical dictation .....	293

Transcribing an audio file .....	294
Transcribing a streaming medical dictation .....	297
Creating and using medical custom vocabularies .....	299
Creating a text file for your medical custom vocabulary .....	299
Using a text file to create a medical custom vocabulary .....	302
Transcribing an audio file using a medical custom vocabulary .....	304
Transcribing a real-time stream using a medical custom vocabulary .....	305
Character sets for Amazon Transcribe Medical .....	307
Identifying PHI in a transcript .....	308
Identifying PHI in an audio file .....	309
Identifying PHI in a real-time stream .....	313
Generating alternative transcriptions .....	315
Guidelines and quotas .....	317
Supported AWS Regions .....	7
Guidelines .....	317
Quotas .....	317
VPC endpoints (AWS PrivateLink) .....	319
Considerations for Amazon Transcribe Medical VPC endpoints .....	319
Creating an interface VPC endpoint for Amazon Transcribe Medical .....	319
Creating a VPC endpoint policy for Amazon Transcribe Medical streaming .....	319
Document history .....	321
AWS glossary .....	329

# What is Amazon Transcribe?

Amazon Transcribe is an automatic speech recognition service that uses machine learning models to convert audio to text.

With Amazon Transcribe, you can ingest audio input, produce easy-to-read transcripts, improve accuracy with language customization, and filter content to ensure customer privacy. Practical use cases for Amazon Transcribe include transcribing and analyzing customer-agent calls and creating closed captions for videos.

With Amazon Transcribe, you can add speech-to-text capabilities to any application.

**Tip**

Information on the **Amazon Transcribe API** is located in the [API Reference](#).

See [What is Amazon Transcribe?](#) for a short video tour of this service.

## Amazon Transcribe use cases

Amazon Transcribe is a robust speech-to-text service that offers a diverse array of features, many of which can be combined between Amazon Transcribe and other AWS services.

- Gain insight into agent-customer calls using [Call Analytics \(p. 153\)](#). This feature automatically analyzes 11 different criteria without any customization on your part. For each speaker, you get sentiment data, talk time, non-talk time, loudness, interruptions, and talk speed. Call summarization, call categorization, and turn-by-turn output are provided for the whole call.

We've also launched two new analytics options for call center audio: [post-call analytics](#) (designed for audio files located in an Amazon S3 bucket) and [live-call analytics](#) (designed for live audio streams).

- Get a summary of customer-agent interactions with [call summarization \(p. 155\)](#). Get an at-a-glance summary of issues, action items, and outcomes for every call.
- Teach Amazon Transcribe industry-specific terms, unique spelling, acronyms, and any words that are not being rendered correctly in your transcription results using [custom vocabularies \(p. 64\)](#). Providing Amazon Transcribe with custom vocabularies can improve the accuracy of your transcription output. See also: [Custom language models \(p. 107\)](#).
- Create [subtitles \(p. 188\)](#) for your video files. You can also use [content redaction \(p. 176\)](#) (only in US English) and [vocabulary filtering \(p. 118\)](#) when generating subtitles to ensure your content is audience-appropriate. Note that filtered or redacted content shows as white space, \*\*\*, or [PII] in your transcript and subtitle files, but **the audio itself is not altered**.
- Redact personally identifiable information (PII), such as social security numbers, from your transcripts using standard [content redaction \(p. 176\)](#) or Call Analytics [sensitive data redaction \(p. 156\)](#). Call Analytics can also [redact your audio](#) by replacing spoken PII with silence.
- Identify individual speakers in an audio clip using [speaker diarization \(p. 145\)](#). When you activate speaker diarization, Amazon Transcribe attaches a unique attribute to each speaker in your transcription output.
- Remove proprietary terms from your transcript using [vocabulary filtering \(p. 118\)](#). For example, you can mask the name of a new product in a pre-launch stakeholder meeting. Vocabulary filtering can also be used to mask profane, offensive, or audience-inappropriate terms.
- Using multi-channel audio, you can have Amazon Transcribe produce a separate transcript for each channel, or have all channels transcribed in one output file. See [Transcribing multi-channel audio \(p. 129\)](#).

- If your audio is not in a language you speak, let Amazon Transcribe identify the language for you using [language identification \(p. 136\)](#). You can then use [Amazon Translate](#) to translate your transcript, and have [Amazon Polly](#) read your transcript back to you.
- Improve streaming transcription accuracy with [partial result stabilization \(p. 45\)](#), which can also be used to [adjust the latency of your transcript](#).

## Use cases in action

Here are some diverse examples of how individuals and organizations are using Amazon Transcribe.

- Live transcriptions of F1 races using [Amazon Transcribe](#)
- Generate high-quality meeting notes using [Amazon Transcribe](#) and [Amazon Comprehend](#)
- Boost transcription accuracy of class lectures with custom language models for [Amazon Transcribe](#)
- Make your audio and video files searchable using [Amazon Transcribe](#) and [Amazon Kendra](#)
- Perform medical transcription analysis in real-time with [AWS AI services](#) and [Twilio Media Streams](#)

## Pricing

Amazon Transcribe is a pay-as-you-go service; pricing is based on seconds of transcribed audio, billed on a monthly basis. For more information on cost, including cost-breakdown examples for various AWS Regions, see [Amazon Transcribe Pricing](#).

# Supported languages

The languages supported by Amazon Transcribe are listed in the following table; also listed are the features that are language-specific. Please verify that the feature you wish to use is supported for the language in your media before proceeding with your transcription. To view the complete list of Amazon Transcribe features, refer to [Amazon Transcribe features \(p. 5\)](#).

In the following table, 'batch' refers to transcribing a media file located in an Amazon S3 bucket and 'streaming' refers to transcribing streamed media in real time.

**Supported languages and language-specific features**

Language	Language Code	Data input (p. 1)	Transcriber digits (p. 1)	Acronyms	Custom language models (p. 1)	Redacting transcripts	Call Analytics (p. 1)	
Afrikaans ( <a href="#">p. 77</a> ZA)		batch	no	batch	no	no	no	
Arabic ( <a href="#">p. 75</a> )-AE Gulf		batch	no	no	no	no	batch	
Arabic ( <a href="#">p. 75</a> )-SA Modern Standard		batch	no	no	no	no	no	
Chinese, Simplified ( <a href="#">p. 76</a> )	zh-CN	batch, streaming	no	no	no	no	batch	
Chinese, Traditional ( <a href="#">p. 77</a> )	zh-TW	batch	no	no	no	no	no	
Danish ( <a href="#">p. 78</a> )-DK		batch	no	batch	no	no	no	
Dutch ( <a href="#">p. 79</a> )-NL		batch	no	batch	no	no	no	
English ( <a href="#">p. 80</a> )-AU Australian		batch, streaming	batch, streaming	batch, streaming	batch	no	batch	
English ( <a href="#">p. 80</a> )-GB British		batch, streaming	batch, streaming	batch, streaming	batch, streaming	no	batch	
English ( <a href="#">p. 80</a> )-IN Indian		batch	batch	batch	no	no	batch	
English ( <a href="#">p. 80</a> )-IE Irish		batch	batch	batch	no	no	batch	
English ( <a href="#">p. 80</a> )-NZ New Zealand		batch	batch	batch	no	no	no	
English ( <a href="#">p. 80</a> )-AB Scottish		batch	batch	batch	no	no	batch	
English ( <a href="#">p. 80</a> )-ZA South African		batch	batch	batch	no	no	no	

Language	Language Code	Data input (p. 1)	Transcriber digits (p. 1)	Acronyms	Custom language models (p. 1)	Redacting transcripts	Call Analytics (p. 1)	
English (p. 80) US	en-US	batch, streaming	batch, streaming	batch, streaming	batch, streaming	batch, streaming	batch	
English (p. 80) Welsh	en-WL	batch	batch	batch	no	no	batch	
French (p. 81) FR	fr-FR	batch, streaming	no	batch, streaming	no	no	batch	
French (p. 81) Canadian	fr-CA	batch, streaming	no	batch, streaming	no	no	batch	
Farsi (p. 81) fa-IR	fa-IR	batch	no	no	no	no	no	
German (p. 84) DE	de-DE	batch, streaming	batch, streaming	batch, streaming	no	no	batch	
German (p. 84) CH	de-CH	batch	batch	batch	no	no	batch	
Hebrew (p. 86) IL	he-IL	batch	no	no	no	no	no	
Hindi (p. 87) hi-IN	hi-IN	batch	no	batch	batch	no	batch	
Indonesian (id-ID)	id-ID	batch	no	batch	no	no	no	
Italian (p. 90) IT	it-IT	batch, streaming	no	batch, streaming	no	no	batch	
Japanese (ja-JP)	ja-JP	batch, streaming	no	no	no	no	batch	
Korean (p. 92) ko-KR	ko-KR	batch, streaming	no	no	no	no	batch	
Malay (p. 93) ms-MY	ms-MY	batch	no	batch	no	no	no	
Portuguese (pt-PT)	pt-PT	batch	no	batch	no	no	batch	
Portuguese (pt-BR), Brazilian	pt-BR	batch, streaming	no	batch, streaming	no	no	batch	
Russian (p. 96) RU	ru-RU	batch	no	no	no	no	no	
Spanish (p. 98) ES	es-ES	batch	no	batch	no	no	batch	
Spanish (p. 98) US	es-US	batch, streaming	no	batch, streaming	batch	no	batch	
Tamil (p. 99) ta-IN	ta-IN	batch	no	no	no	no	no	
Telugu (p. 101) JN	te-JN	batch	no	no	no	no	no	
Thai (p. 103) th-TH	th-TH	batch	no	batch	no	no	no	
Turkish (p. 105) TR	tr-TR	batch	no	batch	no	no	no	

# Amazon Transcribe features

Amazon Transcribe offers several services that each support a different subset of features.

In the following table, 'batch' refers to transcribing a file that is located in an Amazon S3 bucket and 'streaming' refers to transcribing media in real time.

Feature	Amazon Transcribe	Amazon Transcribe Medical (p. 241) <sup>1</sup>	Amazon Transcribe Call Analytics (p. 153)
<i>Configuration options</i>			
Identifying languages (p. 136)	batch, streaming	N/A	batch
Transcribing multi-channel audio (p. 129)	batch, streaming	batch, streaming	batch
Job queuing (p. 18)	batch	No	batch
Speaker diarization (p. 145)	batch, streaming	batch, streaming	batch
Transcribing digits (p. 11) <sup>2</sup>	batch, streaming	batch, streaming	batch
<i>Conversation analytics</i>			
Action items (p. 155) <sup>2</sup>	No	No	batch
Call outcomes (p. 155) <sup>2</sup>	No	No	batch
Issue detection (p. 155) <sup>2</sup>	No	No	batch
Non-talk time (silence) (p. 155) <sup>2</sup>	No	No	batch
Speaker interruptions (p. 155) <sup>2</sup>	No	No	batch
Speaker sentiment (p. 156) <sup>2</sup>	No	No	batch
Talk speed (p. 155) <sup>2</sup>	No	No	batch
<i>Language customization</i>			
Custom language models (p. 107) <sup>2</sup>	batch, streaming	No	batch
Custom vocabularies (p. 64)	batch, streaming	batch, streaming	batch
<i>Resource organization</i>			

Feature	Amazon Transcribe	Amazon Transcribe Medical (p. 241) <sup>1</sup>	Amazon Transcribe Call Analytics (p. 153)
Tagging resources (p. 60)	batch	batch, streaming	No
<i>Sensitive data</i>			
Identifying personal health information (p. 308) <sup>2</sup>	No	batch, streaming	No
Identifying personally identifiable information (p. 180) <sup>2</sup>	streaming	No	No
Redacting audio (p. 156) <sup>2</sup>	No	No	batch
Redacting transcripts (p. 176) <sup>2</sup>	batch, streaming	No	batch
Vocabulary filtering (p. 118)	batch, streaming	batch, streaming	batch
<i>Video</i>			
Subtitles (p. 188)	batch	No	No

<sup>1</sup> Amazon Transcribe Medical is only available in US English.

<sup>2</sup> This feature is not available for all languages; review the [Supported languages \(p. 3\)](#) table for more details.

# Guidelines and quotas

Amazon Transcribe has several guidelines to follow in order to achieve optimal results. There are also quotas that may impact your transcriptions; some of these can be increased. Refer to the following sections for details.

## Supported AWS Regions

For a list of AWS Regions where Amazon Transcribe is available, see [Amazon Transcribe Endpoints and Quotas](#) in the *AWS General Reference*.

## Guidelines

For best results:

- Use a lossless format, such as FLAC or WAV with PCM 16-bit encoding.
- Use a sample rate of 8,000 Hz for low-fidelity audio and 16,000–48,000 Hz for high-fidelity audio.

If you don't need to process all of your transcription jobs concurrently, use [Job queuing \(p. 18\)](#). This enables Amazon Transcribe to keep track of your transcription jobs and process them when slots are available. You can request an increase to the job queue bandwidth ratio to run more transcription jobs. The quota for the transcription jobs in your job queue is the product of the number of transcription jobs you can run concurrently and the bandwidth ratio. For example, if you have a bandwidth ratio of 5 and a quota of 100 for the number of transcription jobs you can run concurrently then you can have 500 transcription jobs in your job queue.

### Note

Amazon Transcribe may temporarily store your content to continuously improve the quality of its analysis models. See the [Amazon Transcribe FAQ](#) to learn more. To request the deletion of content that may have been stored by Amazon Transcribe, open a case with [AWS Support](#).

## Quotas

The following quotas **cannot** be changed:

Description	Quota
Audio file length	4:00:00 (four) hours (14,400 seconds)
Audio stream duration	4:00:00 (four) hours (14,400 seconds)
Audio file size	2 GB
Audio file size (call analytics)	500 MB
Size of a custom vocabulary	50 KB
Length of a custom vocabulary phrase	256 characters

Description	Quota
Size of a vocabulary filter	50 KB
Number of vocabulary filters	100
Number of channels for channel identification	2
Number of days job records are retained	90
Minimum audio file duration	500 milliseconds (ms)

You can request a quota increase for the following resources:

Resource	Default
Number of concurrent batch transcription jobs	250
Number of concurrent batch transcription jobs (Call Analytics)	100
Job queue bandwidth ratio	0.9
Number of concurrent HTTP/2 requests	25
Number of concurrent WebSocket requests	25
Total number of vocabularies per account	100
Number of pending vocabularies	10
Number of concurrently training custom language models	3
Total number of custom language models per account	10
Number of categories per account (Call Analytics)	200
Number of rules per category (Call Analytics)	20

The following operations limits can also be increased upon request:

Operation	Transactions per second
<a href="#">StartTranscriptionJob</a>	25

Operation	Transactions per second
<a href="#">StartStreamTranscription</a>	25
<a href="#">GetTranscriptionJob</a>	30
<a href="#">DeleteTranscriptionJob</a>	5
<a href="#">ListTranscriptionJobs</a>	5
<a href="#">CreateVocabulary</a>	10
<a href="#">UpdateVocabulary</a>	10
<a href="#">DeleteVocabulary</a>	5
<a href="#">GetVocabulary</a>	20
<a href="#">ListVocabularies</a>	5
<a href="#">StartCallAnalyticsJob</a>	10
<a href="#">GetCallAnalyticsJob</a>	20
<a href="#">ListCallAnalyticsJobs</a>	5
<a href="#">DeleteCallAnalyticsJob</a>	5
<a href="#">CreateCallAnalyticsCategory</a>	10
<a href="#">UpdateCallAnalyticsCategory</a>	10
<a href="#">DeleteCallAnalyticsCategory</a>	5
<a href="#">GetCallAnalyticsCategory</a>	20
<a href="#">ListCallAnalyticsCategories</a>	5

**Note**

For information about requesting a quota increase, see [AWS service quotas](#) in the *AWS General Reference*.

# How Amazon Transcribe works

Amazon Transcribe analyzes media that contains speech and uses advanced machine learning techniques to transcribe the voice data into text. You can then use the transcription as you would any text document.

Transcription methods can be separated into two main categories:

- Batch transcription jobs: Transcribing media that have been uploaded into an Amazon S3 bucket.
- Streaming transcriptions: Transcribing media streams in real time.

Batch transcription jobs can be created with the [AWS CLI \(p. 31\)](#), [AWS Management Console \(p. 23\)](#), and various [AWS SDKs \(p. 34\)](#).

Streaming transcriptions can be created with the [AWS Management Console \(p. 23\)](#), [HTTP/2 \(p. 50\)](#), [WebSockets \(p. 54\)](#), and various [AWS SDKs \(p. 34\)](#).

For information on supported file containers and formats for both batch and streaming options, see [Data input and output \(p. 10\)](#).

## API operations to get you started

Batch: [GetTranscriptionJob](#), [ListTranscriptionJobs](#), [StartTranscriptionJob](#)

Streaming: [StartStreamTranscription](#)

## Data input and output

Amazon Transcribe takes audio data, either a media file in an Amazon S3 bucket or a media stream, and converts it to text data.

If you're working with media files stored in an Amazon S3 bucket, you're performing batch transcription jobs; if you're working with media streams, you're performing streaming transcriptions. These two processes have different rules and requirements.

## Media formats

Supported media types differ between batch transcriptions and streaming transcriptions, though lossless formats are recommended for both. See the following table for details:

	<b>Batch</b>	<b>Streaming</b>
Supported formats	<ul style="list-style-type: none"><li>• AMR</li><li>• FLAC</li><li>• MP3</li><li>• MP4</li><li>• Ogg</li></ul>	<ul style="list-style-type: none"><li>• FLAC</li><li>• OPUS-encoded audio in an Ogg container</li><li>• PCM signed 16-bit little-endian audio (note that this does <b>not</b> include WAV)</li></ul>

	<b>Batch</b>	<b>Streaming</b>
	<ul style="list-style-type: none"><li>• WebM</li><li>• WAV</li></ul>	
Recommended formats	<ul style="list-style-type: none"><li>• FLAC</li><li>• WAV with PCM 16-bit encoding</li></ul>	<ul style="list-style-type: none"><li>• FLAC</li><li>• PCM encoding</li></ul>

## Sample rates

With batch transcription jobs, you can choose to provide a sample rate, though this parameter is optional. If you do include it in your request, make sure the value you provide matches the actual sample rate in your audio; if you provide a sample rate that doesn't match your audio, your job may fail.

With streaming transcriptions, you must include a sample rate in your request. As with batch transcription jobs, make sure the value you provide matches the actual sample rate in your audio.

Sample rates for low fidelity audio, such as telephone recordings, typically use 8,000 Hz. For high fidelity audio, Amazon Transcribe supports values between 16,000 Hz and 48,000 Hz.

## Output

All batch transcripts are stored in Amazon S3 buckets. You can either choose to save your transcript in your own Amazon S3 bucket, or have Amazon Transcribe use a secure default bucket. To learn more about creating and using Amazon S3 buckets, see [Working with buckets](#).

If you want your transcript stored in an Amazon S3 bucket that you own, specify the bucket's URI in your transcription request. Make sure you give Amazon Transcribe write permissions for this bucket prior to starting your batch transcription job. If you specify your own bucket, your transcript remains in that bucket until you remove it.

If you don't specify an Amazon S3 bucket, Amazon Transcribe uses a secure service-managed bucket and provides you with a temporary URI that you can use to download your transcript. Note that temporary URLs are valid for 15 minutes. If you get an `AccessDenied` error when using the provided URI, run a `getTranscriptionJob` request to get a new temporary URI for your transcript.

If you opt for a default bucket, your transcript is deleted when your job expires (90 days). So if you want to keep your transcript, you must download it before the expiration date.

Streaming transcripts are returned via the method you're using for your stream (WebSocket or HTTP/2).

## Transcribing numbers and punctuation

Amazon Transcribe treats numbers differently depending on the language and the context in which they're used. For most languages, numbers are transcribed into their word forms. English and German language variants support digit transcription, as outlined in [Rules for transcribing numbers in English \(p. 12\)](#) and [Rules for transcribing numbers in German \(p. 13\)](#).

Amazon Transcribe automatically adds punctuation to all supported languages, and capitalizes words appropriately for languages that use case distinction in their writing systems.

For a list of supported languages, see [Supported languages and language-specific features \(p. 3\)](#).

## Rules for transcribing numbers in English

For all English language variants, numbers are transcribed according to the following rules.

Rule	Examples (input audio → output text)
Convert cardinal numbers greater than ten to numbers.	<ul style="list-style-type: none"> <li>"Fifty five" → 55</li> <li>"a hundred" → 100</li> <li>"One thousand and thirty one" → 1031</li> <li>"One hundred twenty-three million four hundred fifty six thousand seven hundred eight nine" → 123,456,789</li> </ul>
Convert cardinal numbers followed by "million" or "billion" to numerals followed by a word when "million" or "billion" is not followed by a number.	<ul style="list-style-type: none"> <li>"one hundred million" → 100 million</li> <li>"one billion" → 1 billion</li> <li>"two point three million" → 2.3 million</li> </ul>
Convert ordinal numbers greater than ten to numbers.	<ul style="list-style-type: none"> <li>"Forty third" → 43rd</li> <li>"twenty sixth avenue" → 26th avenue</li> </ul>
Convert fractions to their numeric format.	<ul style="list-style-type: none"> <li>"a quarter" → 1/4</li> <li>"three sixteenths" → 3/16</li> <li>"a half" → 1/2</li> <li>"a hundredth" → 1/100</li> </ul>
Convert numbers less than ten to digits if there are more than one in a row.	<ul style="list-style-type: none"> <li>"three four five" → 345</li> <li>"My phone number is four two five five five five one two one two" → My phone number is 4255551212</li> </ul>
The words "dot" or "point" are displayed as a decimal.	<ul style="list-style-type: none"> <li>"three hundred and three dot five" → 303.5</li> <li>"three point twenty three" → 3.23</li> <li>"zero point four" → 0.4</li> <li>"point three" → 0.3</li> </ul>
Convert the word "percent" after a number to a percent symbol (%).	<ul style="list-style-type: none"> <li>"twenty three percent" → 23%</li> <li>"twenty three point four five percent" → 23.45%</li> </ul>
Convert the words "dollar," "U S dollar," "Australian dollar," "AUD," or "USD" after a number to a dollar sign (\$) before the number.	<ul style="list-style-type: none"> <li>"one dollar and fifteen cents" → \$1.15</li> <li>"twenty three USD" → \$23</li> <li>"twenty three Australian dollars" → \$23</li> </ul>
Convert the words "pounds," "British pounds," or "GDB" after a number to pound sign (£) before the number.	<ul style="list-style-type: none"> <li>"twenty three pounds" → £23</li> <li>"I have two thousand pounds" → I have £2,000</li> <li>"five pounds thirty three pence" → £5.33</li> </ul>
Convert the words "rupees," "Indian rupees," or "INR" after a number to rupee sign (₹) before the number.	<ul style="list-style-type: none"> <li>"twenty three rupees" → ₹23</li> <li>"fifty rupees thirty paise" → ₹50.30</li> </ul>
Convert times to numbers.	<ul style="list-style-type: none"> <li>"seven a m eastern standard time" → 7 a.m. eastern standard time</li> <li>"twelve thirty p m" → 12:30 p.m.</li> </ul>

Rule	Examples (input audio → output text)
Years expressed as two numbers are represented as four digits; this is only valid for years in the 20th, 21st, and 22nd centuries.	<ul style="list-style-type: none"> <li>"nineteen sixty two" → 1962</li> <li>"the year is twenty twelve" → the year is 2012</li> <li>"twenty nineteen" → 2019</li> <li>"twenty one thirty" → 2130</li> </ul>
Convert dates to numbers.	<ul style="list-style-type: none"> <li>"May fifth twenty twelve" → May 5th 2012</li> <li>"May five twenty twelve" → May 5 2012</li> <li>"five May twenty twelve" → 5 May 2012</li> </ul>
Separate spans of numbers by the word "to".	<ul style="list-style-type: none"> <li>"twenty three to thirty seven" → 23 to 37</li> </ul>

## Rules for transcribing numbers in German

For all German language variants, numbers are transcribed according to the following rules.

Rule	Examples (input audio → output text)
Convert cardinal numbers greater than ten to numbers.	<ul style="list-style-type: none"> <li>"fünfundfünfzig" → 55</li> <li>"vier tausend sechs hundert einundachtzig" → 4681</li> <li>"eine Sache" → "eine Sache"</li> </ul>
Convert cardinal numbers followed by "million" or "billion" to numerals followed by a word when "million" or "billion" is not followed by a number.	<ul style="list-style-type: none"> <li>"zehn Millionen Menschen" → 10 Millionen Menschen</li> <li>"zehn Millionen fünf hundert tausend" → 10.500.000</li> </ul>
Convert ordinal numbers greater than ten to numbers.	<ul style="list-style-type: none"> <li>"dreiundzwanzigste" → 23</li> <li>"vierzigster" → 40</li> <li>"ich war Erster" → "ich war Erster"</li> </ul>
Fractions are not converted into a numeric format.	<ul style="list-style-type: none"> <li>"ein Drittel" → "ein Drittel"</li> </ul>
Convert numbers less than ten to digits if there are more than one in a row.	<ul style="list-style-type: none"> <li>"eins zwei drei" → 123</li> <li>"plus vier neun zwei vier eins" → +49241</li> </ul>
Decimals are indicated by ",".	<ul style="list-style-type: none"> <li>"zweiundzwanzig komma drei" → 22,3</li> </ul>
Convert the word "percent" after a number to a percent symbol (%).	<ul style="list-style-type: none"> <li>"fünf Prozent Hürde" → 5% Hürde</li> <li>"dreiundzwanzig komma vier Prozent" → 23,4%</li> </ul>
Convert the words "Euro" to a euro sign.	<ul style="list-style-type: none"> <li>"ein euro" → 1 €</li> <li>"ein Euro vierzig" → 1,40 €</li> <li>"ein Euro vierzig Cent" → 1,40 €</li> </ul>
Convert times to numbers.	<ul style="list-style-type: none"> <li>"vierzehn Uhr fünfzehn" → 14:15 Uhr</li> </ul>
Convert dates to numbers.	<ul style="list-style-type: none"> <li>"dritter Dezember neunzehn hundert sechundfünfzig" → 3. Dezember 1956</li> </ul>
Display slashes and dashes.	<ul style="list-style-type: none"> <li>"55 Schrägstrich 13" → 55/13</li> </ul>

Rule	Examples (input audio → output text)
Display numbered paragraphs.	<ul style="list-style-type: none"> <li>• "55 Strich 13" → 55-13</li> <li>• "Paragraf 17" → § 17</li> </ul>

## Alternative transcriptions

When Amazon Transcribe transcribes audio, it creates different versions of the same transcript and assigns a confidence score to each version. In a typical transcription, you only get the version with the highest confidence score.

If you enable alternative transcriptions, Amazon Transcribe returns other versions of your transcript that have lower confidence levels. You can choose to have up to ten alternative transcriptions returned. If you specify a greater number of alternatives than what Amazon Transcribe identifies, only the actual number of alternatives is returned.

All alternatives are located in the same transcription output file and are presented at the segment level. Segments are natural pauses in speech, such as a change in speaker or a pause in the audio.

Alternative transcriptions are only available for batch transcriptions.

Your transcription output is structured as follows. The ellipses (...) in the code examples indicate where content has been removed for brevity.

1. A complete final transcription for a given segment.

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

2. A confidence score for each word in the preceding `transcript` section.

```
"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. Your alternative transcriptions, located in the segments portion of your transcription output. The alternatives for each segment are ordered by descending confidence score.

```
"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. A status at the end of your transcription output.

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

## Requesting alternative transcriptions

You can request alternative transcriptions using the **AWS Management Console**, **AWS CLI**, or **AWS SDK**; see the following for examples:

### AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Transcription jobs**, then select the **Create job** button (top right). This opens the **Specify job details** page.

## Specify job details [Info](#)

### Job settings

Name  The name can be up to 200 characters long. Valid characters are a-z, A-Z, 0-9, . (period), \_ (underscore), and – (hyphen).

Model type [Info](#)  
Choose the type of model to use for the transcription job.

General model  
To use a model that is not specialized for a particular use case, choose this option. Configuration options vary between languages.

Custom language model  
To use a model that you trained for your specific use case, choose this option. This model has fewer configuration options than the general model.

Language settings  
You can transcribe your audio file in a language that you specify or have Amazon Transcribe identify and transcribe it in the predominant language.

Specific language [Info](#)  
If you know the language spoken in your source audio, choose this option to get the most accurate results. The options available for additional processing vary between languages.

Automatic language identification [Info](#)  
If you don't know the language spoken in your audio files, choose this option. You have access to fewer options for additional processing than if you choose Specific language.

Language  
Choose the language of the input audio.  
 ▾

▶ Additional settings

- Fill in any fields you wish to include on the **Specify job details** page, then click the **Next** button. This takes you to the **Configure job - optional** page.

Select **Alternative results** and specify the maximum number of alternative transcription result you want in your transcript.

## Configure job - *optional* Info

### Audio settings

#### Audio identification Info

Choose to split multi-channel audio into separate channels for transcription, or identify speakers in the input audio.

#### Alternative results Info

Enable to view more transcription results

#### Maximum alternatives

Provide the number of alternative transcription to provide in the text output.

3

The maximum number of alternative results is 10.

- Click the **Create job** button to run your transcription job.

## AWS CLI

This example uses the [start-transcription-job](#) command and `ShowAlternatives` parameter. For more information, see [StartTranscriptionJob](#) and [ShowAlternatives](#).

Note that if you include `ShowAlternatives=true` in your request, you must also include `MaxAlternatives`.

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

Here's another example using the [start-transcription-job](#) command, and a request body that identifies the language.

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

The file `my-first-alt-transcription-job.json` contains the following request body.

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

```
}
```

## AWS SDK for Python (Boto3)

The following example uses the AWS SDK for Python (Boto3) to request alternative transcriptions by using the `ShowAlternatives` argument for the [start\\_transcription\\_job](#) method. For more information, see [StartTranscriptionJob](#) and [ShowAlternatives](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

Note that if you include '`ShowAlternatives':True`' in your request, you must also include `MaxAlternatives`.

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

## Job queuing

When you send transcription jobs to Amazon Transcribe, there is a limit to the total number of jobs that can run at one time. By default, there are 250 slots for jobs. When the limit is reached, you must wait until one or more jobs have finished and freed up a slot before you can send your next job.

To queue jobs so that they run as soon as a slot becomes available, you can use *job queuing*. Job queuing creates a queue on your behalf that contains your jobs. When a slot is available, Amazon Transcribe takes the next job from the queue and immediately starts processing it. To allow resources for new jobs to be submitted and processed, Amazon Transcribe uses at most 90 percent of your slots to process jobs in the queue.

You can turn on job queuing with the AWS Management Console, or you can set the `AllowDeferredExecution` field of the `JobExecutionSettings` parameter to `true` when you call the [StartTranscriptionJob](#) API.

When you submit a job with job queuing turned on, one of the following things happens.

- If slots are available, the job is processed immediately.
- If no slots are available, the job is sent into a queue. When slots become available, jobs are removed from the queue in FIFO order (first in, first out).

You can see the progress of a queued job using the console or by using the [GetTranscriptionJob](#) API. When a job is queued, the `Status` field of the `TranscriptionJob` object returned by the [StartTranscriptionJob](#) API is set to `QUEUED`. The status changes to `IN_PROGRESS` when Amazon Transcribe starts processing the audio, and then changes to either `COMPLETED` or `FAILED` when processing is finished. You can use the `TranscriptionJobName` field with the [GetTranscriptionJob](#) API to monitor the status of a job.

You can submit up to 10,000 jobs to the queue. If you exceed 10,000 jobs, you get a `LimitExceededConcurrentJobException` exception.

## IAM policies for job queuing

To use job queuing, you must provide Amazon Transcribe with a data access role that permits access to the audio you want transcribed. You can choose the data access role using the AWS Management Console, or you use the `DataAccessRoleArn` field of the `JobExecutionSettings` parameter of the [StartTranscriptionJob](#) API to specify the role to use.

The role policies that you use depend on where you are storing your input files, where you are storing your output files, and whether you are encrypting the output with an AWS KMS key. The IAM policies in this section are required for the role. The policies enable Amazon Transcribe to work on your behalf, allow access to the input and output locations for your jobs, and enable Amazon Transcribe to use a KMS key to encrypt your transcriptions.

### Trust policy

The data access role that you use for transcription must have a trust policy that enables Amazon Transcribe to assume the role. Use the following trust policy.

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

### Input bucket policy

The following IAM policy gives the data access role permission to read files from your input bucket.

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

```
        "s3:GetObject",
        "s3>ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::input-bucket-name",
        "arn:aws:s3:::input-bucket-name/*"
    ]
}
}
```

## Output bucket policy

The following IAM policy gives the data access role permission to write files to your output bucket.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::output-bucket-name/*"
        ]
    }
}
```

## KMS key policy for input buckets

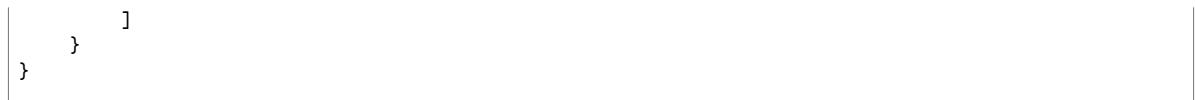
If you have encrypted your input files, the data access role needs permission to use the KMS key to decrypt the files. The following policy provides that permission.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt"
        ],
        "Resource": [
            "arn:aws:kms:us-west-2:111122223333:key/input-bucket-key-id"
        ]
    }
}
```

## KMS key policy for output buckets

To encrypt your output transcriptions, the data access role needs permission to use the KMS key. The following policy provides that permission.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey*"
        ],
        "Resource": [
            "arn:aws:kms:us-west-2:111122223333:key/output-bucket-key-id"
        ]
    }
}
```



# Getting started with Amazon Transcribe

Before you can create transcriptions, you have a few prerequisites:

- [Sign up for an AWS account \(p. 22\)](#)
- [Install the AWS CLI and SDK \(p. 22\)](#) (if you're using the AWS Management Console for your transcriptions, you can skip this step)
- [Create an IAM user with administrator permissions \(p. 23\)](#)
- [Set up an Amazon S3 bucket \(p. 23\)](#)

Once you complete these prerequisites, you're ready to transcribe. Select your preferred transcription method from the following list to get started.

- [AWS CLI \(p. 31\)](#)
- [AWS Management Console \(p. 23\)](#)
- [AWS SDK \(p. 34\)](#)
- [HTTP \(p. 42\)](#)
- [WebSockets \(p. 42\)](#)

Because streaming using HTTP/2 and WebSockets is more complicated than the other transcription methods, we advise reviewing the [Setting up a streaming transcription \(p. 48\)](#) section before getting started with these methods.

## Signing up for an AWS account

You can sign up for either a [free tier](#) account or a [paid account](#). Both options give you access to all AWS services. The free tier has a trial period during which you can explore AWS services and get a feel for your usage. Once your trial period expires, you can migrate to a paid account. Fees are accrued on a pay-as-you-use basis; see [Amazon Transcribe Pricing](#) for details.

**Tip**

When setting up your account, make note of your AWS account ID because you need it to create an IAM user or group.

## Installing the AWS CLI and SDK

To use the Amazon Transcribe API, you must first install the AWS CLI. The current AWS CLI is version 2. You can find installation instructions for [Linux](#), [Mac](#), [Windows](#), and [Docker](#) in the [AWS Command Line Interface User Guide](#).

Once you have the AWS CLI installed, you need to [configure](#) it for your security credentials and AWS Region.

If you want to use Amazon Transcribe with an SDK, click on your preferred language for installation instructions:

- .NET
- C++
- Go
- Java V2
- JavaScript
- PHP V3
- [AWS SDK for Python \(Boto3\)](#) — batch transcriptions only
- [Python](#) — streaming transcriptions only
- [Ruby V3](#)

## Creating an IAM user

IAM, or Identity and Access Management, is a means of controlling which users can access which resources. All AWS services use IAM credentials and it is best practice to use an IAM admin user to access your resources. Using your AWS account root user credentials to access resources is **not** recommended. To learn more about IAM, see [What is IAM?](#)

You can create an IAM admin user with the AWS Management Console or AWS CLI. For instructions, see [Creating your first IAM admin user and user group](#). For information on signing in to your account using IAM user credentials, refer to [How IAM users sign in to your AWS account](#).

For AWS CLI-specific IAM instructions, see [Using an IAM role in the AWS CLI](#).

## Creating an Amazon S3 bucket

Amazon S3 is a secure object storage service. Amazon S3 stores your files (called *objects*) in containers (called *buckets*).

To run a batch transcription, you must first upload your media files into an Amazon S3 bucket. If you don't specify an Amazon S3 bucket for your transcription output, Amazon Transcribe puts your transcript in a temporary AWS-managed Amazon S3 bucket. Transcription output in AWS-managed buckets is automatically deleted after 90 days.

Learn how to [Create your first S3 bucket](#) and [Upload an object to your bucket](#).

## Transcribing with the AWS Management Console

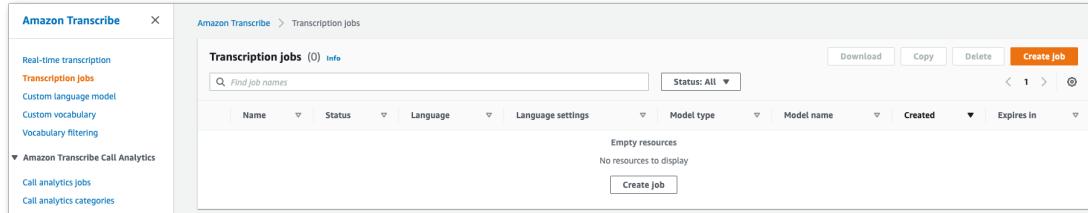
Prior to starting a batch transcription, you must first upload your media file to an Amazon S3 bucket. For streaming transcriptions using the AWS Management Console, you must use your computer microphone and transcribe real-time speech.

To view supported media formats and other media requirements and constraints, see [Data input and output \(p. 10\)](#).

### Batch transcriptions

First make sure that you've uploaded the media file you want to transcribe into an Amazon S3 bucket. If you're unsure how to do this, refer to the Amazon S3 User Guide: [Upload an object to your bucket](#).

- From the [AWS Management Console](#), select **Transcription jobs** in the left navigation pane. This takes you to a list of your transcription jobs.



Click the **Create job** button.

- Complete the fields on the **Specify job details** page.

## Specify job details Info

---

**Job settings**

**Name**

The name can be up to 200 characters long. Valid characters are a-z, A-Z, 0-9, . (period), \_ (underscore), and - (hyphen).

**Model type Info**

Choose the type of model to use for the transcription job.

**General model**

To use a model that is not specialized for a particular use case, choose this option. Configuration options vary between languages.

**Custom language model**

To use a model that you trained for your specific use case, choose this option. This model has fewer configuration options than the general model.

**Language settings**

You can transcribe your audio file in a language that you specify or have Amazon Transcribe identify and transcribe it in the predominant language.

**Specific language Info**

If you know the language spoken in your source audio, choose this option to get the most accurate results. The options available for additional processing vary between languages.

**Automatic language identification Info**

If you don't know the language spoken in your audio files, choose this option. You have access to fewer options for additional processing than if you choose **Specific language**.

**Language**

Choose the language of the input audio.

English, US (en-US)

**► Additional settings**

The input location *must* be an object within an Amazon S3 bucket. For output location, you can choose a secure Amazon S3 service-managed bucket or you can specify your own Amazon S3 bucket.

If you choose a service-managed bucket, you can view a transcript preview in the AWS Management Console and you can download your transcript from the job details page (see below).

If you choose your own Amazon S3 bucket, you cannot see a preview in the AWS Management Console and must go to the Amazon S3 bucket to download your transcript.

**Input data** [Info](#)

**Input file location on S3**  
Choose an input audio or video file in Amazon S3.

[Browse S3](#)

Valid file formats: MP3, MP4, WAV, FLAC, AMR, OGG, and WebM.

**Output data**

**Output data location type** [Info](#)

**Service-managed S3 bucket**  
The output will be removed after 90 days when the job expires.

**Customer specified S3 bucket**  
The output will not be removed from bucket even after the job expires.

**Subtitle file format** [Info](#)

SRT (SubRip)  
 VTT (WebVTT)

**Tags - optional**  
A tag is a label you can add to a resource as metadata to help you organize, search, or filter your data. Each tag consists of a key and an optional value, in the form 'key:value'.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Next](#)

Click **Next**.

3. Select any desired options on the **Configure job** page. If you want to use [Custom vocabularies](#) (p. 64) or [Custom language models](#) (p. 107) with your transcription, you must create these before starting your transcription job.

## Configure job - *optional* Info

### Audio settings

**Audio identification** Info  
Choose to split multi-channel audio into separate channels for transcription, or identify speakers in the input audio.

**Alternative results** Info  
Enable to view more transcription results

### 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.

**Automatic content redaction** Info  
Automatic content redaction removes personally identifiable information (PII) in your transcripts. Redactions in transcripts show up as [PII].

**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.

Cancel
Previous
Create job

Click **Create job**.

4. You're now the **Transcription jobs** page. Here you can see the status of the transcription job. Once complete, click on the name of your transcription.

Transcription jobs (5) <small>Info</small>							<input type="button" value="Download"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	<input type="button" value="Create job"/>
							Status: All <small>▼</small>			
Name	Status	Language	Language settings	Model type	Model name	Created				
my-first-transcription	Complete	English, US (en-US)	Specific language	General	-	October 18 2021, 16:48 (UTC-07:00)	89 days			

5. You're now viewing the **Job details** page for your transcription. Here you can view all of the options you specified when setting up your transcription job.

To view your transcript, select the linked filepath in the right column under **Output data location**. This takes you to the Amazon S3 output folder you specified. Click on the name of your output file, which now has a .json extension.

my-first-transcription-job			
Job details			
Name my-first-transcription-job	Model None	Audio identification Disabled	Input data location <a href="#">s3://DOC-EXAMPLE-BUCKET/my-media-file.flac</a>
Status <span style="color: green;">⌚ Complete</span>	Created 2/4/2022, 12:11:31 PM	Alternative results Disabled	Output data location Service-managed S3 bucket
Language English, US (en-US)	Started 2/4/2022, 12:11:31 PM	Custom vocabulary None	
Language settings Specific language	Ended 2/4/2022, 12:12:50 PM	PII redaction Disabled	
Expiration <a href="#">Info</a> The transcription is available for 87 more days.	Input file format flac	Vocabulary filter -	
	Audio sampling rate 44100 Hz		

6. How you download your transcript depends on whether you chose a service-managed Amazon S3 bucket or your own Amazon S3 bucket.
  - a. If you chose a service-managed bucket, you can see a **Transcription preview** pane on your transcription job's information page, along with a **Download** button.

The screenshot shows the AWS Management Console interface for a transcription job named "my-first-transcription-job".

**Job details:**

Name	Model	Audio Identification	Input data location
my-first-transcription-job	None	Disabled	s3://DOC-EXAMPLE-BUCKET/my-media-file.flac
Status	Created	Alternative results	Output data location
⌚ Complete	2/4/2022, 12:11:31 PM	Disabled	Service-managed S3 bucket
Language	Started	Custom vocabulary	
English, US (en-US)	2/4/2022, 12:11:31 PM	None	
Language settings	Ended	PII redaction	
Specific language	2/4/2022, 12:12:50 PM	Disabled	
Expiration <a href="#">Info</a>	Input file format	Vocabulary filter	
The transcription is available for 87 more days.	flac	-	
	Audio sampling rate		
	44100 Hz		

**Transcription preview:**

You can see the first 5,000 characters of the transcription text below. To download the full text, choose Download full transcript.

**Text** | **Audio identification** | **Subtitles**

This is a preview of the content of your transcript. If your transcript is long, you may have to scroll to see the complete preview.

Click on the **Download** button and choose **Download transcript**.

- b. If you chose your own Amazon S3 bucket, you don't see any text in the **Transcription preview** pane on your transcription job's information page. Instead, you see a blue information box with a link to the Amazon S3 bucket you chose.

The screenshot shows the AWS Management Console interface for a transcription job named "my-first-transcription-job".

**Job details:**

Name	Model	Audio identification	Input data location
my-first-transcription-job	None	Disabled	<a href="#">S3://DOC-EXAMPLE-BUCKET/my-media-file.flac</a>
Status	Created	Alternative results	Output data location
<span style="color: green;">Complete</span>	2/7/2022, 11:42:17 AM	Disabled	<a href="#">https://s3.us-west-2.amazonaws.com/DOC-EXAMPLE-BUCKET</a>
Language	Started	Custom vocabulary	
English, US (en-US)	2/7/2022, 11:42:17 AM	None	
Language settings	Ended	PII redaction	
Specific language	2/7/2022, 11:43:37 AM	Disabled	
Expiration <a href="#">Info</a>	Input file format	Vocabulary filter	
The transcription is available for 89 more days.	flac	-	
	Audio sampling rate		
	44100 Hz		

**Transcription preview:**

Select download to save a local copy of the transcription.

[Download ▾](#)

**Text** | **Audio identification** | **Subtitles**

**Info:** When you use your own S3 bucket for transcription output, Amazon Transcribe does not show the output in the console. You open the output file from your [S3 Bucket](#).

To access your transcript, go to the specified Amazon S3 bucket using either the link under **Output data location** in the **Job details** pane or the **S3 Bucket** link within the blue information box in the **Transcription preview** pane.

## Streaming transcriptions

- From the [AWS Management Console](#), select **Real-time transcription** in the left navigation pane. This takes you to the main streaming page where you can select options prior to starting your stream.

The screenshot shows the "Real-time transcription" page in the AWS Management Console.

**Left sidebar:**

- Real-time transcription** (selected)
- Transcription jobs
- Custom language model
- Custom vocabulary
- Vocabulary filtering
- Amazon Transcribe Call Analytics** (collapsed)
  - Call analytics jobs
  - Call analytics categories

**Right pane:**

**Real-time transcription [Info](#)**

See how Amazon Transcribe creates a text copy of speech in real time. Choose **Start streaming** and talk.

**Transcription**

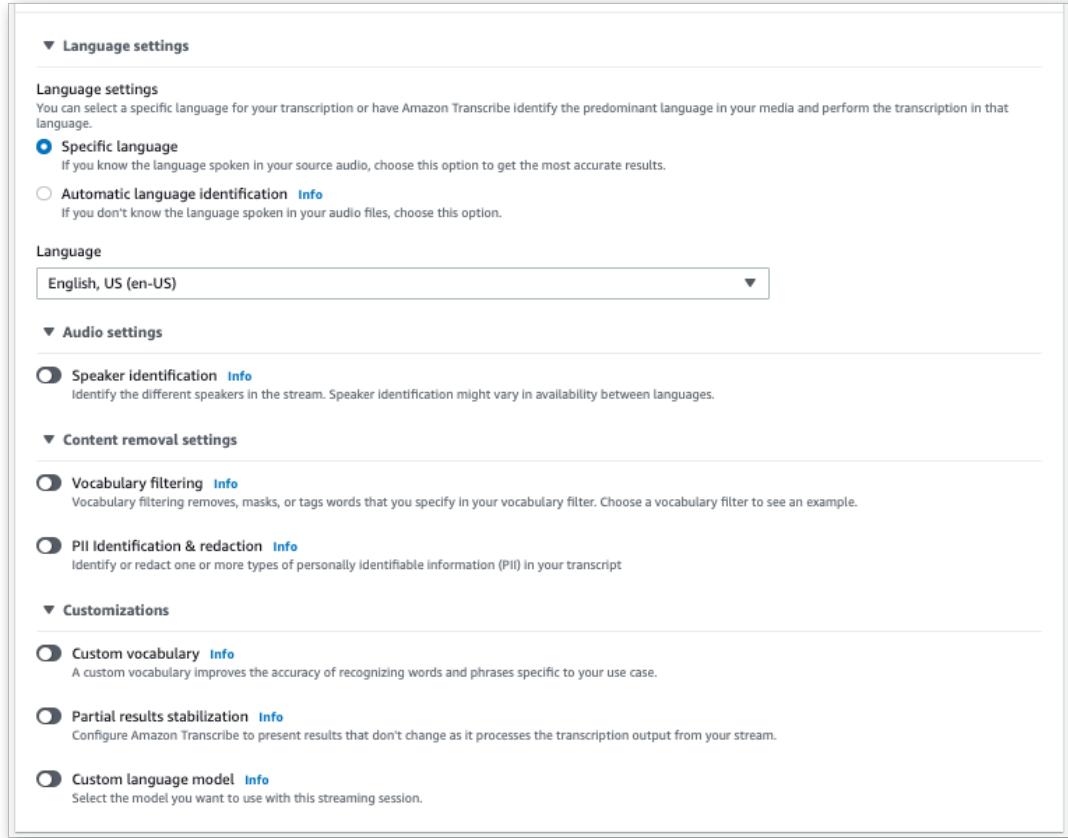
[Download full transcript](#) | **Start streaming**

**Transcription output**

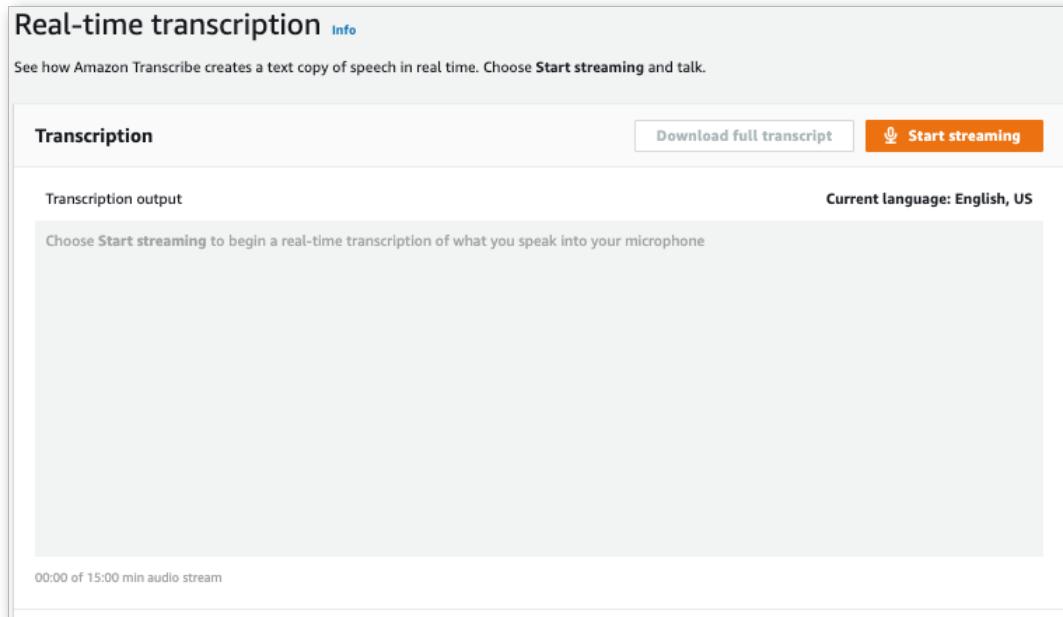
Choose Start streaming to begin a real-time transcription of what you speak into your microphone

Current language: English, US

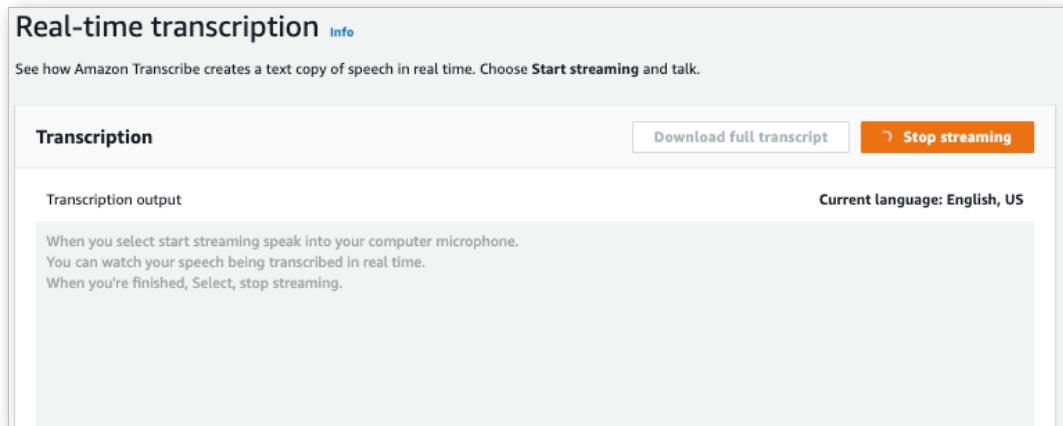
2. Below the **Transcription output** box, you have the option to select various language and audio settings.



3. After you've selected the appropriate settings, scroll to the top of the page and choose **Start streaming**, then begin speaking into your computer microphone. You can see your speech transcribed in real time.



4. When you're finished, select **Stop streaming**.



You can now download your transcript by selecting **Download full transcript**.

## Transcribing with the AWS CLI

When using the AWS CLI to start a transcription, you can either run all commands at the CLI level or you can run the Amazon Transcribe command you wish to use followed by the AWS Region and the location of a JSON file that contains a request body. Examples throughout this guide show both methods; however, this section focuses on the former method.

The AWS CLI does not support streaming transcriptions.

Before you continue, make sure you've:

- Uploaded your media file into an Amazon S3 bucket. If you're unsure how to create an Amazon S3 bucket or upload your file, refer to [Create your first Amazon S3 bucket](#) and [Upload an object to your bucket](#).
- Installed the [AWS CLI \(p. 22\)](#).

You can find all AWS CLI commands for Amazon Transcribe in the [AWS CLI Command Reference](#).

## Starting a new transcription job

To start a new transcription, use the `start-transcription-job` command.

1. In a terminal window, type the following:

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

A '`'` appears on the next line and you can now continue adding required parameters, as described in the next step.

You can also omit the '`\`' and append all parameters, separating each with a space.

2. With the `start-transcription-job` command, you must include `region`, `transcription-job-name`, `media`, and either `language-code` or `identify-language`.

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

If appending all parameters, this request looks like:

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

If you choose not to specify an output bucket using `output-bucket-name`, Amazon Transcribe places your transcription output in a service-managed bucket. Transcripts stored in a service-managed bucket expire after 90 days.

Amazon Transcribe responds with:

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

To see if your transcription job is successful, as indicated by `TranscriptionJobStatus` changing from `IN_PROGRESS` to `COMPLETED`, use the `get-transcription-job` or `list-transcription-jobs` command, as shown in the following section.

## Getting the status of a transcription job

To get information about your transcription job, use the `get-transcription-job` command.

The only required parameters for this command are the AWS Region where the job is located and the name of the job.

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

Amazon Transcribe responds with:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "my-first-transcription-job",
        "TranscriptionJobStatus": "COMPLETED",
        "LanguageCode": "en-US",
        "MediaSampleRateHertz": 48000,
        "MediaFormat": "flac",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
        },
        "Transcript": {
            "TranscriptFileUri": "https://s3.the-URI-where-your-job-is-located"
        },
        "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
        }
    }
}
```

If you've selected your own Amazon S3 bucket for your transcription output, this bucket is listed with `TranscriptFileUri`. If you've selected a service-managed bucket, a temporary URI is provided; use this URI to download your transcript.

**Note**

Temporary URIs for service-managed Amazon S3 buckets are only valid for 15 minutes. If you get an `AccessDenied` error when using the URI, run the `get-transcription-job` request again to get a new temporary URI.

## Listing your transcription jobs

To list all your transcription jobs in a given AWS Region, use the `list-transcription-jobs` command.

The only required parameter for this command is the AWS Region in which your transcription jobs are located.

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

Amazon Transcribe responds with:

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

## Deleting your transcription job

To delete your transcription job, use the `delete-transcription-job` command.

The only required parameters for this command are the AWS Region where the job is located and the name of the job.

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

To confirm your delete request is successful, you can run the `list-transcription-jobs` command; your job should no longer appear in the list.

## Transcribing with the AWS SDKs

You can use SDKs for both batch and streaming transcriptions.

To get started with batch transcriptions, refer to [Batch transcriptions \(p. 34\)](#). Note that prior to starting a batch transcription, you must first upload your media file into an Amazon S3 bucket.

To get started with streaming transcriptions, refer to [Streaming transcriptions \(p. 37\)](#). For streaming transcriptions, you can transcribe a streamed media file or a live media stream. Note that streaming transcriptions are not supported with the following AWS SDKs: Java V1, .NET, Python, or PowerShell. If you wish to use Python for streaming, there's a custom [Python SDK for Amazon Transcribe](#) available.

To view supported media formats and other media requirements and constraints, see [Data input and output \(p. 10\)](#).

**Tip**

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

### Batch transcriptions

You can create batch transcriptions using the URI of a media file located in an Amazon S3 bucket. If you're unsure how to create an Amazon S3 bucket or upload your file, refer to [Create your first S3 bucket](#) and [Upload an object to your bucket](#).

Java

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

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

    public static void main(String args[]) {

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

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

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

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

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

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

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

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

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

### JavaScript

```

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

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

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

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

startTranscriptionRequest();

```

### Python

```

import time
import boto3

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

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

```

```
        print(f"Waiting for {job_name}. Current status is {job_status}.")  
        time.sleep(10)  
  
def main():  
    transcribe_client = boto3.client('transcribe', region_name='us-west-2')  
    file_uri = 's3://DOC-EXAMPLE-BUCKET/my-media-file.flac'  
    transcribe_file('Example-job', file_uri, transcribe_client)  
  
if __name__ == '__main__':  
    main()
```

## Streaming transcriptions

You can create streaming transcriptions using a streamed media file or a live media stream.

Note that the standard AWS SDK for Python (Boto3) is not supported for Amazon Transcribe streaming. To start a streaming transcription using Python, use this [async Python SDK for Amazon Transcribe](#).

### Java

The following example is a Java program that transcribes streaming audio.

To run this example, note the following:

- You must use the [AWS SDK for Java 2.x](#).
- Clients must use Java 1.8 to be compatible with the [AWS SDK for Java 2.x](#).
- The sample rate you specify must match the actual sample rate of your audio stream.

See also: [Retry client for Amazon Transcribe streaming \(Java SDK\)](#). This code manages the connection to Amazon Transcribe and retries sending data when there are errors on the connection. For example, if there is a transient error on the network, this client resends the request that failed.

```
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 16kHz, 16 bit sample size, mono  
        int sampleRate = 16000;  
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
```

```

        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

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

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

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

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

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

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("== All records stream successfully ===");
            })
            .subscriber(event -> {
                List<Result> results = ((TranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {
                        System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
            .build();
    }

    private InputStream getStreamFromFile(String 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 final 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

The following example is a Python program that transcribes streaming audio.

To run this example, note the following:

- You must use this [SDK for Python](#).
- The sample rate you specify must match the actual sample rate of your audio stream.

```
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 realtime 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()
```

## Using this service with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
<a href="#">AWS SDK for C++</a>	<a href="#">AWS SDK for C++ code examples</a>
<a href="#">AWS SDK for Go</a>	<a href="#">AWS SDK for Go code examples</a>
<a href="#">AWS SDK for Java</a>	<a href="#">AWS SDK for Java code examples</a>
<a href="#">AWS SDK for JavaScript</a>	<a href="#">AWS SDK for JavaScript code examples</a>
<a href="#">AWS SDK for .NET</a>	<a href="#">AWS SDK for .NET code examples</a>
<a href="#">AWS SDK for PHP</a>	<a href="#">AWS SDK for PHP code examples</a>
<a href="#">AWS SDK for Python (Boto3)</a>	<a href="#">AWS SDK for Python (Boto3) code examples</a>
<a href="#">AWS SDK for Ruby</a>	<a href="#">AWS SDK for Ruby code examples</a>

For examples specific to this service, see [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#).

### Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

## Transcribing with HTTP or WebSockets

Amazon Transcribe supports HTTP for both batch (HTTP/1.1) and streaming (HTTP/2) transcriptions. WebSockets are supported for streaming transcriptions.

Both HTTP and WebSockets require you to authenticate your request using AWS Signature Version 4 headers. Refer to [Signing AWS API requests](#) for more information.

### Batch transcriptions

You can make an HTTP request for batch transcriptions using the following headers:

- host
- x-amz-target
- content-type
- x-amz-content-sha256
- x-amz-date
- authorization

Here's an example of a StartTranscriptionJob request:

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

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

Additional operations and parameters are listed in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section. Other signature elements are detailed in [Elements of an AWS Signature Version 4 request](#).

## Streaming transcriptions

Streaming transcriptions using HTTP/2 and WebSockets are more involved than using SDKs, so we advise reviewing the [Setting up a streaming transcription \(p. 48\)](#) section before setting up your first stream.

For more information on these methods, refer to [Setting up an HTTP/2 stream \(p. 50\)](#) or [Setting up a WebSocket stream \(p. 54\)](#).

# Transcribing streaming audio

Using Amazon Transcribe streaming, you can produce real-time transcriptions for your media content. Unlike batch transcriptions, which involve uploading media files, streaming media is delivered to Amazon Transcribe in real time. Amazon Transcribe then returns a transcript, also in real time.

Streaming can include pre-recorded media (movies, music, and podcasts) and real-time media (live news broadcasts). Common streaming use cases for Amazon Transcribe include live closed captioning for sporting events and real-time monitoring of call center audio.

Streaming content is delivered as a series of sequential data packets, or 'chunks,' that Amazon Transcribe transcribes instantaneously. The advantages to using streaming over batch include real-time speech-to-text capabilities in your applications and faster transcription times. However, this increased speed may have accuracy limitations in some cases.

Amazon Transcribe offers the following options for streaming:

- [AWS SDKs](#)
- [HTTP/2 \(p. 50\)](#)
- [WebSockets \(p. 54\)](#)
- [AWS Management Console](#)

We recommend using the AWS Management Console or an SDK, rather than using HTTP/2 or WebSockets directly.

Audio formats supported for streaming transcriptions are:

- FLAC
- OPUS-encoded audio in an Ogg container
- PCM (only signed 16-bit little-endian audio formats, which does **not** include WAV)

Lossless formats (FLAC or PCM) are recommended.

**Note**

Streaming transcriptions are not supported with all languages. See [Supported languages and language-specific features \(p. 3\)](#) for details.

## Streaming and partial results

Because streaming works in real time, transcripts are produced in *partial results*. Amazon Transcribe breaks up the incoming audio stream based on natural speech segments, such as a change in speaker or a pause in the audio. The transcription is returned to your application in a stream of transcription events, with each response containing more transcribed speech until an entire segment is transcribed.

An approximation of this is shown in the following code block. You can view this process in action by signing into the [AWS Management Console](#), selecting **Real-time transcription**, and speaking into your microphone. Watch the **Transcription output** pane as you speak.

In this example, each line is the partial result of an audio segment.

```
The  
The ad.
```

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

These partial results are present in your transcription output within the [Results](#) objects. Also in this object block is an **IsPartial** field. If this field is true, your transcription segment is not yet complete. You can view the difference between an incomplete and a complete segment below:

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

"Transcript": "The Amazon is the largest"
    }
],
"EndTime": 3.275,
"IsPartial": true,
"ResultId": "55421b61-7cc6-4ef1-8d55-08a97159870e",
"StartTime": 1.26
}

"IsPartial": false (complete segment)

"Transcript": "The Amazon is the largest rain forest on the planet."
    }
],
"EndTime": 5.735,
"IsPartial": false,
"ResultId": "55421b61-7cc6-4ef1-8d55-08a97159870e",
"StartTime": 1.26
}
```

Each word within a complete segment has an associated confidence score, which is a value between 0 and 1. A larger value indicates a greater likelihood that the word is correctly transcribed.

**Tip**

The `StartTime` and `EndTime` of an audio segment can be used to synchronize transcription output with video dialogue.

If you're running an application that requires low latency, you may want to use [partial-result stabilization \(p. 45\)](#).

## Partial-result stabilization

Amazon Transcribe starts returning transcription results as soon as you start streaming your audio. It returns these partial results incrementally until it generates a finished result at the level of a natural speech segment. A natural speech segment is continuous speech that contains a pause or a change in speaker.

Amazon Transcribe continues outputting partial results until it generates the final transcription result for a speech segment. Because speech recognition may revise words as it gains more context, streaming transcriptions can change slightly with each new partial result output.

This process gives you two options for each speech segment:

- Wait for the finished segment
- Use the segment's partial results

Partial result stabilization changes how Amazon Transcribe produces the final transcription result for each complete segment. When activated, only the last few words from the partial results can change. Because of this, transcription accuracy may be affected. However, your transcript is returned faster than without partial-results stabilization. This reduction in latency can help use cases, such as subtitleing videos or generating captions for live streams.

The following examples show how the same audio stream is handled when partial-results stabilization is not activated and when it is. Note that you can set the stability level to low, medium, or high. Low stability provides the highest accuracy. High stability transcribes faster, but with slightly lower accuracy. You can also see that by the end of the segment, transcription time is quite similar.

<b>Transcription output</b>	<b>"EndTime":</b>
Partial-result stabilization not enabled	
The The Amazon.  The Amazon is the large  The Amazon is the largest  The Amazon is the largest  The Amazon is the largest rainforest on  The Amazon is the largest rainforest on the planet. The Amazon is the largest rainforest on the planet.	0.515 1.015 1.515 2.015 2.515 3.015 3.515 3.885
Partial-result stabilization enabled (high stability)	
The The Amazon.  The Amazon is the large  The Amazon is the largest  The Amazon is the largest rain for  The Amazon is the largest rain forest on  The Amazon is the largest rain forest on the plan. The Amazon is the largest rain forest on the planet.	0.095 1.015 1.515 2.015 2.495 3.015 3.515 4.015

When you activate partial-result stabilization, Amazon Transcribe uses the `Stable` field to indicate whether an 'item' is stable, where 'item' refers to a transcribed word or phrase. Values for `Stable` are either `true`, indicating that the item is stable, or `false`, indicating that the item is not stable.

Items flagged as `false` (not stable) are more likely to change as your segment is transcribed. Conversely, items flagged as `true` (stable) don't change.

You can choose to render non-stable words so your captions align with speech. Even if captions change slightly as context is added, this is a better user experience than periodic text bursts, which may or may not align with speech.

You can also choose to display non-stable words in a different format, such as italics, to indicate to viewers that these words may change. Displaying partial results limits the amount of text displayed at a given time. This can be important when you're dealing with space constraints, as with video captions.

Here's a blog post that dives a little deeper: [Improve the streaming transcription experience with Amazon Transcribe partial results stabilization](#)

## Partial-result stabilization example output

The following example output shows `Stable` flags for an incomplete segment (`"IsPartial": true`). You can see that the words "*the large*" are not stable and therefore could change before the segment is finalized.

```
"Transcript": {
    "Results": [
        {
            "Alternatives": [
                {
                    "Items": [
                        {
                            "Content": "The",
                            "EndTime": 1.4475,
                            "Stable": true,
                            "StartTime": 1.26,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": "Amazon",
                            "EndTime": 2.1725,
                            "Stable": true,
                            "StartTime": 1.4475,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": "is",
                            "EndTime": 2.3975,
                            "Stable": true,
                            "StartTime": 2.1725,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": "the",
                            "EndTime": 2.5875,
                            "Stable": false,
                            "StartTime": 2.3975,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": "large",
                            "EndTime": 2.775,
                            "Stable": false,
                            "StartTime": 2.5875,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        }
                    ]
                }
            ]
        }
    ]
}
```

```
        "StartTime": 2.5875,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    }
],
"Transcript": "The Amazon is the large"
],
"EndTime": 2.775,
"IsPartial": true,
"ResultId": "55421b61-7cc6-4ef1-8d55-08a97159870e",
"StartTime": 1.26
}
]
```

## Setting up a streaming transcription

This section expands on the main [streaming \(p. 44\)](#) section and is intended to provide information for users who want to set up their stream with HTTP/2 or WebSockets directly, rather than with an AWS SDK. The information in this chapter can also be used to build your own SDK.

To start streaming using an AWS SDK, see [Transcribing with the AWS SDKs \(p. 34\)](#).

### Event stream encoding

Amazon Transcribe uses a format called event stream encoding for streaming transcriptions.

Event stream encoding provides bidirectional communication between a client and a server. Data frames sent to the Amazon Transcribe streaming service are encoded in this format. The response from Amazon Transcribe also uses this encoding.

Each message consists of two sections: the prelude and the data. The prelude consists of:

1. The total byte length of the message
2. The combined byte length of all headers

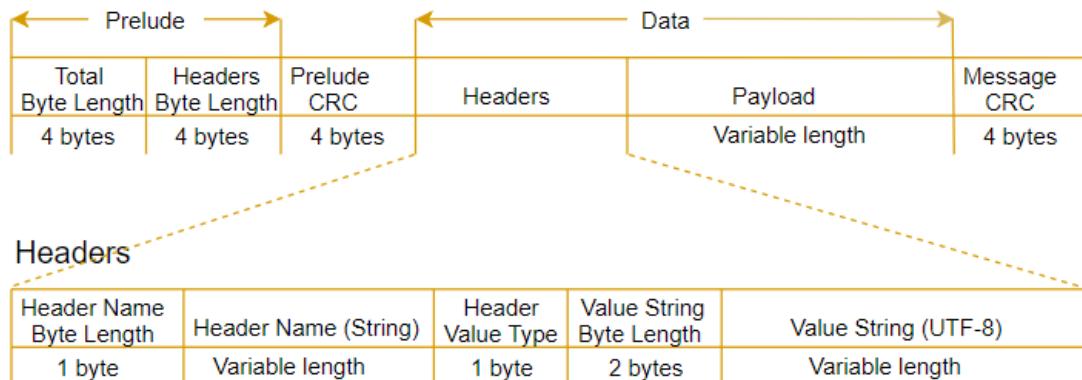
The data section consists of:

1. Headers
2. Payload

Each section ends with a 4-byte big-endian integer cyclic redundancy check (CRC) checksum. The message CRC checksum is for both the prelude section and the data section. Amazon Transcribe uses CRC32 (often referred to as GZIP CRC32) to calculate both CRCs. For more information about CRC32, see [GZIP file format specification version 4.3](#).

Total message overhead, including the prelude and both checksums, is 16 bytes.

The following diagram shows the components that make up a message and a header. There are multiple headers per message.



Each message contains the following components:

- **Prelude:** Consists of two, 4-byte fields, for a fixed total of 8 bytes.
  - *First 4 bytes:* The big-endian integer byte-length of the entire message, inclusive of this 4-byte length field.
  - *Second 4 bytes:* The big-endian integer byte-length of the 'headers' portion of the message, excluding the 'headers' length field itself.
- **Prelude CRC:** The 4-byte CRC checksum for the prelude portion of the message, excluding the CRC itself. The prelude has a separate CRC from the message CRC to ensure that Amazon Transcribe can detect corrupted byte-length information immediately without causing errors, such as buffer overruns.
- **Headers:** Metadata annotating the message; for example, message type and content type. Messages have multiple headers, which are key:value pairs, where the key is a UTF-8 string. Headers can appear in any order in the 'headers' portion of the message, and each header can appear only once.
- **Payload:** The audio content to be transcribed.
- **Message CRC:** The 4-byte CRC checksum from the start of the message to the start of the checksum. That is, everything in the message except the CRC itself.

The header frame is the authorization frame for the streaming transcription. Amazon Transcribe uses the value of the authorization header as the seed for generating a chain of authorization headers for the data frames in the request.

Each header contains the following components; there are multiple headers per frame.

- **Header name byte-length:** The byte-length of the header name.
- **Header name:** The name of the header that indicates the header type. For valid values, see the following frame descriptions.
- **Header value type:** A number indicating the header value. The following list shows the possible values for the header and what they indicate.
  - 0 – TRUE
  - 1 – FALSE
  - 2 – BYTE
  - 3 – SHORT
  - 4 – INTEGER
  - 5 – LONG
  - 6 – BYTE ARRAY
  - 7 – STRING
  - 8 – TIMESTAMP
  - 9 – UUID

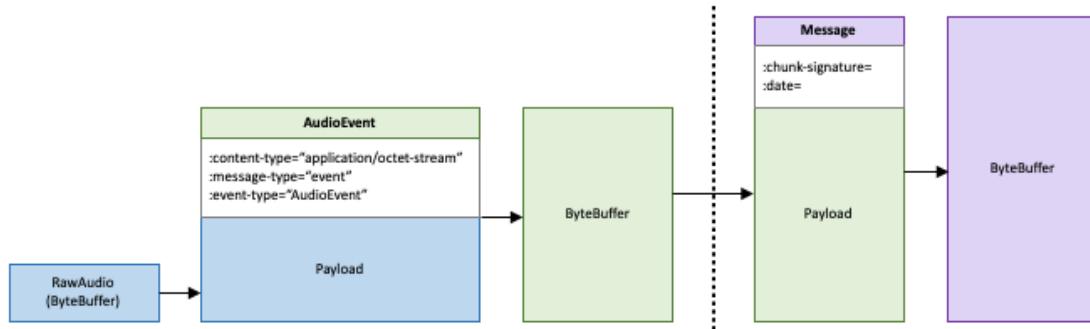
- **Value string byte length:** The byte length of the header value string.
- **Header value:** The value of the header string. Valid values for this field depend on the type of header. See [Setting up an HTTP/2 stream \(p. 50\)](#) or [Setting up a WebSocket stream \(p. 54\)](#) for more information.

## Data frames

Each streaming request contains one or more data frames. There are two steps to creating a data frame:

1. Combine raw audio data with metadata to create the payload of your request.
2. Combine the payload with a signature to form the event message that is sent to Amazon Transcribe.

The following diagram shows how this works.



You can find more information on data frames in the [Setting up an HTTP/2 stream \(p. 50\)](#) and [Setting up a WebSocket stream \(p. 54\)](#) sections.

## Setting up an HTTP/2 stream

The key components for an [HTTP/2 protocol](#) for streaming transcription requests with Amazon Transcribe are:

- A header frame. This contains the HTTP/2 headers for your request, and a signature in the authorization header that Amazon Transcribe uses as a seed signature to sign the data frames.
- One or more message frames in [event stream encoding \(p. 48\)](#) that contain metadata and raw audio bytes.
- An end frame. This is a signed message in [event stream encoding \(p. 48\)](#) with an empty body.

### Note

Amazon Transcribe only supports one stream per HTTP/2 session. If you attempt to use multiple streams, your transcription request fails.

1. Attach the following policy to the IAM role that makes the request. See [Adding IAM policies](#) for more information.

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

```
        }
    ]
}
```

2. To start the session, send an HTTP/2 request to Amazon Transcribe.

```
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
```

Additional operations and parameters are listed in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

Amazon Transcribe sends the following response:

```
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. Create an audio event that contains your audio data. Combine the headers—described in the following table—with a chunk of audio bytes in an event-encoded message. To create the payload for the event message, use a buffer in raw-byte format.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	24	application/ octet-stream
11	:event-type	7	10	AudioEvent
13	:message-type	7	5	event

Binary data in this example request are base64-encoded. In an actual request, data are raw bytes.

```
:content-type: "application/vnd.amazon.eventstream"
:event-type: "AudioEvent"
:message-type: "event"
Uk1GRjzxPQBXQVFZm10IBAAAAABAAEAgD4AAAB9AACABAAZGF0YVTwPQAAAAAAAAAAAAAAD//wIA/
f8EAA==
```

4. Create an audio message that contains your audio data.

- a. Your audio message data frame contains event-encoding headers that include the current date and a signature for the audio chunk and the audio event.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value
16	:chunk-signature	6	varies	generated signature
5	:date	8	8	timestamp

Binary data in this request are base64-encoded. In an actual request, data are raw bytes.

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

AAAAA0gAAAIKVоРFcTTcjb250ZW50LXR5cGUHABhhcHBsaWNhdGlvbi9vY3RldC1zdHJlYW0LOmV2ZW50LXR5
cGUHAApBdWRpb0V2ZW50DTptZXNzYwd1LXR5cGUHAAV1dmVudAxDb256ZW50LVR5cGUHABphcHBsaWNhdGlv
bi94LWFtei1qc29uLTEuMVJJRKy88T0AV0FWRWZtdCAQAAAAAQABAIA
+AAAfQAAAgaQAGRhdGFU8D0AAAAAA
AAAAAAAAAA//8CAP3/BAC7QLFF
```

- b. Construct a string to sign, as outlined in [Create a string to sign for Signature Version 4](#). Your string follows this format:

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

- **DateTime**: The date and time the signature is created. The format is YYYYMMDDTHHMMSSZ, where YYYY=year, MM=month, DD=day, HH=hour, MM=minute, SS=seconds, and 'T' and 'Z' are fixed characters. For more information, refer to [Handling Dates in Signature Version 4](#).
  - **Keypath**: The signature scope in the format date/region/service/aws4\_request. For example, 20220127/us-west-2/transcribe/aws4\_request.
  - **Hex**: A function that encodes input into a hexadecimal representation.
  - **priorSignature**: The signature for the previous frame. For the first data frame, use the signature of the header frame.
  - **HexHash**: A function that first creates a SHA-256 hash of its input and then uses the Hex function to encode the hash.
  - **nonSignatureHeaders**: The DateTime header encoded as a string.
  - **payload**: The byte buffer containing the audio event data.
- c. Derive a signing key from your AWS secret access key and use it to sign the `stringToSign`. For a greater degree of protection, the derived key is specific to the date, service, and AWS Region. For more information, see [Calculate the signature for AWSSignature Version 4](#).

Make sure you implement the `GetSignatureKey` function to derive your signing key. If you have not yet derived a signing key, refer to [Examples of how to derive a signing key for Signature Version 4](#).

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

- **HMACSHA256:** A function that creates a signature using the SHA-256 hash function.
- **derivedSigningKey:** The Signature Version 4 signing key.
- **stringToSign:** The string you calculated for the data frame.

After you've calculated the signature for the data frame, construct a byte buffer containing the date, signature, and audio event payload. Send the byte array to Amazon Transcribe for transcription.

5. To indicate the audio stream is complete, send an end frame (an empty data frame) that contains only the date and signature. You construct this end frame the same way you construct a data frame.

Amazon Transcribe responds with a stream of transcription events, sent to your application. This response is event stream encoded. It contains the standard prelude and the following headers.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	15	TranscriptEvent
13	:message-type	7	5	event

The events are sent in raw-byte format. In this example, the bytes are base64-encoded.

```
AAAAUwAAEPIRHpyBTPkYXRlCAAAAWixUkMLEDpjAHVuay1zaWduYXR1cmUGACCt6Zy+uymwEK2SrLp/zVBI  
5eGn83jdBwCaRUBJA+eaDafqjqI=
```

To see the transcription results, decode the raw bytes using event stream encoding.

```
:content-type: "application/vnd.amazon.eventstream"  
:event-type: "TranscriptEvent"  
:message-type: "event"  
  
{  
    "Transcript":  
        {  
            "Results":  
                [  
                    results  
                ]  
        }  
}
```

6. To end your stream, send an empty audio event to Amazon Transcribe. Create the audio event exactly like any other, except with an empty payload. Sign the event and include the signature in the `:chunk-signature` header, as follows:

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

## Handling HTTP/2 streaming errors

If an error occurs when processing your media stream, Amazon Transcribe sends an exception response. The response is event stream encoded.

The response contains the standard prelude and the following headers:

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	19	BadRequestException
13	:message-type	7	9	exception

When the exception response is decoded, it contains the following information:

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

Exception message
```

## Setting up a WebSocket stream

The key components for a [WebSocket protocol](#) for streaming transcription requests with Amazon Transcribe are:

- The upgrade request. This contains the query parameters for your request, and a signature that Amazon Transcribe uses as a seed signature to sign the data frames.
- One or more message frames in [event stream encoding \(p. 48\)](#) that contain metadata and raw audio bytes.
- An end frame. This is a signed message in [event stream encoding \(p. 48\)](#) with an empty body.

### Note

Amazon Transcribe only supports one stream per WebSocket session. If you attempt to use multiple streams, your transcription request fails.

1. Attach the following policy to the IAM role that makes the request. See [Adding IAM policies](#) for more information.

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

2. To start the session, create a pre-signed URL in the following format. Line breaks have been added for readability.

```
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=250
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type;host;x-amz-date
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
```

Additional operations and parameters are listed in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

To construct the URL for your request and create the [Signature Version 4 signature](#), refer to the following steps. Examples are in pseudocode.

- Create a canonical request. A canonical request is a string that includes information from your request in a standardized format. This ensures that when AWS receives the request, it can calculate the same signature you created for your URL. For more information, see [Create a Canonical Request for Signature Version 4](#).

```
# 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"
```

- Create a canonical URI, which is the part of the URI between the domain and the query string.

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

- Create the canonical headers and signed headers. Note the trailing \n in the canonical headers.
  - Append the lowercase header name followed by a colon (:).
  - Append a comma-separated list of values for that header. Do not sort values in headers that have multiple values.
  - Append a new line (\n).

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

- Match the algorithm to the hashing algorithm. You must use SHA-256.

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

- e. Create the credential scope, which scopes the derived key to the date, AWS Region, and service. For example, `20220127/us-west-2/transcribe/aws4_request`.

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

- f. Create the canonical query string. Query string values must be URI-encoded and sorted by name.
  - Sort the parameter names by character code point in ascending order. Parameters with duplicate names should be sorted by value. For example, a parameter name that begins with the uppercase letter F precedes a parameter name that begins with the lowercase letter b.
  - Do not URI-encode any of the unreserved characters that [RFC 3986](#) defines: A-Z, a-z, 0-9, hyphen (-), underscore (\_), period (.), and tilde (~).
  - Percent-encode all other characters with %XY, where X and Y are hexadecimal characters (0-9 and uppercase A-F). For example, the space character must be encoded as %20 (don't include '+' as some encoding schemes do); extended UTF-8 characters must be in the form %XY%ZA %BC.
  - Double-encode any equals (=) characters in parameter values.

```
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=250"
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. Create a hash of the payload. For a GET request, the payload is an empty string.

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

- h. Combine the following elements to create the canonical request.

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

3. Create the string to sign, which contains meta information about your request. You use the string to sign in the next step when you calculate the request signature. For more information, see [Create a String to Sign for Signature Version 4](#).

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

4. Calculate the signature. To do this, derive a signing key from your AWS secret access key. For a greater degree of protection, the derived key is specific to the date, service, and AWS Region. Use this derived key to sign the request. For more information, see [Calculate the Signature for AWS Signature Version 4](#).

Make sure you implement the `GetSignatureKey` function to derive your signing key. If you have not yet derived a signing key, refer to [Examples of how to derive a signing key for Signature Version 4](#).

```
#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
```

The function `HMAC(key, data)` represents an HMAC-SHA256 function that returns results in binary format.

5. Add signing information to the request and create the request URL.

After you calculate the signature, add it to the query string. For more information, see [Add the Signature to the Request](#).

First, add the authentication information to the query string.

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

Second, create the URL for the request.

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

Use the request URL with your WebSocket library to make the request to Amazon Transcribe.

6. The request to Amazon Transcribe must include the following headers. Typically these headers are managed by your WebSocket client library.

```
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. When Amazon Transcribe receives your WebSocket request, it responds with a WebSocket upgrade response. Typically your WebSocket library manages this response and sets up a socket for communications with Amazon Transcribe.

The following is the response from Amazon Transcribe. Line breaks have been added for readability.

```
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=250
&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. Make your WebSocket streaming request.

After the WebSocket connection is established, the client can start sending a sequence of audio frames, each encoded using [event stream encoding \(p. 48\)](#).

Each data frame contains three headers combined with a chunk of raw audio bytes; the following table describes these headers.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	24	application/octet-stream
11	:event-type	7	10	AudioEvent
13	:message-type	7	5	event

9. To end the data stream, send an empty audio chunk in an event stream encoded message.

The response contains event stream encoded raw bytes in the payload. It contains the standard prelude and the following headers.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	15	TranscriptEvent
13	:message-type	7	5	event

When you decode the binary response, you end up with a JSON structure containing the transcription results.

## Handling WebSocket streaming errors

If an exception occurs while processing your request, Amazon Transcribe responds with a terminal WebSocket frame containing an event stream encoded response. This response contains the headers described in the following table; the body of the response contains a descriptive error message. After sending the exception response, Amazon Transcribe sends a close frame.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	16	application/json
15	:exception-type	7	varies	varies, see below

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:message-type	7	9	exception

The exception-type header contains one of the following values:

- **BadRequestException**: There was a client error when the stream was created, or an error occurred while streaming data. Make sure your client is ready to accept data and try your request again.
- **InternalFailureException**: Amazon Transcribe had a problem during the handshake with the client. Try your request again.
- **LimitExceededException**: The client exceeded the concurrent stream limit. For more information, see [Amazon Transcribe Limits](#). Reduce the number of streams you're transcribing.
- **UnrecognizedClientException**: The WebSocket upgrade request was signed with an incorrect access key or secret key. Make sure you're correctly creating the access key and try your request again.

Amazon Transcribe can also return any of the common service errors. For a list, see [Common Errors](#).

# Tagging resources

A tag is a custom metadata label that you can add to a resource in order to make it easier to identify, organize, and find in a search. Tags are comprised of two individual parts: A tag key and a tag value. This is referred to as a key:value pair.

A tag key typically represents a larger category, while a tag value represents a subset of that category. For example you could have *tag key=Color* and *tag value=Blue*, which would produce the key:value pair *Color:Blue*. Note that you can set the value of a tag to an empty string, but you can't set the value of a tag to null. Omitting the tag value is the same as using an empty string.

## Tip

AWS Billing and Cost Management can use tags to separate your bills into dynamic categories.

For example, if you add tags to represent different departments within your company, such as *Department:Sales* or *Department:Legal*, AWS can provide you with your cost distribution per department.

In Amazon Transcribe, you can tag the following resources:

- Transcription jobs
- Medical transcription jobs
- Vocabularies
- Medical vocabularies
- Vocabulary filters
- Custom language models

Tag keys can be up to 128 characters in length and tag values can be up to 256 characters in length; both are case sensitive. Amazon Transcribe supports up to 50 tags per resource. For a given resource, each tag key must be unique with only one value. Note that your tags cannot begin with *aws:* because AWS reserves this prefix for system-generated tags. You cannot add, modify, or delete *aws:\** tags, and they don't count against your tags-per-resource limit.

## API operations specific to resource tagging

[ListTagsForResource](#), [TagResource](#), [UntagResource](#)

To use the tagging APIs, you need to include an Amazon Resource Name (ARN) with your request. ARNs have the format `arn:partition:service:region:account-id:resource-type/resource-id`. For example, the ARN associated with a transcription job may look like: `arn:aws:transcribe:us-west-2:111122223333:transcription-job/my-transcription-job-name`.

To learn more about tagging, including best practices, see [Tagging AWS resources](#).

## Tag-based access control

You can use tags to control access within your AWS accounts. For tag-based access control, you provide tag information in the condition element of an IAM policy. You can then use tags and their associated tag condition key to control access to:

- **Resources:** Control access to your Amazon Transcribe resources based on the tags you've assigned to those resources.
- Use the `aws:ResourceTag/key-name` condition key to specify which tag key:value pair must be attached to the resource.

- **Requests:** Control which tags can be passed in a request.
  - Use the `aws:RequestTag/key-name` condition key to specify which tags can be added, modified, or removed from an IAM user or role.
- **Authorization processes:** Control tag-based access for any part of your authorization process.
  - Use the `aws:TagKeys/` condition key to control whether specific tag keys can be used on a resource, in a request, or by a principal. In this case, the key value doesn't matter.

For more detailed information on tag-based access control, see [Controlling access to AWS resources using tags](#).

## Tagging your Amazon Transcribe resources

You can add tags before or after you run your Amazon Transcribe job. The existing **Create\*** and **Start\*** APIs allow you to add tags with your transcription job.

You can add, modify or delete tags using the **AWS Management Console**, **AWS CLI**, or **AWS SDK**; see the following for examples:

### AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Transcription jobs**, then select the **Create job** button (top right). This opens the **Specify job details** page.
3. Scroll to the bottom of the **Specify job details** page to find the **Tags - optional** box and click the **Add new tag** button.

**Tags - optional**  
A tag is a label that you can add to a resource as metadata to help you organize, search, or filter your data. Each tag consists of a key and an option value.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

4. Enter information for the **Key** field and, optionally, the **Value** field.

**Tags - optional**  
A tag is a label that you can add to a resource as metadata to help you organize, search, or filter your data. Each tag consists of a key and an option value.

Key	Value - optional
color	blue

Add new tag

You can add up to 49 more tags.

5. Fill in any other fields you wish to include on the **Specify job details** page, then click the **Next** button. This takes you to the **Configure job - optional** page.

Click the **Create job** button to run your transcription job.

6. You can view the tags associated with a transcription job by navigating to the **Transcription jobs** page, selecting a transcription job, and scrolling to the bottom of that job's information page. If you wish to edit your tags, you can do so by clicking the **Manage tags** button.



## AWS CLI

This example uses the [start-transcription-job](#) command and `Tags` parameter. For more information, see [StartTranscriptionJob](#) and [Tag](#).

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

Here's another example using the [start-transcription-job](#) command, and a request body that adds tags to that job.

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

The file *my-first-tagging-job.json* contains the following request body.

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

## AWS SDK for Python (Boto3)

The following example uses the AWS SDK for Python (Boto3) to add a tag by using the `Tags` argument for the [start\\_transcription\\_job](#) method. For more information, see [StartTranscriptionJob](#) and [Tag](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

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

# Improving transcription accuracy

If you want to improve the accuracy of your transcriptions, you can use custom vocabularies, custom language models, or both.

[Custom vocabularies \(p. 64\)](#) involve the input of specific words. For example, if Amazon Transcribe is not correctly rendering terms that are specific to your use case, you can create a vocabulary file with the correct terms for Amazon Transcribe to use in the transcription.

Custom vocabularies do not use context, and instead focus on individual words that you input manually.

[Custom language models \(p. 107\)](#) are more generalized and require the input of large amounts of training and, optionally, tuning data. Although custom language models often require more work upfront, they can be more widely applied once created.

Custom language models also recognize individual terms, but, unlike custom vocabularies, they take into account the context associated with each term when transcribing your audio. This context-specific approach can produce significant accuracy improvements over custom vocabularies alone.

**Tip**

To achieve the highest transcription accuracy with batch transcriptions, use custom vocabularies in conjunction with your custom language models.

## Custom vocabularies

Use custom vocabularies to improve transcription accuracy for one or more specific words. These are generally domain-specific words and phrases, words that Amazon Transcribe isn't recognizing, and proper nouns. We recommend creating separate, small vocabularies tailored to specific audio recordings instead of creating a single, large vocabulary for use with all of your recordings.

**Important**

You are responsible for the integrity of your own data when you use Amazon Transcribe. Do not enter confidential information, personal information (PII), or protected health information (PHI) into a custom vocabulary.

Considerations when creating a custom vocabulary:

- You can have up to 100 vocabularies in your account.
- The size limit for each custom vocabulary is 50 Kb.
- Each entry must contain fewer than 256 characters, including hyphens.
- You can only use characters that are supported for your language. Refer to your language's [character set \(p. 72\)](#) for details.
- Each vocabulary file can be in either table or list format; table format is strongly recommended.
- Your vocabulary files must be stored in an Amazon S3 bucket if using a table format. If using a list, you can upload a text file using the AWS Management Console or include your list of words within an API call.

**Tip**

You can test your vocabulary using the AWS Management Console. Once your vocabulary is ready to use, log into the AWS Management Console, select **Real-time transcription**, scroll to **Customizations**, toggle on **Custom vocabulary**, and select your vocabulary from the drop-

down list. Then select **start streaming**. Speak some of the words in your vocabulary into your microphone to see if they render correctly.

### Should I use a table or a list for my custom vocabulary?

Table format is strongly preferred.

Tables give you more options for—and more control over—the input and output of words within your custom vocabulary. With tables, you must specify multiple categories (Phrase, IPA, SoundsLike, and DisplayAs), allowing you to fine-tune your output. If you have domain-specific or non-standard words, use the table format and add a pronunciation hint in the 'SoundsLike' column.

Lists don't have additional options, so you can only type in entries as you want them to appear in your transcript, replacing any spaces with hyphens. Vocabulary lists are not supported with the [CreateMedicalVocabulary](#) operation.

The AWS Management Console, AWS CLI, and AWS SDKs all use custom vocabulary tables in the same way; lists are used differently for each method and thus may require additional formatting for successful use between each method.

For more information, see [Creating a custom vocabulary using a table \(p. 65\)](#) and [Creating a custom vocabulary using a list \(p. 69\)](#).

## Creating a custom vocabulary using a table

Using a table format is the most robust way to create your custom vocabulary. Vocabulary tables consist of four columns, and these columns can be in any order:

Phrase	SoundsLike	IPA	DisplayAs
Required. Every row in your table must contain an entry in this column.	Optional. Rows in this column can be left empty.	Optional. Rows in this column can be left empty.	Optional. Rows in this column can be left empty.
If your entry contains multiple words, separate each word with a hyphen (-); do not use spaces. For example, <b>Andorra-la-Vella</b> or <b>Los-Angeles</b> .	Break your entry down into hyphen-separated syllables that mimic how the word sounds. For example, <b>Andorra-la-Vella</b> sounds like <b>ann-dor-ah-la-bey-ah</b> and <b>Los-Angeles</b> sounds like <b>los-ann-gel-es</b> .	This column is intended for phonetic spellings using only characters in the <a href="#">International Phonetic Alphabet (IPA)</a> . For example, 'Los Angeles' is <b>l # s æ n # # l # s</b> and 'CLI' is <b>s # # l a#</b> .	Defines the how you want your entry to look in your transcription output. For example, <b>Andorra-la-Vella</b> in the Phrase column is <b>Andorra la Vella</b> in the DisplayAs column.
For acronyms, any pronounced letters must be separated by a period. If your acronym is plural, you must use a hyphen between the acronym and the 's'. For example, 'CLI' is <b>C.I.I.</b> and 'ABCs' is <b>A.B.C.-s</b> .	Do not use spaces in this column.  If you have an entry in this column, your IPA column must be empty.	You must add a single space between every IPA character (single-byte) or valid IPA character pair (double-byte).  If you have an entry in this column, your SoundsLike column must be empty.	If a row in this column is empty, Amazon Transcribe uses the contents of the Phrase column to determine output.
If your phrase consists of both a word and an acronym, these two components must be separated by a			You can use spaces in this column.

Phrase	SoundsLike	IPA	DisplayAs
hyphen. For example, 'DynamoDB' is <b>Dynamo-D.B.</b> .  Do not use spaces in this column.			

Things to note when creating your table:

- Your table must contain all four columns (Phrase, SoundsLike, IPA, and DisplayAs), but the Phrase column is the only one that must contain an entry on each row. All other columns can be left empty.
- In a given row, you cannot have entries for both IPA and SoundsLike fields. You must choose one or the other, or leave both blank.
- You can only use characters that are supported for your language. Refer to your language's [character set \(p. 72\)](#) for details.
- Only use spaces *within* the IPA and DisplayAs columns. Separate columns with a TAB character. Do not use spaces to separate columns; doing so results in an error.
- Only add content in the SoundsLike column if your entry includes a non-standard word, such as a brand name.
- You must save your table as a text (\*.txt) file and upload it into an Amazon S3 bucket before you can convert it into a useable vocabulary.
- Your text file must be in LF format. If you use any other format, such as CRLF, your custom vocabulary is not accepted by Amazon Transcribe.

#### Note

Enter acronyms, or other words whose letters should be pronounced individually, as single letters separated by periods (**A.B.C.**). To enter the plural form of an acronym, such as 'ABCs', separate the 's' from the acronym with a hyphen (**A.B.C.-s**). You can use upper or lower case letters to define an acronym. Acronyms are not supported in all languages; refer to [Supported languages and language-specific features \(p. 3\)](#).

Here is a sample custom vocabulary table (where [TAB] represents a tab character):

```
Phrase[TAB]SoundsLike[TAB]IPA[TAB]DisplayAs
Los-Angeles[TAB][TAB]l # s æ n # # l # s[TAB]Los Angeles
Eva-Maria [TAB]ay-va-ma-ree-ah[TAB][TAB]
A.B.C.-s[TAB]ay-bee-sees[TAB]ABCs
Amazon-dot-com[TAB][TAB][TAB]Amazon.com
C.L.I.[TAB][TAB]s # # l a#[TAB]CLI
Andorra-la-Vella[TAB]ann-do-rah-la-bey-ah[TAB][TAB]Andorra la Vella
Dynamo-D.B.[TAB][TAB]DynamoDB
```

Here is the same table with aligned columns for visual clarity. **Do not** add spaces between columns in your vocabulary table; your table should look misaligned like the preceding example.

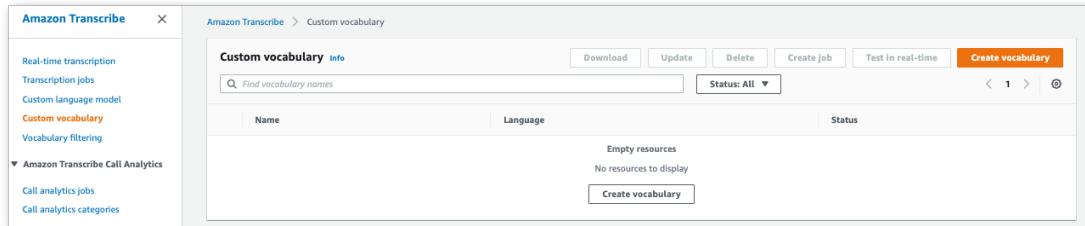
Phrase	[TAB]SoundsLike	[TAB]IPA	[TAB]DisplayAs
Los-Angeles	[TAB]	[TAB]l # s æ n # # l # s[TAB]	Los Angeles
Eva-Maria	[TAB]ay-va-ma-ree-ah	[TAB]	[TAB]
A.B.C.-s	[TAB]ay-bee-sees	[TAB]	[TAB]ABCs
amazon-dot-com	[TAB]	[TAB]	[TAB]amazon.com
C.L.I.	[TAB]	[TAB]s # # l a#	[TAB]CLI
Andorra-la-Vella	[TAB]ann-do-rah-la-bey-ah[TAB]		[TAB]Andorra la Vella
Dynamo-D.B.	[TAB]	[TAB]	[TAB]DynamoDB

To process a custom vocabulary table for use with Amazon Transcribe, see the following for examples:

## AWS Management Console

Before continuing, save your vocabulary as a text (\*.txt) file, then upload it into an Amazon S3 bucket.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Custom vocabulary**. This opens the **Custom vocabulary** page where you can view existing vocabularies or create a new one.
3. Select the **Create vocabulary** button.



This takes you to the **Create vocabulary** page. Enter a name for your new vocabulary.

Select the **S3 location** option under **Vocabulary input source**. Then, either manually enter the Amazon S3 path or select **Browse S3** to locate your vocabulary.

Vocabulary settings	
Name	MyVocabulary01
The name can be up to 200 characters long. Valid characters are a-z, A-Z, and 0-9.	
Language	English, US (en-US)
Vocabulary input source	<a href="#">Info</a>
<input type="radio"/> File upload List format only	
<input checked="" type="radio"/> S3 location Table format only	
Vocabulary file location on S3	
Provide a path to the S3 location where your vocabulary file is stored. To find a path, go to <a href="#">Amazon S3</a>	
<input type="text" value="s3://MyBucketName/VocabularyFileName"/> <span style="border: 1px solid #ccc; padding: 2px 10px; margin-left: 10px;"><a href="#">Browse S3</a></span>	
File format: txt, maximum size 50 KB.	

4. Optionally, add tags to your vocabulary. Once you have all fields completed, click the **Create vocabulary** button at the bottom of the page. This takes you back to the **Custom vocabulary** page where you can view the status of your custom vocabulary. When the status changes from 'Pending' to 'Ready' your vocabulary can be used with a transcription.

Name	Language	Status
my-first-vocabulary-1	English, US (en-US)	Pending

5. If the status changes to 'Failed', click on the name of your vocabulary to go to its information page.

Name	Language	Status
my-first-vocabulary-2	English, US (en-US)	Ready
my-first-vocabulary-1	English, US (en-US)	Failed

There is a **Failure reason** banner at the top of this page that provides information on why your vocabulary failed. Correct the error in your text file and try again.

Name	Status	Language	Modified
my-first-vocabulary-1	Failed	English, US (en-US)	October 13 2021, 10:03 (UTC-07:00)

## AWS CLI

This example uses the [create-vocabulary](#) command with a table-formatted vocabulary file. For more information, see [CreateVocabulary](#).

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

Here's another example using the [create-vocabulary](#) command, and a request body that creates your vocabulary.

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

The file *my-first-vocab-table.json* contains the following request body.

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

Once **VocabularyState** changes from **PENDING** to **READY**, your vocabulary is ready to use with a transcription. To view the current status of your vocabulary, run:

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

## AWS SDK for Python (Boto3)

This example uses the AWS SDK for Python (Boto3) to create a custom vocabulary from a table using the [create\\_vocabulary](#) method. For more information, see [CreateVocabulary](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

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

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

To use an existing custom vocabulary in a transcription job, set the `VocabularyName` in the [Settings](#) field when you call the [StartTranscriptionJob](#) operation or, from the AWS Management Console, choose the vocabulary from the drop-down list.

## Creating a custom vocabulary using a list

You can create custom vocabularies from lists using the AWS Management Console, AWS CLI, or AWS SDKs.

If using the [AWS Management Console](#), you must create a text file for your vocabulary list. You can either place each entry on its own line, or put multiple entries on a single line, separating each entry with a comma. Note that if your entry is a phrase, you must hyphenate each word; for example, 'Los Angeles' looks like **Los-Angeles** and 'Andorra la Vella' looks like **Andorra-la-Vella**.

Here are examples of valid lists:

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

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

If using the AWS CLI or an SDK to create your vocabulary, you must include your entries within the API call using the [Phrases](#) flag.

Things to note when creating your list:

- You can only use characters that are supported for your language. Refer to your language's [character set \(p. 72\)](#) for details.
- If using the AWS Management Console, you must save your list as a text (\*.txt) file in `LF` format. If you use any other format, such as `CRLF`, your custom vocabulary is not accepted by Amazon Transcribe.

**Note**

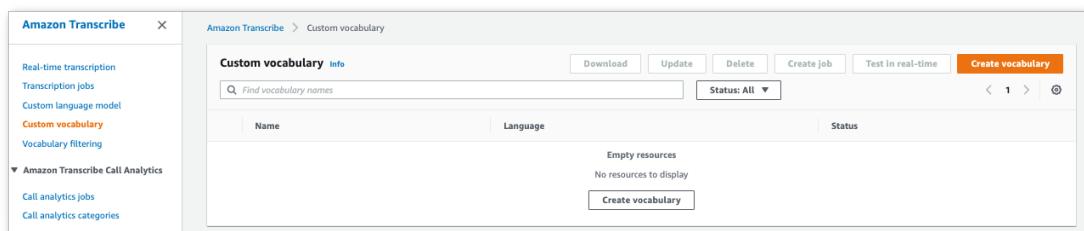
Vocabulary lists are not supported with the [CreateMedicalVocabulary](#) operation.

To process a custom vocabulary list for use with Amazon Transcribe, see the following for examples:

## AWS Management Console

Before continuing, save your vocabulary as a text (\*.txt) file.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Custom vocabulary**. This opens the **Custom vocabulary** page where you can view existing vocabularies or create a new one.
3. Select the **Create vocabulary** button.



This takes you to the **Create vocabulary** page. Enter a name for your new vocabulary.

Select the **File upload** option under **Vocabulary input source**. Then, select **Choose file** to upload your vocabulary.

Vocabulary settings	
Name	MyVocabulary01
The name can be up to 200 characters long. Valid characters are a-z, A-Z, and 0-9.	
Language	English, US (en-US)
Vocabulary input source <a href="#">Info</a>	
<input checked="" type="radio"/> File upload List format only	
<input type="radio"/> S3 location Table format only	
Vocabulary file	
<a href="#">Choose File</a> <small>File format: txt, maximum size 50 KB.</small>	

4. Optionally, add tags to your vocabulary. Once you have all fields completed, click the **Create vocabulary** button at the bottom of the page. This takes you back to the **Custom vocabulary** page where you can view the status of your custom vocabulary. When the status changes from 'Pending' to 'Ready' your vocabulary can be used with a transcription.

Name	Language	Status
my-first-vocabulary-1	English, US (en-US)	Pending

5. If the status changes to 'Failed', click on the name of your vocabulary to go to its information page.

Name	Language	Status
my-first-vocabulary-2	English, US (en-US)	Ready
my-first-vocabulary-1	English, US (en-US)	Failed

There is a **Failure reason** banner at the top of this page that provides information on why your vocabulary failed. Correct the error in your text file and try again.

Name	Status	Language	Modified
my-first-vocabulary-1	Failed	English, US (en-US)	October 13 2021, 10:03 (UTC-07:00)

## AWS CLI

This example uses the [create-vocabulary](#) command with a list-formatted vocabulary file. For more information, see [CreateVocabulary](#).

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

Here's another example using the [create-vocabulary](#) command, and a request body that creates your vocabulary.

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

The file *my-first-vocab-list.json* contains the following request body.

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

Once `VocabularyState` changes from `PENDING` to `READY`, your vocabulary is ready to use with a transcription. To view the current status of your vocabulary, run:

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

## AWS SDK for Python (Boto3)

This example uses the AWS SDK for Python (Boto3) to create a custom vocabulary from a list using the `create_vocabulary` method. For more information, see [CreateVocabulary](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

```
from __future__ import print_function  
import time  
import boto3  
transcribe=boto3.client('transcribe', 'us-west-2')  
job_name = "my-first-vocabulary"  
response = transcribe.create_vocabulary(  
    LanguageCode='en-US',  
    VocabularyName=job_name,  
    Phrases=[  
        'CLI', 'Eva-Maria', 'ABCs'  
    ]  
)  
  
while True:  
    status = transcribe.get_vocabulary(VocabularyName=job_name)  
    if status['VocabularyState'] in ['READY', 'FAILED']:  
        break  
    print("Not ready yet...")  
    time.sleep(5)  
print(status)
```

## Character sets for custom vocabularies

Amazon Transcribe limits the characters that you can use to create custom vocabularies. You can use the following character sets for each language.

### Important

Be sure to check that your custom vocabulary file uses only the supported Unicode code points and code point sequences listed within the following character sets.

Many Unicode characters can appear identical in popular fonts, even if they use different code points. **Only the code points listed in this guide are supported.** For example, the French word `déjà` can be rendered using *precomposed* characters (where one Unicode value represents an accented character) or *decomposed* characters (where two Unicode values represent an accented character, one value for the base character and another for the accent).

- **Precomposed version:** 0064 00E9 006A 00E0 (renders as `déjà`)
- **Decomposed version:** 0064 0065 0301 006A 0061 0300 (renders as `déjà`)

## Topics

- [Afrikaans character set \(p. 73\)](#)
- [Arabic character set \(p. 75\)](#)
- [Chinese, Mandarin \(Mainland China\), Simplified character set \(p. 76\)](#)
- [Chinese, Mandarin \(Taiwan\), Traditional character set \(p. 77\)](#)
- [Danish character set \(p. 78\)](#)
- [Dutch character set \(p. 79\)](#)
- [English character set \(p. 80\)](#)
- [Farsi character set \(p. 81\)](#)
- [French character set \(p. 83\)](#)
- [German character set \(p. 84\)](#)
- [Hebrew character set \(p. 86\)](#)
- [Hindi character set \(p. 87\)](#)
- [Indonesian character set \(p. 89\)](#)
- [Italian character set \(p. 90\)](#)
- [Japanese character set \(p. 92\)](#)
- [Korean character set \(p. 93\)](#)
- [Malay character set \(p. 93\)](#)
- [Portuguese character set \(p. 94\)](#)
- [Russian character set \(p. 96\)](#)
- [Spanish character set \(p. 98\)](#)
- [Tamil character set \(p. 99\)](#)
- [Telugu character set \(p. 101\)](#)
- [Thai character set \(p. 103\)](#)
- [Turkish character set \(p. 105\)](#)

## Afrikaans character set

For Afrikaans custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

Character	Code	Character	Code
á	00E1	í	00EF
è	00E8	ó	00F3
é	00E9	ô	00F4
ê	00EA	ö	00F6
ë	00EB	ú	00FA

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
í	00ED	û	00FB
î	00EE	ü	00FC

You can use the following International Phonetic Alphabet characters in the `IPA` field of the vocabulary input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a	0061	z	007A
b	0062	æ	00E6
c	0063	ð	00F0
d	0064	ø	00F8
e	0065	ŋ	014B
e~	0065 0303	œ	0153
f	0066	ɛ	0250
g	0067	ɑ	0251
h	0068	ã	0251 0303
i	0069	ɔ	0252
j	006A	ɔ	0254
k	006B	ɔ	0254 0303
l	006C	ə	0259
m	006D	ɛ	025B
n	006E	ɛ	025B 0303
o	006F	ɔ	025C
o~	006F 0303	ɪ	026A
p	0070	ɸ	026C
r	0072	ɹ	0279
s	0073	ʂ	0281
t	0074	ʃ	0283
u	0075	ʊ	028A
v	0076	ʌ	028C
w	0077	ɔ	0292
x	0078	θ	03B8

Character	Code	Character	Code
y	0079		

## Arabic character set

For Arabic custom vocabularies, you can use the following Unicode characters in the `Phrase` and `SoundsLike` fields. You can also use the hyphen (-) character to separate words.

Character	Code	Character	Code
ؚ	0621	ؘ	0633
ؙ	0622	ؙ	0634
ؚ	0623	ؚ	0635
ؔ	0624	ؔ	0636
ؖ	0625	ؖ	0637
ؗ	0626	ؗ	0638
ؘ	0627	ؘ	0639
ؙ	0628	ؙ	063A
؜	0629	؜	0641
؝	062A	؝	0642
؞	062B	؞	0643
؟	062C	؟	0644
ؠ	062D	ؠ	0645
آ	062E	آ	0646
ؤ	062F	ؤ	0647
إ	0630	إ	0648
ئ	0631	ئ	0649
ا	0632	ا	064A

You can use the following International Phonetic Alphabet characters in the `IPA` field of the vocabulary input file:

Character	Code	Character	Code
a	0061	ۏ	0074 02E4
a:	0061 02D0	ۏ	0075
b	0062	ۏ	0075 02D0

Character	Code	Character	Code
d	0064	v	0076
d̄	0064 02E4	w	0077
f	0066	x	0078
h	0068	z	007A
i	0069	z̄	007A 02E4
i:	0069 02D0	ð	00F0
j	006A	ð̄	00F0 02E4
k	006B	ḥ	0127
l	006C	χ	0263
m	006D	ı	026A
n	006E	†	026B
p	0070	ſ	0283
q	0071	ȝ	0292
r	0072	?	0294
s	0073	ſ	0295
s̄	0073 02E4	θ	03B8
t	0074	χ	03C7

## Chinese, Mandarin (Mainland China), Simplified character set

For Chinese (Simplified) custom vocabularies, the `Phrase` field can use any of the characters listed in the following file on GitHub.

- [zh-cn-character-set.txt](#)

The `SoundsLike` field can contain the pinyin syllables listed in the following file on GitHub.

- [pinyin-set.txt](#)

When you use pinyin syllables in the `SoundsLike` field, separate the syllables with a hyphen (-).

Amazon Transcribe represents the four tones in Chinese (Simplified) using numbers. The following table shows how tone marks are mapped for the word 'ma'.

Tone	Tone mark	Tone number
Tone 1	mā	ma1
Tone 2	má	ma2

Tone	Tone mark	Tone number
Tone 3	mǎ	ma3
Tone 4	mà	ma4

**Note**

For the 5th (neutral) tone, you can use Tone 1, with the exception of 'er', which must be mapped to Tone 2. For example, 打转儿 would be represented as 'da3-zhuan4-er2'.

Chinese (Simplified) custom vocabularies don't use the `IPA` field, but you must still include the `IPA` header in the vocabulary table.

The following example is an input file in text format. The example uses spaces to align the columns. Your input files should use TAB characters to separate the columns. Include spaces only in the `DisplayAs` column.

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

## Chinese, Mandarin (Taiwan), Traditional character set

For Chinese (Traditional) custom vocabularies, the `Phrase` field can use any of the characters listed in the following file on GitHub.

- [zh-tw-character-set.txt](#)

The `SoundsLike` field can contain the zhuyin syllables listed in the following file on GitHub.

- [zhuyin-set.txt](#)

When you use zhuyin syllables in the `SoundsLike` field, separate the syllables with a hyphen (-).

Amazon Transcribe represents the four tones in Chinese (Traditional) using numbers. The following table shows how tone marks are mapped for the word □ Y.

Tone	Tone mark
Tone 1	ㄇㄚ
Tone 2	ㄇㄚˊ
Tone 3	ㄇㄚˇ
Tone 4	ㄇㄚˋ

Chinese (Traditional) custom vocabularies don't use the `IPA` field, but you must still include the `IPA` header in the vocabulary table.

The following example is an input file in text format. The example uses spaces to align the columns. Your input files should use TAB characters to separate the columns. Include spaces only in the `DisplayAs` column.

Phrase	SoundsLike	IPA	DisplayAs
##	#####-##		
##	###^--##		
####	#####-###-###-##^		
#####	##-#####-##-##^ -##-##	###	

## Danish character set

For Danish custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z
- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

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

You can use the following International Phonetic Alphabet characters in the `IPA` field of the vocabulary input file:

Character	Code	Character	Code
a	0061	æ	00E6
b	0062	ð	00F0
d	0064	ø	00F8
e	0065	ŋ	014B
f	0066	œ	0153
h	0068	ɛ	0250
i	0069	ɑ	0251
j	006A	ɒ	0252
k <sup>h</sup>	006B 02B0	ɔ	0254
l	006C	ç	0255
m	006D	ə	0259
n	006E	ɛ	025B

Character	Code	Character	Code
o	006F	g	0261
p <sup>h</sup>	0070 02B0	I	026A
s	0073	š	0281
t <sup>s</sup>	0074 02E2	ú	028A
u	0075	ú	028B
w	0077	ʌ	028C
y	0079		

## Dutch character set

For Dutch custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

Character	Code	Character	Code
à	00E0	î	00EE
á	00E1	ï	00EF
â	00E2	ñ	00F1
ä	00E4	ò	00F2
ç	00E7	ó	00F3
è	00E8	ô	00F4
é	00E9	ö	00F6
ê	00EA	ù	00F9
ë	00EB	ú	00FA
ì	00EC	û	00FB
í	00ED	ü	00FC

You can use the following International Phonetic Alphabet characters in the `IPA` field of the vocabulary input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a:	0061 003A	ø:	00F8 003A
b:	0062 02D0	ŋ	014B
b	0062	œy	0153 0079
d	0064	œ:	0153 02D0
e:	0065 02D0	ɑ	0251
f	0066	ɔ	0254
g	0067	ɔu	0254 0075
i	0069	ɔ:	0254 02D0
j	006A	ə	0259
k	006B	ɛ	025B
l	006C	ɛ:	025B 003A
m	006D	ɛi	025B 0069
n	006E	ħ	0266
o:	006F 02D0	ɪ	026A
p	0070	ŋ	0272
s	0073	r	027E
t	0074	ʃ	0283
u	0075	ɣ	028F
v	0076	ʒ	0292
w	0077	χ	03C7
y	0079	ɣ	0263
z	007A		

## English character set

For English custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can use the following International Phonetic Alphabet characters in the **IPA** field of the vocabulary input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
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	ə	0251
g	0067	ɔ	0254
h	0068	ɔɪ	0254 026A
i	0069	ə	0259
j	006A	ɛ	025B
k	006B	ʒ	025D
l	006C	g	0261
l̄	006C 0329	ɪ	026A
m	006D	ɹ	0279
n	006E	ʃ	0283
ɳ	006E 0329	ʊ	028A
oʊ	006F 028A	ʌ	028C
p	0070	ʍ	028D
s	0073	ʒ	0292
t	0074	dʒ	02A4
u	0075	tʃ	02A7
v	0076	θ	03B8

## Farsi character set

For Farsi custom vocabularies, you can use the following characters in the **Phrase** and **SoundsLike** fields.

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
س	0621	ش	0638
ی	0622	غ	0639

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
أ	0623	غ	063A
ڦ	0624	ڦ	0641
ڙ	0626	ڦ	0642
ا	0627	ڢ	0644
ٻ	0628	ڻ	0645
ڻ	062A	ڻ	0646
ڻ	062B	ڦ	0647
ڻ	062C	ڦ	0648
ڻ	062D	-	064E
ڻ	062E	ڻ	064F
ڻ	062F	-	0650
ڻ	0630	-	0651
ڻ	0631	ڦ	067E
ڻ	0632	ڦ	0686
ڻ	0633	ڦ	0698
ڻ	0634	ڪ	06A9
ڻ	0635	ڱ	06AF
ڻ	0636	ڦ	06CC
ڻ	0637		

You can use the following International Phonetic Alphabet in the IPA field of your vocabulary file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
b	0062	u	0075
d	0064	v	0076
f	0066	z	007A
g	0067	æ	00E6
h	0068	ɒ	0252
i	0069	ɛ	025B
j	006A	r	027E
k	006B	ݔ	0281
l	006C	݂	0283

Character	Code	Character	Code
m	006D	ȝ	0292
n	006E	? ȝ	0294
o	006F	? ȝ	0294
p	0070	ȝ	02A4
s	0073	ȝ	02A7
t	0074	ȝ	03C7

## French character set

For French custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

Character	Code	Character	Code
À	00C0	à	00E0
Â	00C2	â	00E2
Ҫ	00C7	ҫ	00E7
È	00C8	è	00E8
É	00C9	é	00E9
Ê	00CA	ê	00EA
Ӭ	00CB	ӗ	00EB
Î	00CE	î	00EE
Ӥ	00CF	ӥ	00EF
Ӯ	00D4	Ӯ	00F4
Ӯ	00D6	Ӯ	00F6
ӻ	00D9	ӻ	00F9
ӻ	00DB	ӻ	00FB
ӻ	00DC	ӻ	00FC

You can use the following International Phonetic Alphabet in the `IPA` field of your vocabulary file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a	0061	z	007A
b	0062	ã	00E3
d	0064	ð	00F5
e	0065	ø	00F8
f	0066	ŋ	014B
i	0069	œ	0153
j	006A	œ̃	0153 0303
k	006B	æ	0250
l	006C	ɔ	0254
m	006D	ə	0259
n	006E	ɛ	025B
o	006F	g	0261
p	0070	ɥ	0265
s	0073	ɳ	0272
t	0074	ㅂ	0281
u	0075	ʃ	0283
v	0076	ڙ	0292
w	0077	ڦ	1EBD
y	0079		

## German character set

For German custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
ä	00E4	Ä	00C4
ö	00F6	Ö	00D6
ü	00FC	Ü	00DC
ß	00DF		

You can use the following International Phonetic Alphabet characters in the **IPA** field of the vocabulary input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a	0061	ts	0074 0073
ai	0061 026A	u:	0075 02D0
au	0061 028A	v	0076
a:	0061 02D0	x	0078
b	0062	z	007A
d	0064	y:	0079 02D0
e:	0065 02D0	ã	00E3
f	0066	ç	00E7
g	0067	ø:	00F8 02D0
h	0068	ŋ	014B
i:	0069 02D0	œ	0153
j	006A	æ	0250 032F
k	006B	ɔ	0254
l	006C	ɔʏ	0254 028F
l̄	006C 0329	ə	0259
m	006D	ɛ	025B
m̄	006D 0329	ɛ:	025B 02D0
n	006E	ɪ	026A
n̄	006E 0329	ʊ	0281
o:	006F 02D0	ʃ	0283
p	0070	ʊ	028A
pf	0070 0066	ʏ	028F
s	0073	ʈʃ	02A7

Character	Code	Character	Code
t	0074		

## Hebrew character set

For Hebrew custom vocabularies, you can use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

Character	Code	Character	Code
-	002D	□	05DD
ָ	05D0	ַ	05DE
ְ	05D1	ִ	05DF
ֲ	05D2	ֵ	05E0
ֳ	05D3	ֶ	05E1
ִ	05D4	ַ	05E2
ֵ	05D5	ָ	05E3
ֶ	05D6	ֹ	05E4
ַ	05D7	־	05E5
ָ	05D8	ֻ	05E6
ֹ	05D9	ּ	05E7
ְ	05DA	ֻ	05E8
ֻ	05DB	ּ	05E9
ּ	05DC	ֽ	05EA

You can use the following International Phonetic Alphabet characters in the `IPA` field of the vocabulary input file:

Character	Code	Character	Code
a	0061	p	0070
b	0062	s	0073
d	0064	t	0074
e	0065	u	0075
f	0066	v	0076
g	0067	w	0077
h	0068	z	007A

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
i	0069	ି	014B
j	006A	ୟ	0263
k	006B	ୱ	0283
l	006C	୳	0292
m	006D	୰	0294
n	006E	୯	03C7
o	006F		

## Hindi character set

For Hindi custom vocabularies, you can use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
-	002D	ଥ	0925
.	002E	ଦ	0926
ঁ	0901	ଧ	0927
ঁ	0902	ନ	0928
:	0903	ପ	092A
অ	0905	ଫ	092B
আ	0906	ବ	092C
ই	0907	ଭ	092D
ঁ	0908	ମ	092E
ঁ	0909	ଯ	092F
ঁ	090A	ର	0930
ঁ	090B	ଲ	0932
ঁ	090F	ବ	0935
ঁ	0910	ଶ	0936
ঁ	0911	ଷ	0937
ঁ	0913	ସ	0938
ঁ	0914	ହ	0939
ক	0915	ଠ	093E
খ	0916	ଫ	093F

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
ଗ	0917	ଟ	0940
ଘ	0918	୬	0941
ଡ	0919	୭	0942
ଚ	091A	୮	0943
ଙ୍କ	091B	୯	0945
ଜ	091C	ୱ	0947
ଝ	091D	୳	0948
ଅ	091E	ୣ	0949
ର	091F	୤	094B
ଠ	0920	୥	094C
ଡ	0921	୦	094D
ଢ	0922	୩	095B
ଣ	0923	୪	095C
ତ	0924	୫	095D

Amazon Transcribe maps the following characters:

<b>Character</b>	<b>Mapped to</b>
ନ (0929)	ନ (0928)
ର (0931)	ର (0930)
କ୍ର (0958)	କ୍ର (0915)
ଖ୍ର (0959)	ଖ୍ର (0916)
ଣ୍ଟ (095A)	ଣ୍ଟ (0917)
ଫ୍ର (095E)	ଫ୍ର (092B)
ୟ (095F)	ୟ (092F)

You can use the following International Phonetic Alphabet characters in the IPA field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a:	0097 0720	ଏ	0331
b	0098	ବ	0598
b <sup>h</sup>	0098 0689	ବ୍ହ	0596 0720
d	0100	ଦ	0598 0689

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
d <sup>h</sup>	0100 0689	ə	0601
e:	0101 0720	ɛ:	0603 0720
f	0102	g	0609
i:	0105 0720	g <sup>h</sup>	0609 0689
j	0106	h	0614
k	0107	ɪ	0618
k <sup>h</sup>	0107 0688	ŋ	0626
l	0108	ɳ	0627
m	0109	r	0638
n	0110	ʂ	0642
o:	0111 0720	ʃ	0643
p	0112	t	0648
p <sup>h</sup>	0112 0688	t <sup>h</sup>	0648 0688
r	0114	ʊ	0650
s	0115	ʊ	0651
t	0116	dʒ	0676
t <sup>h</sup>	0116 0688	dʒ <sup>h</sup>	0676 0689
u:	0117 0720	tʃ	0679
z	0122	tʃ <sup>h</sup>	0679 0688

## Indonesian character set

For Indonesian custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can use the following International Phonetic Alphabet characters in the `IPA` field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a	0061	r	0072
ai	0061 0069	s	0073

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
au	0061 0075	t	0074
b	0062	tʃ	0074 0283
d	0064	u	0075
dʒ	0064 0292	v	0076
e	0065	w	0077
f	0066	x	0078
h	0068	y	0079
i	0069	ŋ	014B
j	006A	ɔ	0254
k	006B	ə	0259
l	006C	ɛ	025B
m	006D	g	0261
n	006E	ɣ	0263
o	006F	ɪ	026A
ő	006F 0069 032F	ɲ	0272
p	0070	ʃ	0283
q	0071	ʊ	028A

## Italian character set

For Italian custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
À	00C0	à	00E0
Ä	00C4	ä	00E4
Ç	00C7	ç	00E7
È	00C8	è	00E8

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
É	00C9	é	00E9
Ê	00CA	ê	00EA
Ë	00CB	ë	00EB
Ì	00CC	ì	00EC
Ò	00D2	ò	00F2
Ù	00D9	ù	00F9
Ü	00DC	ü	00FC

You can use the following International Phonetic Alphabet characters in the `IPA` field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a	0061	ss	0073 0073
b	0062	t	0074
bb	0062 0062	tt	0074 0074
d	0064	u	0075
dd	0064 0064	v	0076
e	0065	vv	0076 0076
f	0066	w	0077
ff	0066 0066	z	007A
gg	0067 0067	ɔ	0254
i	0069	ɛ	025B
j	006A	g	0261
k	006B	ŋ	0272
kk	006B 006B	ŋŋ	0272 0272
l	006C	ʃ	0283
ll	006C 006C	ʒ	0283 0283
m	006D	ʌ	028E
mm	006D 006D	ʌʌ	028E 028E
n	006E	dʒ	02A3
nn	006E 006E	dʒdʒ	02A3 02A3
o	006F	dʒ	02A4
p	0070	dʒdʒ	02A4 02A4

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
p <small>p</small>	0070 0070	t <small>ts</small>	02A6
r <small>r</small>	0072	t <small>ts</small> s	02A6 02A6
rr <small>rr</small>	0072 0072	t <small>ʃ</small>	02A7
s <small>s</small>	0073	t <small>ʃʃ</small>	02A7 02A7

## Japanese character set

For Japanese custom vocabularies, the `Phrase` and `SoundsLike` fields can use any of the characters listed in the following file on GitHub.

- [ja-jp-character-set.txt](#)

You can use the following International Phonetic Alphabet characters in the `IPA` field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a	0061	p	0070
a:	0061 02D0	s	0073
b	0062	t	0074
d	0064	ts	0074 0073
dz	0064 007A	tç	0074 0255
dz̥	0064 0291	w	0077
e	0065	z	007A
e:	0065 02D0	ç	00E7
g	0067	ŋ	014B
h	0068	ç	0255
i	0069	ɯ	026F
i:	0069 02D0	ɯ:	026F 02D0
j	006A	ɳ	0274
k	006B	ɸ	0278
m	006D	r	027E
n	006E	z	0291
o	006F	?	0294
o:	006F 02D0		

## Korean character set

For Korean custom vocabularies, you can use any of the Hangul syllables in the `Phrase` and `SoundsLike` fields. For more information, see [Hangul Syllables](#) on Wikipedia.

You can use the following International Phonetic Alphabet characters in the `IPA` field of your input file:

Character	Code	Character	Code
a	0061	s	0073
e	0065	s#	0073 0348
h	0068	t	0074
i	0069	tç	0074 0255
je	006A 0065	tçʰ	0074 0255 02B0
jo	006A 006F	tʰ	0074 02B0
ju	006A 0075	t#	0074 0348
jɛ	006A 025B	ʈ	0074 031A
jʌ	006A 028C	t#ç	0074 0348 0255
ja	006A 0061	u	0075
k	006B	we	0077 0065
kʰ	006B 02B0	wi	0077 0069
k#	006B 0348	wɛ	0077 025B
ጀ	006B 031A	wʌ	0077 028C
l	006C	wa	0077 0061
m	006D	ø	00F8
n	006E	ŋ	0014B
o	006F	ɛ	0025B
p	0070	ɯ	026F
pʰ	0070 02B0	wi	006F 0069
p#	0070 0348	r	027E
ጀ	0070 031A	ʌ	028C

## Malay character set

For Malay custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z

- ' (apostrophe)
- - (hyphen)
- . (period)

You can use the following International Phonetic Alphabet characters in the **IPA** field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
F	0046	r	0072
a	0061	s	0073
ai	0061 0069	t	0074
au	0061 0075	tʃ	0074 0283
b	0062	v	0076
d	0064	w	0077
dʒ	0064 0292	x	0078
e	0065	y	0079
h	0068	ŋ	014B
i	0069	ɔ	0254
j	006A	ə	0259
k	006B	ɛ	025B
l	006C	g	0261
m	006D	ɣ	0263
n	006E	ɪ	026A
o	006F	ɲ	0272
ɔ̃	006F 0069 32F	ʃ	0283
p	0070	ʊ	028A
q	0071	ʊi	028A 0069

## Portuguese character set

For Portuguese custom vocabularies, you can use the following characters in the **Phrase** and **SoundsLike** fields:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
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
Ö	00D6	ö	00F6
Ú	00DA	ú	00FA
Ü	00DC	ü	00FC

You can use the following International Phonetic Alphabet characters in the `IPA` field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a	0061	s	0073
b	0062	t	0074
d	0064	tʃ	0074 0283
e	0065	u	0075
ě	1EBD	ũ	00169
f	0066	v	0076
g	0067	w	0077
i	0069	w~	0077 0303
ĩ	00129	z	007A

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
j	006A	ѣ	0250 0303
҃	006A 0303	Ԃ	0254
k	006B	Ԃ	025B
l	006C	Ԃ	0272
m	006D	Ր	027E
n	006E	Կ	0281
o	006F	Ժ	0283
õ	00F5	Ղ	028E
p	0070	Յ	0292
r	0072	Ճ	02A4

## Russian character set

For Russian custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
'	0027	҃	043F
-	002D	ր	0440
.	002E	Ԃ	0441
ա	0430	ր	0442
բ	0431	յ	0443
վ	0432	Փ	0444
Շ	0433	Խ	0445
Շ	0434	Ծ	0446
ե	0435	Չ	0447
ժ	0436	Շ	0448
զ	0437	թ	0449
ի	0438	՚	044A
յ	0439	՚	044B
կ	043A	՚	044C
լ	043B	՚	044D
մ	043C	յ	044E

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
H	043D	я	044F
о	043E	ë	0451

You can use the following International Phonetic Alphabet characters in the IPA field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
b	0062	t	0074
b'	0062 02B2	tʃ	0074 0283
d	0064	t̄	0074 02B2
d'	0064 02B2	u	0075
f	0066	v	0076
f'	0066 02B2	v̄	0076 02B2
g	0067	x	0078
g'	067 02B2	x̄	0078 02B2
i	0069	z	007A
j	006A	z̄	007A 02B2
k	006B	æ	00E6
k'	006B 02B2	ə	0259
l	006C	ɛ	025B
l'	006C 02B2	ɪ	0268
m	006D	ʃ	0283
m'	006D 02B2	ʒ	0283 02B2
n	006E	ʊ	028A
n'	006E 02B2	ʌ	028C
p	0070	ɜ	0292
p'	0070 02B2	'i	02C8 0069
r	0072	'o	02C8 006F
r'	0072 02B2	'v	02C8 0075
s	0073	'ɛ	02C8 025B
s'	0073 02B2	't̄	02C8 0268
ts	0074 0073	'a	02C8 0061

## Spanish character set

For Spanish custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

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

You can use the following International Phonetic Alphabet characters in the `IPA` field of your input file:

Character	Code	Character	Code
a	0061	r	0072
b	0062	s	0073
d	0064	t	0074
e	0065	u	0075
f	0066	v	0076
g	0067	w	0077
h	0068	x	0078
i	0069	z	007A
j	006A	ŋ	014B
k	006B	ɳ	0272
l	006C	r	027E
m	006D	ʃ	0283

Character	Code	Character	Code
n	006E	ᬁ	029D
o	006F	ᬁ	02A7
p	0070	θ	03B8

## Tamil character set

For Tamil custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

Character	Code	Character	Code
அ	0B85	ர	0BB0
ஆ	0B86	ல	0BB2
இ	0B87	வ	0BB5
ஏ	0B88	ய	0BB4
உ	0B89	ஞ	0BB3
ஊ	0B8A	ங	0BB1
ஏ	0B8E	ன	0BA9
ஏ	0B8F	ஃ	0B9C
ஃ	0B90	#	0BB6
ஃ	0B92	ஷ	0BB7
ஃ	0B93	ஸ	0BB8
ஓள்	0B94	ஹ	0BB9
ஃ	0B83	.	0BCD
க	0B95	ஈ	0BBE
ங	0B99	ஞ	0BBF
ங	0B9A	ஏ	0BC0
ங	0B9E	ஔ	0BC1
உ	0B9F	க	0BC2
ண்ட	0BA3	ட	0BC6
ங	0BA4	ங	0BC7
ங	0BA8	ண	0BC8
உ	0BAA	ஞ	0BCA
உ	0BAE	ஞ	0BCB

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
w	0BAF	ଶ୍ରୀ	0BCC

You can use the following International Phonetic Alphabet characters in the IPA field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a	0061	v	0076
a:	0061 02D0	w	0077
b	0062	z	007A
d	0064	æ	00E6
dʒ	0064 0292	ð	00F0
e	0065	ŋ	014B
f	0066	ɑ	0251
g	0067	ɔ	0254
h	0068	ə	0259
i	0069	ɛ	025B
i:	0069 02D0	g	0261
j	006A	ɪ	026A
k	006B	l	026D
l	006C	ɲ	0272
m	006D	ɳ	0273
n	006E	ʐ	0279
ɳ	006E 032A	ʐ	0279
o	006F	ɸ	0279 0329
o:	006F 02D0	r	027E
p	0070	ʂ	0282
r	0072	ʃ	0283
s	0073	t	0288
t	0074	ʊ	028A
ʈ	0074 032A	ʊ	028B
ʈʃ	0074 0283	ʌ	028C
u	0075	ڙ	0292
u:	0075 02D0	θ	03B8

## Telugu character set

For Telugu custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

Character	Code	Character	Code
-	002D	ం	0C24
ఁ	0C01	ఁ	0C25
ం	0C02	ం	0C26
ే	0C03	ఁ	0C27
ో	0C05	న	0C28
ౌ	0C06	స	0C2A
్య	0C07	శ	0C2B
ఁఁ	0C08	బ	0C2C
ఁఁ	0C09	ఁ	0C2D
ఁఁఁ	0C0A	ప	0C2E
ఁఁఁ	0C0B	య	0C2F
ఁ	0C0C	ఱ	0C30
ఁ	0C0E	ణ	0C31
ఁ	0C0F	ల	0C32
ఁ	0C10	ఁ	0C33
ఁ	0C12	ఁ	0C35
ఁ	0C13	ఁ	0C36
ఁ	0C14	స	0C37
ఁ	0C15	స	0C38
ఁ	0C16	స్	0C39
ఁ	0C17	ఠ	0C3E
ఁ	0C18	ఱ	0C3F
ఁ	0C19	ఁ	0C40
ఁ	0C1A	ఁ	0C41
ఁ	0C1B	ఱ	0C42
ఁ	0C1C	ఁ	0C43
ఁ	0C1D	ఱ	0C44
ఁ	0C1E	ఁ	0C47

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
é	0C1F	í	0C48
ó	0C20	ú	0C4A
ú	0C21	íó	0C4B
úó	0C22	íú	0C4C
éó	0C23	éú	0C4D

You can use the following International Phonetic Alphabet characters in the **IPA** field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
đ	0064 032A	ð	00F0
đ	0064 032A 0324	ŋ	014B
dʒ	0064 0292	ɑ	0251
dʒ	0064 0292 0324	ɔ	0254
e	0065	ɖ	0256
e:	0065 02D0	ɖ	0256 0324
f	0066	ə	0259
h	0068	ɛ	025B
i	0069	g	0261
iz	0069 0290	g	0261 0324
j	006A	ɪ	026A
k	006B	l	026D
kʰ	006B 02B0	ɲ	0272
l	006C	ɳ	0273
m	006D	ɹ	0279
n	006E	ɻ	0279 0329
o	006F	ɾ	027D
o:	006F 02D0	ʂ	0282
p	0070	ʃ	0283
pʰ	0070 02B0	ʈ	0288
r	0072	ʈʰ	0288 02B0
s	0073	ڻ	028A
t	0074	ڻ	028B

Character	Code	Character	Code
҂	0074 032A	߈	028C
҃	0074 032A 02B0	߃	0292
߄	0074	߅	03B8
߆	0074 0283 02B0		

## Thai character set

For Thai custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

Character	Code	Character	Code
ໜ	0E01	ລ	0E25
ໝ	0E02	ກ	0E26
ໝ	0E03	ຈ	0E27
ໝ	0E04	ສ	0E28
ໝ	0E05	ໝ	0E29
ໝ	0E06	ສ	0E2A
ໝ	0E07	ໜ	0E2B
ໝ	0E08	ໜ	0E2C
ໝ	0E09	ອ	0E2D
ໝ	0E0A	ໝ	0E2E
ໝ	0E0B	ໍ	0E2F
ໝ	0E0C	໌	0E30
ໝ	0E0D	໌	0E31
ໝ	0E0E	ໍ	0E32
ໝ	0E0F	໌	0E34
ໝ	0E10	໌	0E35
ໝ	0E11	໌	0E36
ໝ	0E12	໌	0E37
ໝ	0E13	,	0E38

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
ର	0E14	ୟ	0E39
ତ	0E15	୷	0E3A
ଳ	0E16	୬	0E40
ଥ	0E17	୭	0E41
ଙ	0E18	୮	0E42
ମ	0E19	୯	0E43
ପ	0E1A	ୱ	0E44
ଜ	0E1B	୲	0E45
ଫ	0E1C	୳	0E46
ବ	0E1D	୵	0E47
ମ	0E1E	୶	0E48
ଳ	0E1F	୷	0E49
ଙ	0E20	୹	0E4A
ମ	0E21	୺	0E4B
ପ	0E22	୻	0E4C
ବ	0E23	୻	0E4D
ଳ	0E24		

You can use the following International Phonetic Alphabet characters in the IPA field of your input file:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
a	0061	p <sup>h</sup>	0070 02B0
a:	0061 02D0	r	0072
b	0062	s	0073
d	0064	t	0074
e	0065	t <sup>h</sup>	0074 02B0
e:	0065 02D0	t̚ç	0074 0361 0255
f	0066	t̚ç <sup>h</sup>	0074 0361 0255 02B0
h	0068	u	0075
i	0069	u:	0075 02D0
i:	0069 02D0	u:a	0075 02D0 0061
i:a	0069 02D0 0061	w	0077

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
j	006A	ŋ	014B
k	006B	ɔ	0254
kw	006B 0077	ɔ:	0254 02D0
k <sup>h</sup>	006B 02B0	ɛ	025B
k <sup>h</sup> w	006B 02B0 0077	ɛ:	025B 02D0
l	006C	ɣ	0264
m	006D	ɣ:	0264 02D0
n	006E	ɯ	026F
o	006F	ɯ:	026F 02D0
o:	006F 02D0	ɯ:a	026F 02D0 0061
p	0070	?	0294

## Turkish character set

For Turkish custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` fields:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` and `SoundsLike` fields:

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
ç	00C7	ö	00F6
Ö	00D6	û	00FB
Ü	00DC	ü	00FC
â	00E2	Ğ	011E
ä	00E4	ğ	011F
ç	00E7	ı	0130
è	00E8	ı	0131
é	00E9	ş	015E
ê	00EA	ş	015F
í	00ED	ſ	0161

Character	Code	Character	Code
î	00EE	ž	017E
ó	00F3		

You can use the following International Phonetic Alphabet characters in the **IPA** field of your input file:

Character	Code	Character	Code
a	0061	u	0075
a:	0061 02D0	u:	0075 02D0
b	0062	v	0076
c	0063	w	0077
d	0064	y	0079
e	0065	y:	0079 02D0
e:	0065 02D0	z	007A
f	0066	ø	00F8
g	0067	ø:	00F8 02D0
h	0068	ŋ	014B
i	0069	ʃ	025F
i:	0069 02D0	χ	0263
j	006A	t̪	026B
k	006B	w	026F
l	006C	w:	026F 02D0
m	006D	r̪	027E
n	006E	ʃ̪	0283
o	006F	ʒ̪	0292
o:	006F 02D0	?	0294
p	0070	dʒ̪	02A4
s	0073	tʃ̪	02A7
t	0074		

For a video demo showing how to create a custom vocabulary—in both list and table form—using the AWS Management Console, see [Using a custom vocabulary](#).

For a real-life example of how Formula 1 is using custom vocabularies to create live captions for their races, see [Live transcriptions of F1 races using Amazon Transcribe](#).

### API operations specific to custom vocabularies

[CreateVocabulary](#), [DeleteVocabulary](#), [GetVocabulary](#), [ListVocabularies](#), [UpdateVocabulary](#)

## Custom language models

Custom language models use your text data (training data) to improve transcription accuracy for your specific use case. For example, you can provide Amazon Transcribe with industry-specific terms or acronyms that it might not otherwise recognize.

In order to produce accurate transcriptions, your text data must be representative of the audio you want to transcribe. Domain-specific text data can include website content, instruction manuals, technical documentation, and audio transcripts. The text data you provide must be related to your use case and, if using transcripts, these must be accurate. Using inaccurate data to train language models results in inaccurate models, which in turn affects the accuracy of any transcription results that use those models.

The quality of your training data is much more important than the quantity of data when creating a custom language model. You can provide up to 2 GB of training data and 200 MB of tuning data.

There are two ways to upload your text data to create a custom language model:

1. Upload your text as *training data*, which is used to train your custom language model for your specific use case.
2. Upload your text as *tuning data*, which is used to optimize your custom language model and increase its transcription accuracy.

If you're unsure on whether to use your data for training or tuning, this table may help you decide:

If you have	Upload this
A large amount of domain-specific text and a much smaller amount of audio transcript text data	Domain-specific text as training data. Upload your transcription text as tuning data.
A minimum of 10,000 words of audio transcript text	Audio transcript text as training data.
At least 100,000 words of audio transcript text and a large amount of additional domain-specific text	Audio transcript text as training data. Typically, this method leads to the greatest increase in transcription accuracy. If this method doesn't produce the desired increase in transcription accuracy, follow the first method described in this table.
Domain-specific text only	Domain-specific text as training data. This is the least robust option for training your model.

Custom language models are not available with all Amazon Transcribe-supported languages; refer to [Supported languages and language-specific features \(p. 3\)](#) for more information.

For a video walkthrough of creating and using custom language models, see [Using Custom Language Models \(CLM\) to supercharge transcription accuracy](#).

This video is part of the blog post: [Boost transcription accuracy of class lectures with custom language models for Amazon Transcribe](#).

### API operations specific to custom language models

[CreateLanguageModel](#), [DeleteLanguageModel](#), [DescribeLanguageModel](#), [ListLanguageModels](#)

When you're ready to create your own custom language model, refer to the [following section \(p. 108\)](#).

## Creating a custom language model

The steps involved in creating and using a custom language model are:

- [Step 1: Preparing your data \(p. 108\)](#)
- [Step 2: Providing Amazon Transcribe with data permissions \(p. 109\)](#)
- [Step 3: Creating a custom language model \(p. 110\)](#)
- [Step 4: Transcribing with a custom language model \(p. 113\)](#)
- [Step 5: Viewing and updating your custom language models \(p. 117\)](#)

### Step 1: Preparing your data

Create a custom language model by providing training data in plain text format and by choosing a base model. You can also provide additional tuning data, also in plain text format, for optimization.

To prepare your text data:

1. Properly format it and save it in one or more text files. Make sure each text file:

- Is in the same language as the model you want to create. For example, if you want to create a custom language model that transcribes audio in US English (en-US), your text data must also be in US English (en-US).
- Is in plain text (it's not a file such as a Microsoft Word document, comma-separated value file, or PDF).
- Is encoded in UTF-8.
- Doesn't contain any formatting characters, such as HTML tags.
- Is less than 2 GB in size if you intend to use the file as training data. Amazon Transcribe only accepts a maximum of 2 GB of training data.
- Is less than 200 MB in size if you intend to use the file as tuning data. You can provide a maximum of 200 MB of optional tuning data.

#### Note

Although not a requirement, it is best practice to have only one sentence per line within your text file.

2. Upload those files to Amazon S3. If you intend to tune the model, store your tuning data in a separate Amazon S3 bucket from the one you use for your training data.

Use your own data processing pipelines to prepare your plain text files. If you're extracting text from HTML, remove the HTML tags and unescape the entities.

The following example shows how to format sentences in a text file:

```
Ribosomes help translate RNA into protein.  
RISC is essential in RNA interference.  
Interferon type 1 signaling proteins help prevent viruses from replicating their RNA or  
DNA.  
...
```

It doesn't matter how many text files you use to upload your training or tuning data. For model training, you get the same improvements in transcription accuracy if you use one file with 100,000 words or 10 files with 10,000 words. Prepare your text data in a way that's most convenient for you.

#### Next step

[Step 2: Providing Amazon Transcribe with data permissions \(p. 109\)](#)

## Step 2: Providing Amazon Transcribe with data permissions

To create a custom language model, you need to give Amazon Transcribe access to your text data. To do this, give Amazon Transcribe the `GetObject` and `ListBucket` permissions to your Amazon S3 bucket.

#### To provide data permissions

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, for **Access management**, choose **Roles** and choose one of the following options.
  - Choose the name of the IAM role to use to create your custom language model and run transcription jobs.
  - To create a new IAM role, choose **Create role**.
    - a. Under **Common use cases**, choose **EC2**. You can select any use case, but EC2 is one of the most straightforward ones.
    - b. Choose **Next: Tags**.
    - c. Choose **Next: Review**.
    - d. Specify a name for the role under **Role name**. Remove the text under **Role description**.
    - e. Choose **Create role**.
    - f. Choose the role you've created.
3. Choose **Trust relationships**.
4. Choose **Edit trust relationship**.
5. If you're using an existing role, modify your trust policy with the following code. If you've created a new role using this procedure, replace the trust policy text with the following code.

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

6. In the navigation pane, choose **Policies**.
7. • From the policy list, choose **AmazonTranscribeFullAccess**.
  - a. From **Policy actions**, choose **Attach policy**

- b. Choose the IAM role you want to attach the policy to.
- Choose **Create policy**.
  - a. Choose **JSON**.
  - b. Enter the following code, which adds `GetObject` and `ListBucket` permissions:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "s3:GetObject"  
            ],  
            "Resource": [  
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"  
            ],  
            "Effect": "Allow"  
        },  
        {  
            "Action": [  
                "s3>ListBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET"  
            ],  
            "Effect": "Allow"  
        }  
    ]  
}
```

- c. Choose **Review policy**.
- d. Specify a name for the policy.
- e. Choose **Create policy**.
- f. Attach the policy to the IAM role.

Amazon Transcribe now has the necessary permissions to access the data for your custom language model.

#### Next step

[Step 3: Creating a custom language model \(p. 110\)](#)

## Step 3: Creating a custom language model

To create a custom language model, you must specify a base model and provide text data using an Amazon S3 prefix.

There are two base model options:

- **NarrowBand** - Use this option for audio with a sample rate of less than 16,000 Hz. This model type is typically used for telephone conversations recorded at 8,000 Hz.
- **WideBand** - Use this option for audio with a sample rate greater than or equal to 16,000 Hz.

You use prefixes in Amazon S3 to specify training data and, optionally, tuning data. Here are some Amazon S3 concepts to be aware of before proceeding:

- Object – The entity stored in your Amazon S3 bucket. In this case, the object is your training or tuning text file.
- Bucket – A container where you store your objects.
- Key – The unique identifier for an object within a bucket; for example, a file path including the file name and extension.
- Prefix – Any portion of the key preceding the file name. You use prefixes to organize your data and specify objects in your Amazon S3 bucket.

So, a key might look like this: s3://DOC-EXAMPLE-BUCKET/clm/creation-date/clm-files/my-clm-file.txt. The prefixes for this key could be any of: s3://DOC-EXAMPLE-BUCKET/clm/, s3://DOC-EXAMPLE-BUCKET/clm/creation-date/, or s3://DOC-EXAMPLE-BUCKET/clm/creation-date/clm-files/. For more information on prefixes, see [Organizing objects using prefixes](#).

You specify prefixes for your text data in the following fields:

- `S3Uri` for your training data
- `TuningDataS3Uri` for your tuning data—this field is optional

#### Note

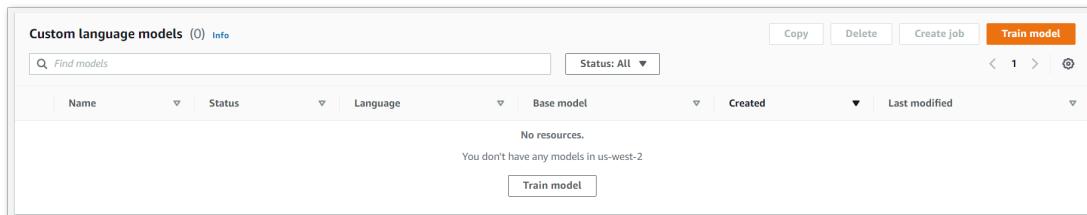
Amazon S3 paths for `S3Uri` and `TuningDataS3Uri` must be the location of the folder in which your text files are located. For example, `S3Uri=s3://DOC-EXAMPLE-BUCKET/clm-training-data/` and `TuningDataS3Uri=s3://DOC-EXAMPLE-BUCKET/clm-tuning-data/`.

Amazon Transcribe uses any object whose key matches one of the prefixes you specify in your CLM request. Amazon Transcribe returns an error for any file that matches a prefix and is not a plain text file.

The following examples show how to create a CLM.

#### AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Custom language model**. This opens the **Custom language models** page where you can view existing language models or train a new model.
3. To train a new model, select the **Train model** button.



This takes you to the **Train model** page where you can add specifications for your model, such as name, base model, training data, and IAM permissions.

**Train model** Info

**Model settings**

**Name**  
MyModel  
The name can be up to 200 characters long. Valid characters: A-Z, a-z, 0-9, and \_ (hyphen).

**Language**  
Choose the language of your model.  
English, US (en-US) ▾

**Base model** Info  
Choose the base model that you want to use to create your custom language model. Choose the model based on the sample rate of your source audio.

- Narrow band**  
For audio that has a sample rate less than 16 KHz. Typically, this is 8 KHz audio from telephone conversations.
- Wide band**  
For audio that has a sample rate of 16 KHz or greater. Typically, this is 16 KHz audio from media sources.

**Training data** Info

**Training data location on S3**  
Type or paste the S3 prefix for the text files that you want to use as training data, or browse to find the files that have matching S3 prefixes.  
s3://MyBucketName/folder Browse S3

The file format must be plain text in the language that you have selected for the model. The maximum file size is 2 GB.

**Tuning data - optional** Info

**Tuning data location on S3**  
Type or paste the S3 prefix for the text files that you want to use as tuning data, or browse to find the files that have matching S3 prefixes.  
s3://MyBucketName/folder Browse S3

The file format must be plain text in the language that you have selected for the model. The maximum file size is 200 MB.

**Access permissions**

**IAM role** Info  
 Use an existing IAM role  
 Create an IAM role  
By choosing Train model you are authorizing creation of this role.

**Role name**  
A role that grants access to the S3 input locations.  
Choose a role ▾

- Once you have all fields completed, click the **Train model** button at the bottom of the page.

## AWS CLI

This example uses the `create-language-model` command. For more information, see [CreateLanguageModel](#) and [LanguageModel](#).

```
aws transcribe create-language-model \
--base-model-name NarrowBand \
--model-name your-language-model-name \
--input-data-config S3Uri=s3://DOC-EXAMPLE-BUCKET,TuningDataS3Uri=s3://DOC-EXAMPLE-BUCKET/S3-prefix/your-filename.txt,DataAccessRoleArn=arn:aws:iam::aws-account-number:role/IAM-role \
```

```
--language-code en-US
```

Here's another example using the [create-language-model](#) command, and a request body that creates your CLM.

```
aws transcribe create-language-model \
--cli-input-json file://filepath/my-first-language-model.json
```

The file *my-first-language-model.json* contains the following request body.

```
{
    "BaseModelName": "your-base-model-name",
    "ModelName": "your-language-model-name",
    "InputDataConfig": {
        "S3Uri": "s3://DOC-EXAMPLE-BUCKET",
        "TuningDataS3Uri": "s3://DOC-EXAMPLE-BUCKET/S3-prefix/your-filename.txt",
        "DataAccessRoleArn": "arn:aws:iam::aws-account-number:role/IAM-role"
    },
    "LanguageCode": "en-US"
}
```

### AWS SDK for Python (Boto3)

This example uses the AWS SDK for Python (Boto3) to create a CLM using the [create\\_language\\_model](#) method. For more information, see [CreateLanguageModel](#) and [LanguageModel](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
model_name = 'my-first-language-model',
transcribe.create_language_model(
    LanguageCode='en-US',
    BaseModelName = 'NarrowBand',
    ModelName = model_name,
    InputDataConfig = {
        'S3Uri': 's3://DOC-EXAMPLE-BUCKET/my-clm-training-data/',
        'TuningDataS3Uri': 's3://DOC-EXAMPLE-BUCKET/my-clm-tuning-data/',
        'DataAccessRoleArn': 'arn:aws:iam::111122223333:role/ExampleRole'
    }
)
while True:
    status = transcribe.get_language_model(ModelName=model_name)
    if status['LanguageModel']['ModelStatus'] in ['COMPLETED', 'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

### Next step

[Step 4: Transcribing with a custom language model \(p. 113\)](#)

## Step 4: Transcribing with a custom language model

You can use a CLM to transcribe your audio using either batch transcription jobs or real-time streams.

## Using a custom language model with a batch transcription job

You can transcribe with a CLM for batch transcription jobs using the **AWS Management Console**, **AWS CLI**, or **AWS SDK**; see the following instructions.

### AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Transcription jobs**, then select the **Create job** button (top right). This opens the **Specify job details** page.
3. In the **Job settings** panel under **Model type**, select the **Custom language model** box.

The screenshot shows the 'Job settings' section of the 'Specify job details' page. It includes fields for 'Name' (set to 'MyTranscriptionJob'), 'Model type' (set to 'Custom language model'), 'Language' (set to 'English, US (en-US)'), and 'Custom model selection' (with a link to 'Create a new one'). A 'Additional settings' section is also visible.

**Job settings**

Name  
MyTranscriptionJob

The name can be up to 200 characters long. Valid characters are a-z, A-Z, 0-9, . (period), \_ (underscore), and – (hyphen).

Model type [Info](#)  
Choose the type of model to use for the transcription job.

General model  
To use a model that is not specialized for a particular use case, choose this option. Configuration options vary between languages.

Custom language model  
To use a model that you trained for your specific use case, choose this option. This model has fewer configuration options than the general model.

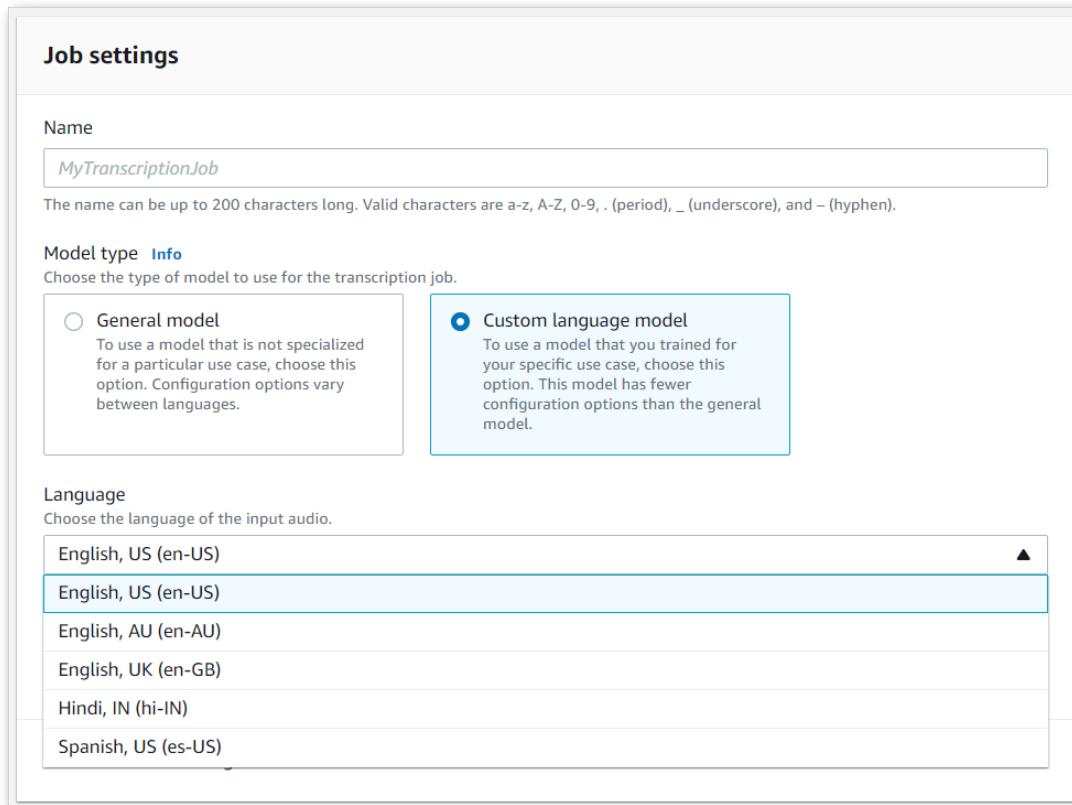
Language  
Choose the language of the input audio.  
English, US (en-US)

Custom model selection  
Choose an existing model or [create a new one](#).

Choose model

► Additional settings

You must also select an input language from the drop-down menu.



4. Under **Custom model selection**, either select an existing custom language model from the drop-down menu or click on 'create a new one'.
- Add the Amazon S3 location of your input file in the **Input data** panel.
5. Click the **Next** button for additional configuration options. Click the **Create job** button to run your transcription job.

## AWS CLI

This example uses the `start-transcription-job` command and `LanguageModelName` parameter. For more information, see [StartTranscriptionJob](#) and [ModelSettings](#).

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

Here's another example using the `start-transcription-job` command, and a request body that uses your CLM with that job.

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

The file `my-first-language-model-job.json` contains the following request body.

```
{
    "TranscriptionJobName": "my-first-transcription-job",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
    },
    "OutputBucketName": "DOC-EXAMPLE-BUCKET",
    "LanguageCode": "en-US",
    "ModelSettings": {
        "LanguageModelName": "my-first-language-model"
    }
}
```

## AWS SDK for Python (Boto3)

This example uses the AWS SDK for Python (Boto3) to transcribe a file with the `LanguageModelName` argument for the `start_transcription_job` method. For more information, see [StartTranscriptionJob](#) and [ModelSettings](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

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

## Using a custom language model with a real-time stream

You can use a CLM to transcribe a real-time stream with either a WebSocket request or the `StartStreamTranscription` operation.

### Note

Only CLMs created after streaming support are available for use with streaming transcriptions. If your CLM was created prior to streaming support, you must re-create it to use it with a real-time stream. To view supported languages for streaming CLMs, see [Supported languages and language-specific features \(p. 3\)](#).

### WebSocket

This example creates a pre-signed URL that uses a custom language model in a WebSocket stream. Line breaks have been added for readability. For more information on using WebSocket streams with

Amazon Transcribe, see [Setting up a WebSocket stream \(p. 54\)](#). For more detail on parameters, see [StartStreamTranscription](#).

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-websocket?  
&X-Amz-Algorithm=AWS4-HMAC-SHA256  
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-west-2%2Ftranscribe%2Faws4_request  
&X-Amz-Date=20220208T235959Z  
&X-Amz-Expires=250  
&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
```

Parameter definitions can be found in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

### HTTP/2 stream

This example creates an HTTP/2 request with a custom language model. For more information on using HTTP/2 streaming with Amazon Transcribe, see [Setting up an HTTP/2 stream \(p. 50\)](#). For more detail on parameters and headers specific to Amazon Transcribe, see [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-language-model-name: my-first-language-model  
transfer-encoding: chunked
```

Parameter definitions can be found in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

### Next step

[Step 5: Viewing and updating your custom language models \(p. 117\)](#)

## Step 5: Viewing and updating your custom language models

The speech recognition capabilities of Amazon Transcribe are constantly improving. Specifically, Amazon Transcribe continually updates the base models used in custom language models.

To see if you're running the latest base model in your custom language model:

1. Use the [DescribeLanguageModel](#) API and check the `UpgradeAvailability` field.
2. If `UpgradeAvailability` is `true`, you're not running the latest version of the base model in your custom language model.

To use the latest base model in a custom language model, you must create a new custom language model. That custom language model will have the updated base model.

# Using vocabulary filtering with unwanted words

Mask, remove, or tag words you don't want in your transcription results using *vocabulary filtering*.

A common use case is the removal of offensive or profane terms from your transcripts; however, vocabulary filters are completely custom, so you can select any words. For example, if you have a new product about to launch, you can mask the product name in meeting transcripts. In this case, you keep stakeholders up-to-date while still keeping the product name secret until launch.

Vocabulary filtering has three different display options:

- **Masked content** — Replaces unwanted words with three asterisks (\*\*\*)�.
- **Content removal** — Content removal: Deletes unwanted words, leaving nothing in their place.
- **Tags** — Adds a tag to each word in your transcription output that matches a words in your vocabulary filter. Tagging is typically used when you want to remove words from select transcripts. For example, if you're creating subtitles for a movie, you may want to leave profanity in for adult screenings, but remove it for family-friendly screenings.

To filter unwanted words, you:

1. Create a list of unwanted words.
2. Create a vocabulary filter.
3. Start your real-time stream or transcription job and specify your vocabulary filter and method. The method (mask, remove, or tag) indicates how you want your transcription output to display filtered words in your transcript.

## API operations specific to vocabulary filtering

[CreateVocabularyFilter](#), [DeleteVocabularyFilter](#), [GetVocabularyFilter](#), [ListVocabularyFilters](#), [UpdateVocabularyFilter](#)

For a video walkthrough of vocabulary filtering, see [Using vocabulary filters](#).

To dive a little deeper and learn how to use Amazon Augmented AI with custom vocabularies, see [Start building a human review along with Amazon Transcribe](#)

## Step 1: Creating a list of unwanted words

To create a vocabulary filter, you can create a list of words to filter from your transcription results and save them in a text file. Or, you can use the [CreateVocabularyFilter](#) API and enter the words that you want to filter as an array of strings in the `Words` parameter. Although listing unwanted words in the [CreateVocabularyFilter](#) API is more convenient, if you use a text file, you can edit your word list later and reuse it in another vocabulary filter.

The following guidelines apply to vocabulary filters:

- Words in a vocabulary filter aren't case sensitive. For example, "curse" and "CURSE" are considered the same word.
- Amazon Transcribe filters only words that exactly match words in the filter. For example, if your filter includes "swear," Amazon Transcribe filters "swear," but not "swears." You must provide every variation of a word that you want to filter.

- Amazon Transcribe doesn't filter words that are contained in other words. For example, if a vocabulary filter contained "marine," but not "submarine," "submarine" would appear in your transcription results.

To create a word list with the console, complete the following procedure. To use the [CreateVocabularyFilter API](#), see [Step 2: Creating a vocabulary filter \(p. 119\)](#).

#### To create a list of unfiltered words (AWS Management Console)

1. In a text editor, create a new file and enter each word on a separate line followed by the newline (\n) as shown in the following example.

```
profanity
curse
swear
...
obscenity
```

2. Save the list as a plain text file locally or in Amazon S3.

#### Next step

[Step 2: Creating a vocabulary filter \(p. 119\)](#)

## Step 2: Creating a vocabulary filter

You can create a vocabulary filter with either the [CreateVocabularyFilter API](#) or the [Amazon Transcribe console](#).

If you use the [CreateVocabularyFilter API](#), you can enter the words in your vocabulary filter as an array of strings into the `Words` parameter. Although this method is more convenient, if you create a text file, you can edit your word list later and reuse it in another vocabulary filter.

### AWS Management Console

To use the AWS Management Console to create a vocabulary filter, you must have a plain text file that contains the words that you want to filter, formatted as described in [Step 1: Creating a list of unwanted words \(p. 118\)](#). Your file can be saved locally or in Amazon S3.

#### To create a vocabulary filter (AWS Management Console)

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Vocabulary filtering**.
3. Choose **Create vocabulary filter**.
4. For **Name**, enter a vocabulary filter name that is unique within your AWS account.
5. For **Language**, choose the language code for the language of your vocabulary filter.
6. For **Vocabulary input source**, choose one of the following:
  - If you saved the file that contains the word that you want to filter locally, choose **File upload**, then choose **Choose file** and choose the file.
  - If you saved the file in Amazon S3, for **S3 location**, enter the URI of the text file or choose **Browse S3** and browse to the file and choose it.
7. Choose **Create vocabulary filter**.

## API

### To create a vocabulary filter (API)

- For the [CreateVocabularyFilter](#) API, specify the following:
  - a. A name for your vocabulary filter that is unique in your AWS account for the `VocabularyFilterName` parameter
  - b. The language code for the language of your source audio in the `LanguageCode` parameter
  - c. The words for your vocabulary filter using one of the following options:
    - Specify the Amazon S3 location of the text file for the `VocabularyFilterFileUri` parameter using this format: `s3://DOC-EXAMPLE-BUCKET/my-vocabulary-filter.txt`.
    - Enter the words as an array of strings in the `Words` parameter, for example `[ "word", "banana", "potato", "chair" ]`.

To see all of the vocabulary filters that you've created, use the [ListVocabularyFilters](#) API. You can then use that information with the [GetVocabularyFilter](#) API to retrieve the download URL for your vocabulary filter and learn more about that filter.

## AWS CLI

The following is an example AWS CLI request to create a vocabulary filter with a text file stored in an Amazon S3 bucket. The commands are followed by the response elements in JSON format.

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

{
    "VocabularyFilterName": "my-first-vocabulary-filter",
    "LanguageCode": "en-US"
}
```

### Next step

[Step 3: Filtering transcriptions \(p. 120\)](#)

## Step 3: Filtering transcriptions

You can filter unwanted words from both batch and streaming transcriptions. When you create a real-time stream or batch transcription job, specify the vocabulary filter that you want to use and the vocabulary filter method. The method specifies how the words are filtered from your transcription results. There are two vocabulary filter methods available for batch transcription and three methods available for real-time streaming.

You can use the following filter methods for both batch and real-time streaming transcriptions:

- To replace the words caught by your vocabulary filter with three asterisks, `***`, use the `mask` method. Use this method to hide unwanted words from your audience, but indicate that they were spoken.
- To remove the words from your transcripts, use the `remove` method. With this method, your audience won't know that unwanted words were spoken.

In real-time streaming transcriptions *only*, you can use the `tag` method to keep unwanted words in your transcription results with a tag that indicates that they were listed in your vocabulary filter. You can then manually remove the words from some transcripts and leave them in others to generate transcripts for multiple audiences from a single stream.

For information about streaming transcriptions, see [Transcribing streaming audio \(p. 44\)](#). For information about batch transcriptions, see [How Amazon Transcribe works \(p. 10\)](#).

### Topics

- [Filtering batch transcriptions \(p. 121\)](#)
- [Filtering streaming transcriptions \(p. 125\)](#)

## Filtering batch transcriptions

Use a vocabulary filter to filter unwanted words from a batch transcription job with either the [AWS Management Console](#) or the [StartTranscriptionJob API](#).

The following code shows the parameters and data types.

```
{  
    "ContentRedaction": {  
        "RedactionOutput": "string",  
        "RedactionType": "string"  
    },  
    "JobExecutionSettings": {  
        "AllowDeferredExecution": boolean,  
        "DataAccessRoleArn": "string"  
    },  
    "LanguageCode": "string",  
    "Media": {  
        "MediaFileUri": "string"  
    },  
    "MediaFormat": "string",  
    "MediaSampleRateHertz": number,  
    "OutputBucketName": "string",  
    "OutputEncryptionKMSKeyId": "string",  
    "Settings": {  
        "ChannelIdentification": boolean,  
        "MaxAlternatives": number,  
        "MaxSpeakerLabels": number,  
        "ShowAlternatives": boolean,  
        "ShowSpeakerLabels": boolean,  
        "VocabularyFilterMethod": "string",  
        "VocabularyFilterName": "string",  
        "VocabularyName": "string"  
    },  
    "TranscriptionJobName": "string"  
}
```

### AWS Management Console

To use the AWS Management Console to start a batch transcription job with vocabulary filtering, you must have created a vocabulary filter, as described in [Step 2: Creating a vocabulary filter \(p. 119\)](#).

#### To filter unwanted words in a transcription job (AWS Management Console)

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose [Transcription jobs](#).

3. Choose **Create job**.
4. For **Name**, specify a name that is unique within your AWS account for your batch transcription job.
5. For **Language**, choose the language that will be spoken in your transcription job.
6. Specify the location of your media file in Amazon S3:
  - For **Input file location on S3** under **Input data**, specify the Amazon S3 URI that identifies the media file that you will transcribe.
    - Choose **Browse S3** under **Input data** to browse for the media file and choose it.
7. Choose **Next**.
8. Enable **Vocabulary filtering** under **Content removal**.
9. Choose your vocabulary filter and vocabulary filtering method under **Filter selection**.
10. Choose **Create**.

## API

### To filter a batch transcription (API)

- For the [StartTranscriptionJob](#) API, specify the following:
  - a. For **TranscriptionJobName**, specify a name unique to your AWS account.
  - b. For **LanguageCode**, specify the language code that corresponds to the language spoken in your media file and the language of your vocabulary filter.
  - c. For the **MediaFileUri** parameter of the **Media** object, specify the Amazon S3 location of the media file you want to transcribe.
  - d. For the **VocabularyFilterName** parameter, specify the name of your vocabulary filter.
  - e. For the **VocabularyFilterMethod** parameter, choose one of the following options:
    - To mask filtered words by replacing them with three asterisks **\*\*\***, specify **mask**. Filtering the word "lazy" from the sentence: "The quick brown fox jumps over the lazy dog." with the **mask** method shows "The quick brown fox jumps over the \*\*\* dog." in the transcription.
    - To remove the filtered words from the transcript, specify **remove**. Filtering the word "lazy" from the sentence "The quick brown fox jumps over the lazy dog." with the **remove** method shows "The quick brown fox jumps over the dog." in the transcript.
    - To mark words that match your vocabulary filter in the transcript, specify **tag**. Use this to produce transcripts that are tailored to different audiences. This enables you to:
      - Present the transcription results to one audience, without removing any of the marked words that match the vocabulary filter.
      - Remove any word that matched the vocabulary filter, and present those results to a different audience.
      - Remove some words that matched the vocabulary filter, and present those results to another audience. You can generate multiple transcriptions for many audiences from the same transcription output. For more information, see [Tailoring transcripts to different audiences with tagging \(p. 122\)](#).

## Tailoring transcripts to different audiences with tagging

You can generate multiple transcriptions tailored to different audiences from the same media content. For the [StartTranscriptionJob](#) API, use the **tag** method to mark the words in the transcription that match the words in your vocabulary filter. You can present the results of the transcription job to the audience that can see the complete transcription, including the words listed in your vocabulary filter. You can then copy your transcription results, remove the words tagged by your vocabulary filter, and show those results to the audience that shouldn't see the unwanted words.

With tagging, you aren't limited to generating transcriptions for two different audiences. You can generate multiple transcriptions for many audiences from the same media content. You can choose to remove some words caught by the vocabulary filter in one transcript and leave them in other transcripts.

For example, if "lazy" were in the vocabulary filter, the sentence "The quick brown fox jumps over the lazy dog." would be unchanged in the transcription results. Instead of being masked in, or removed from, the transcription, the value for the `VocabularyFilterMatch` parameter would be `true` for "lazy."

To enable tagging in your batch transcription job, see [Filtering batch transcriptions \(p. 121\)](#).

The word "bloody" is tagged in the following truncated transcription output.

```
{  
    "jobName": "transcription-job-name",  
    "accountId": "111122223333",  
    "results": [  
        "transcripts": [  
            {  
                "transcript": "...recording bloody well set up this stupid bloody meeting  
anyway...."  
                ...  
                "items": [  
                    ...  
                    {  
                        "start_time": "5.4",  
                        "end_time": "5.95",  
                        "alternatives": [  
                            {  
                                "confidence": "0.9966",  
                                "content": "recording"  
                            }  
                        ],  
                        "type": "pronunciation",  
                        "vocabularyFilterMatch": false  
                    },  
                    {  
                        "start_time": "7.84",  
                        "end_time": "8.54",  
                        "alternatives": [  
                            {  
                                "confidence": "1.0",  
                                "content": "bloody"  
                            }  
                        ],  
                        "type": "pronunciation",  
                        "vocabularyFilterMatch": true  
                    },  
                    {  
                        "start_time": "8.54",  
                        "end_time": "9.03",  
                        "alternatives": [  
                            {  
                                "confidence": "1.0",  
                                "content": "well"  
                            }  
                        ],  
                        "type": "pronunciation",  
                        "vocabularyFilterMatch": false  
                    },  
                    {  
                        "start_time": "9.04",  
                        "end_time": "9.31",  
                        "alternatives": [  
                            ...  
                        ]  
                    }  
                ]  
            }  
        ]  
    ]  
}
```

```
{
    "confidence":"0.9905",
    "content":"set"
},
],
"type":"pronunciation",
"vocabularyFilterMatch":false
},
{
"start_time":"9.31",
"end_time":"9.44",
"alternatives":[
{
    "confidence":"0.9905",
    "content":"up"
},
],
"type":"pronunciation",
"vocabularyFilterMatch":false
},
{
"start_time":"9.44",
"end_time":"9.6",
"alternatives":[
{
    "confidence":"0.8796",
    "content":"this"
},
],
"type":"pronunciation",
"vocabularyFilterMatch":false
},
{
"start_time":"9.6",
"end_time":"10.22",
"alternatives":[
{
    "confidence":"1.0",
    "content":"stupid"
},
],
"type":"pronunciation",
"vocabularyFilterMatch":false
},
{
"start_time":"10.23",
"end_time":"10.66",
"alternatives":[
{
    "confidence":"1.0",
    "content":"bloody"
},
],
"type":"pronunciation",
"vocabularyFilterMatch":true
},
{
"start_time":"10.66",
"end_time":"11.21",
"alternatives":[
{
    "confidence":"0.9994",
    "content":"meeting"
}
],
"type":"pronunciation",
```

```

        "vocabularyFilterMatch":false
    },
    {
        "start_time":"11.21",
        "end_time":"11.55",
        "alternatives":[
            {
                "confidence":"0.9978",
                "content":"anyway"
            }
        ],
        "type":"pronunciation",
        "vocabularyFilterMatch":false
    }
    ...
],
},
"status":"COMPLETED"
}

```

## Filtering streaming transcriptions

Use a vocabulary filter to filter unwanted words in real-time streams with either the [AWS Management Console](#) or the [StartStreamTranscription API](#).

The following syntax shows the parameters and their data types.

```
{
    "LanguageCode" : "enum",
    "MediaSampleRateHertz" : "integer",
    "MediaEncoding" : "enum",
    "VocabularyName" : "string",
    "SessionId" : "string",
    "AudioStream" : "eventstream",
    "VocabularyFilterName" : "string",
    "VocabularyFilterMethod": "enum"
}
```

### To filter a streaming transcription (API)

- For the [StartStreamTranscription API](#), specify the following:
  - a. The language code of your audio in the `LanguageCode` field.
  - b. The sample rate of your audio in the `MediaSampleHertz` field.
  - c. The name of your vocabulary filter in the `VocabularyFilterName` field.
  - d. The filtering method in the `VocabularyFilterMethod` parameter:
    - To mask the filtered words by replacing them with three asterisks (\*\*\*) specify `mask`. Filtering the word "lazy" from the sentence "The quick brown fox jumps over the lazy dog." with the `mask` method shows "The quick brown fox jumps over the \*\*\* dog." in the transcription.
    - To remove the words from the transcript, specify `remove`. Filtering the word "lazy" from the sentence "The quick brown fox jumps over the lazy dog." with the `remove` method shows "The quick brown fox jumps over the dog." in the transcription.
    - To tag words that match the vocabulary filter, specify `tag`. This enables you

to mark the words matching the vocabulary filter without masking or removing them.

To use the same stream to create one transcript with the content filtered and one transcript that is unfiltered, use the tagging method. For information, see [Tailoring transcripts to different audiences with tagging \(p. 126\)](#).

#### To filter a streaming transcription (AWS Management Console)

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Real-time transcription**.
3. In **Language**, choose the language of your real-time stream.
4. Choose the **Additional settings** tab and choose your vocabulary filter and vocabulary filtering method.
5. Choose **Start streaming** to begin your stream with vocabulary filtering enabled.

## Tailoring transcripts to different audiences with tagging

You can use a single stream to generate a transcription that doesn't show unwanted words and one that does. For the [StartStreamTranscription](#) API, use the tag method to mark the words in the transcription that match the words in your vocabulary filter. You can present the results of the real-time stream to the audience that can see the complete transcription, including the words listed in your vocabulary filter. You can then copy your transcription results, remove the words tagged by your vocabulary filter, and show those results to the audience that shouldn't see the unwanted words.

With tagging, you aren't limited to generating transcriptions for two different audiences. You can generate multiple transcriptions for many audiences from the same stream. You can choose to remove some words caught by the vocabulary filter in one transcript and leave them in other transcripts.

#### To enable tagging in a real-time transcription

- For the [StartStreamTranscription](#) API, specify the following:
  - a. For **VocabularyFilterName**, the name of your vocabulary filter.
  - b. For **VocabularyFilterMethod**, specify `tag`.

For example, if "bloody" were in the vocabulary filter, the phrase "recording bloody well set up this stupid bloody meeting anyway..." would be unchanged in the transcription results. Instead of being masked in, or removed from, the transcription, the value for the **VocabularyFilterMatch** parameter would be `true` for "bloody."

The following example JSON output shows this.

```
{  
    "jobName": "my-first-transcription-job",  
    "accountId": "111122223333",  
    "results": {  
        "transcripts": [  
            {  
                "transcript": "...recording bloody well set up this stupid bloody meeting  
anyway..."  
            }  
        ],  
        "items": [  
            {  
                "item": "recording bloody well set up this stupid bloody meeting  
anyway...",  
                "startOffset": 0,  
                "endOffset": 100,  
                "vocabularyFilterMatch": true  
            }  
        ]  
    }  
}
```

```
{
    "start_time": "5.4",
    "end_time": "5.95",
    "alternatives": [
        {
            "confidence": "0.9966",
            "content": "recording"
        }
    ],
    "type": "pronunciation",
    "vocabularyFilterMatch": false
},
{
    "start_time": "7.84",
    "end_time": "8.54",
    "alternatives": [
        {
            "confidence": "1.0",
            "content": "bloody"
        }
    ],
    "type": "pronunciation",
    "vocabularyFilterMatch": true
},
{
    "start_time": "8.54",
    "end_time": "9.03",
    "alternatives": [
        {
            "confidence": "1.0",
            "content": "well"
        }
    ],
    "type": "pronunciation",
    "vocabularyFilterMatch": false
},
{
    "start_time": "9.04",
    "end_time": "9.31",
    "alternatives": [
        {
            "confidence": "0.9905",
            "content": "set"
        }
    ],
    "type": "pronunciation",
    "vocabularyFilterMatch": false
},
{
    "start_time": "9.31",
    "end_time": "9.44",
    "alternatives": [
        {
            "confidence": "0.9905",
            "content": "up"
        }
    ],
    "type": "pronunciation",
    "vocabularyFilterMatch": false
},
{
    "start_time": "9.44",
    "end_time": "9.6",
    "alternatives": [
        {
            "confidence": "0.8796",
            "content": "the"
        }
    ],
    "type": "pronunciation",
    "vocabularyFilterMatch": false
}
}
```

```

        "content": "this"
    },
],
"type": "pronunciation",
"vocabularyFilterMatch": false
},
{
"start_time": "9.6",
"end_time": "10.22",
"alternatives": [
{
"confidence": "1.0",
"content": "stupid"
}
],
"type": "pronunciation",
"vocabularyFilterMatch": false
},
{
"start_time": "10.23",
"end_time": "10.66",
"alternatives": [
{
"confidence": "1.0",
"content": "bloody"
}
],
"type": "pronunciation",
"vocabularyFilterMatch": true
},
{
"start_time": "10.66",
"end_time": "11.21",
"alternatives": [
{
"confidence": "0.9994",
"content": "meeting"
}
],
"type": "pronunciation",
"vocabularyFilterMatch": false
},
{
"start_time": "11.21",
"end_time": "11.55",
"alternatives": [
{
"confidence": "0.9978",
"content": "anyway"
}
],
"type": "pronunciation",
"vocabularyFilterMatch": false
}
]
},
"status": "COMPLETED"
}

```

# Transcribing multi-channel audio

If your audio has multiple channels, you can use *channel identification* to transcribe the speech from each of those channels. Amazon Transcribe identifies the speech from each channel and transcribes that speech in separate transcriptions. It combines those transcriptions into a single transcription output.

You can enable channel identification for both batch processing and real-time streaming. The following list describes how to enable it for each method.

- Batch transcription - [AWS Management Console](#) and [StartTranscriptionJob](#) API
- Streaming transcription - [WebSocket streaming](#) and [StartStreamTranscription](#) API

## Transcribing multi-channel audio

To transcribe multi-channel audio in a batch transcription job, use the [AWS Management Console](#) or the [StartTranscriptionJob](#) API.

### AWS Management Console

To use the AWS Management Console to enable channel identification in your batch transcription job, you enable audio identification and then channel identification. Channel identification is a subset of audio identification in the AWS Management Console.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe, choose **Transcription jobs**.
3. Choose **Create job**.
4. On the **Specify job details** page, provide information about your transcription job.
5. Choose **Next**.
6. Enable **Audio identification**.
7. For **Audio identification type**, choose **Channel identification**.
8. Choose **Create**.

### API

#### To transcribe multi-channel audio (API)

- For the [StartTranscriptionJob](#) API, specify the following.
  - a. For `TranscriptionJobName`, specify a name unique to your AWS account.
  - b. For `LanguageCode`, specify the language code that corresponds to the language spoken in your media file. For available languages and corresponding language codes, see [What is Amazon Transcribe? \(p. 1\)](#).
  - c. For the `MediaFileUri` parameter of the `Media` object, specify the name of the media file you want to transcribe.

- d. For the `Settings` object, set `ChannelIdentification` to `true`.

The following is an example request using the AWS SDK for Python (Boto3).

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName=job_name,
    Media= {'MediaFileUri': job_uri},
    MediaFormat='flac',
    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)
```

## AWS CLI

- Run the following code.

```
aws transcribe start-transcription-job \
--cli-input-json file://filepath/my-first-channel-id-job.json
```

The following is the code of `my-first-channel-id-job.json`.

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

The following is the response.

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "my-first-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
```

```

        "LanguageCode": "en-US",
        "Media": {
            "MediaFileUri": "s3:/DOC-EXAMPLE-BUCKET/my-media-file.flac"
        },
        "StartTime": "2020-09-12T23:20:28.239000+00:00",
        "CreationTime": "2020-09-12T23:20:28.203000+00:00",
        "Settings": {
            "ChannelIdentification": true
        }
    }
}

```

The following code shows the transcription output for audio that has a conversation on two channels.

```

{
    "jobName": "job id",
    "accountId": "account id",
    "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"
                        }
                    ]
                }
            ]
        }
    }
}

```

```
        "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"
        },
        ...
    ]
}
```

Amazon Transcribe transcribes the audio from each channel separately and combines the transcribed text from each channel into a single transcription output.

For each channel in the transcription output, Amazon Transcribe returns a list of *items*. An item is a transcribed word, pause, or punctuation mark. Each item has a start time and an end time. If a person on one channel speaks over a person on a separate channel, the start times and end times of the items for each channel overlap while the individuals are speaking over each other.

## Transcribing multi-channel audio streams

You can transcribe audio from separate channels in either HTTP/2 or WebSocket streams using the [StartStreamTranscription](#) API.

## Transcribing multi-channel audio in an HTTP/2 stream

To transcribe multi-channel audio in an HTTP/2 stream, use the [StartStreamTranscription](#) API and specify the following:

- **LanguageCode** - The language code of the audio.
- **MediaEncoding** - The encoding of the audio.
- **EnableChannelIdentification** - true
- **NumberOfChannels** - The number of channels in your streaming audio.

The following is the syntax for the parameters of an HTTP/2 request.

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
authorization: Generated value
x-amz-target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-EVENTS
x-amz-date: 20220208T235959Z
x-amzn-transcribe-session-id: my-first-http2-channel-id-stream
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
x-amzn-transcribe-enable-channel-identification: true
x-amzn-transcribe-number-of-channels: 2
content-type: application/vnd.amazon.eventstream
transfer-encoding: chunked
```

Parameter descriptions:

- **host:** Update the AWS Region ('us-west-2' in the preceding example) with the AWS Region you are calling. For a list of valid AWS Regions, see [AWS Regions and Endpoints](#).
- **authorization:** This is a generated field. To learn more about creating a signature, see [Signing AWS requests with Signature Version 4](#).
- **x-amz-target:** Don't alter this field; use the content shown in the preceding example.
- **x-amz-content-sha256:** This is a generated field. To learn more about calculating a signature, see [Signing AWS requests with Signature Version 4](#).
- **x-amz-date:** The date and time the signature is created. The format is YYYYMMDDTHHMMSSZ, where YYYY=year, MM=month, DD=day, HH=hour, MM=minute, SS=seconds, and 'T' and 'Z' are fixed characters. For more information, refer to [Handling Dates in Signature Version 4](#).
- **x-amzn-transcribe-session-id:** The name for your streaming session.
- **x-amzn-transcribe-language-code:** The encoding used for your input audio. Refer to [StartStreamTranscription](#) or [Supported languages and language-specific features \(p. 3\)](#) for a list of valid values.
- **x-amzn-transcribe-media-encoding:** The encoding used for your input audio. Valid values are `pcm`, `ogg-opus`, and `flac`.
- **x-amzn-transcribe-sample-rate:** The sample rate of the input audio (in Hertz). Amazon Transcribe supports a range from 8,000 Hz to 48,000 Hz. Low-quality audio, such as telephone audio, is typically around 8,000 Hz. High-quality audio typically ranges from 16,000 Hz to 48,000 Hz. Note that the sample rate you specify **must** match that of your audio.
- **x-amzn-transcribe-enable-channel-identification:** To enable channel identification, this field must be `true`.

- **x-amzn-transcribe-number-of-channels:** Must be 2 when **x-amzn-transcribe-enable-channel-identification** is set to true.
- **content-type:** Don't alter this field; use the content shown in the preceding example.

## Transcribing multi-channel audio in a WebSocket stream

To identify speakers in WebSocket streams, use the following format to create a pre-signed URI and start a WebSocket request. Specify `enable-channel-id` as `true` and the number of channels in your stream in `number-of-channels`. A pre-signed URI contains the information needed to set up bi-directional communication between your application and Amazon Transcribe.

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/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=250
&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=number of channels in your audio stream
```

## Multi-channel streaming output

The output of a streaming transcription is the same for HTTP/2 and WebSocket requests. The following is an example output.

```
{
    "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"  
}
```

For each speech segment, there is a `channelId` flag that indicates which channel the speech belongs to.

# Identifying the dominant language in your media

Amazon Transcribe is able to automatically identify the language spoken in your media without you having to specify a language code. Language identification works for all [supported languages \(p. 3\)](#); note that not all supported languages can be used with streaming transcriptions.

Batch jobs can only be transcribed in one language, while streaming transcriptions can have one language per channel (a maximum of two channels are supported).

To improve language identification accuracy, you can provide a list of languages you think may be present. From this list, Amazon Transcribe chooses the closest matching language to transcribe your audio. Providing language codes is optional with batch jobs and required for streaming transcriptions. If none of the language codes you provide match languages spoken in your media, your transcription accuracy is diminished.

## Note

If none of your specified language codes match any of the languages spoken in your media, Amazon Transcribe selects the closest option in your subset of selected language codes and produces a transcript based on that language. For example, if your media is in US English (en-US) and you provide Amazon Transcribe with the language codes zh-CN, fr-FR, and de-DE, Amazon Transcribe is likely to match your media to German (de-DE) and produce a German-language transcription. Mismatching language codes and spoken languages can result in a very inaccurate transcript, so we advise caution when selecting language codes.

For additional information, see [Language identification with batch transcription jobs \(p. 136\)](#) and [Language identification with streaming transcriptions \(p. 141\)](#).

To learn about monitoring and events with language identification, refer to [Language identification events \(p. 237\)](#).

## Language identification with batch transcription jobs

Use language identification with batch jobs to identify the dominant language in your media file. If your media file contains multiple languages, your transcription is created based on only the dominant language.

To improve language identification accuracy, you can provide a list of languages you think may be present. From this list, Amazon Transcribe chooses the closest matching language to transcribe your audio. If none of the language codes you provide match languages spoken in your media, your transcription accuracy is diminished—this is because Amazon Transcribe prioritizes the languages you select. Although there is no limit on the number of language codes you can include, we recommend using five or less for optimal efficiency and accuracy.

## Tip

Providing a subset of language options is particularly helpful in transcribing regional dialects. For example, if you know your media file contains an English dialect from the United Kingdom, but you're not sure which accent is present, provide Amazon Transcribe with en-GB, en-AB, and en-WL and omit the other English language codes.

Here's an example of what confidence scores look like using a media file with one New Zealand English speaker and one US English speaker:

Without any language codes provided	With language codes (en-US, en-NZ, en-AU) provided
<pre>"language_identification": [     {         "score": "0.4633",         "code": "en-NZ"     },     {         "score": "0.2386",         "code": "en-US"     },     {         "score": "0.1711",         "code": "en-AU"     },     {         "score": "0.0662",         "code": "en-IE"     },     {         "score": "0.0607",         "code": "en-ZA"     } ],</pre>	<pre>"language_identification": [     {         "score": "0.5336",         "code": "en-NZ"     },     {         "score": "0.2681",         "code": "en-US"     },     {         "score": "0.1983",         "code": "en-AU"     } ],</pre>

As you can see in the preceding table, specifying at least one language code improves language identification confidence levels.

You can use batch language identification in combination with any other Amazon Transcribe feature. If combining language identification with another feature, you are limited to the languages supported with those features.

**Important**

If you're using automatic language identification with content redaction enabled, make sure that the language of your media file is [supported with redaction \(p. 3\)](#). If the language in your file is not supported, your transcript is **not** redacted and there are no warnings or job failures.

If using language identification with a language model, vocabulary filter, or custom vocabulary, you must use the [LanguageSettings](#) parameter in your request. Using this parameter, you can specify one language model, one vocabulary filter, and one custom vocabulary for every language code you provide. Note that the language codes for each must match the language codes you specify in your request. For an example of what this parameter looks like in a request, refer to Option 2 in the [AWS CLI](#) and [AWS SDK](#) drop-down panels.

For best results, ensure your media file contains at least 30 seconds of speech.

You can use automatic language identification in a batch transcription job using the [AWS Management Console](#), [AWS CLI](#), or [AWS SDK](#); see the following for examples:

## AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Transcription jobs**, then select the **Create job** button (top right). This opens the **Specify job details** page.
3. In the **Job settings** panel, find the **Language settings** section and select **Automatic language identification**. You also have the option to select multiple languages (from the *Select languages*

drop-down box) if you have an idea of which language your file contains. This option can improve accuracy, but is not required.

## Specify job details [Info](#)

**Job settings**

Name  The name can be up to 200 characters long. Valid characters are a-z, A-Z, 0-9, . (period), \_ (underscore), and – (hyphen).

Model type [Info](#)  
Choose the type of model to use for the transcription job.

General model  
To use a model that is not specialized for a particular use case, choose this option. Configuration options vary between languages.

Custom language model  
To use a model that you trained for your specific use case, choose this option. This model has fewer configuration options than the general model.

Language settings  
You can transcribe your audio file in a language that you specify or have Amazon Transcribe identify and transcribe it in the predominant language.

Specific language [Info](#)  
If you know the language spoken in your source audio, choose this option to get the most accurate results. The options available for additional processing vary between languages.

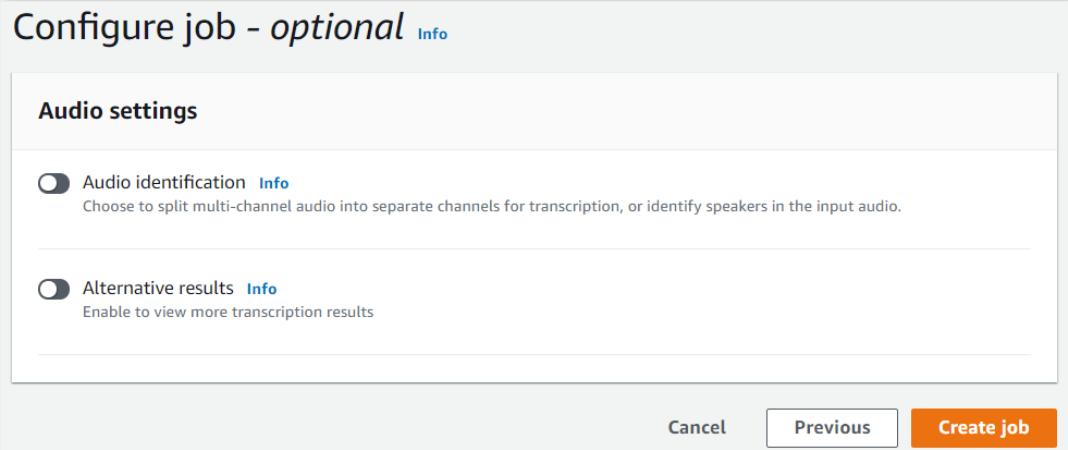
Automatic language identification [Info](#)  
If you don't know the language spoken in your audio files, choose this option. You have access to fewer options for additional processing than if you choose **Specific language**.

Language options for automatic language identification - *optional*  
To improve accuracy, choose at least two languages spoken the most often in your audio library. Amazon Transcribe chooses from one of the languages you've specified to transcribe each audio file. Leave this field empty if you're unsure about which languages to select.

▼ Additional settings

Job queue - *optional* [Info](#)  
Enables you to submit jobs beyond the limit for concurrent jobs (100). You must specify access permissions to the resources that job queuing uses.  
 Add to job queue

- Fill in any other fields you wish to include on the **Specify job details** page, then click the **Next** button. This takes you to the **Configure job - *optional*** page.



- Click the **Create job** button to run your transcription job.

## AWS CLI

This example uses the [start-transcription-job](#) command and `IdentifyLanguage` parameter. For more information, see [StartTranscriptionJob](#) and [LanguageIdSettings](#).

**Option 1:** Without the `language-id-settings` parameter.

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--identify-language
```

**Option 2:** With the `language-id-settings` parameter.

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--identify-language --language-id-settings "{\"en-US\":{\"VocabularyName\":\"my-en-US-vocabulary\"}, \
\"VocabularyFilterName\":\"my-en-US-vocabulary-filter\", \"LanguageModelName\":\"my-en-US- \
language-model\"}, \
\"en-AU\":{\"VocabularyName\":\"my-en-AU-vocabulary\"}}}
```

Here's another example using the [start-transcription-job](#) command, and a request body that identifies the language.

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

The file `my-first-language-id-job.json` contains the following request body.

```
{
```

```

    "TranscriptionJobName": "my-first-transcription-job",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
    },
    "OutputBucketName": "DOC-EXAMPLE-BUCKET",
    "IdentifyLanguage": "true"
}

```

## AWS SDK for Python (Boto3)

This example uses the AWS SDK for Python (Boto3) to identify your file's language using the `IdentifyLanguage` argument for the [start\\_transcription\\_job](#) method. For more information, see [StartTranscriptionJob](#) and [LanguageIdSettings](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

**Option 1:** Without the `LanguageIdSettings` parameter.

```

from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName=job_name,
    Media={
        'MediaFileUri': job_uri
    },
    MediaFormat='flac',
    IdentifyLanguage=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)

```

**Option 2:** With the `LanguageIdSettings` parameter.

```

from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName=job_name,
    Media={'MediaFileUri': job_uri},
    MediaFormat='flac',
    IdentifyLanguage=True,
    LanguageIdSettings={
        'en-US': {
            'VocabularyName': 'my-en-US-vocabulary',
            'VocabularyFilterName': 'my-en-US-vocabulary-filter',
            'LanguageModelName': 'my-en-US-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)
```

## Language identification with streaming transcriptions

Use streaming language identification to identify the dominant language spoken in a media stream. Amazon Transcribe requires a minimum of three seconds of speech to identify the predominant language.

Amazon Transcribe can identify the dominant language spoken in two different channels. In this case, set the [ChannelIdentification](#) parameter to `true` and each channel is transcribed separately. Note that the default for this parameter is `false` and if you don't change it, only the first channel is transcribed.

You must provide a minimum of two language codes with streaming language identification, and you can select only one language variant (locale) per language per stream. This means that you cannot select `en-US` and `en-AU` as language options for the same transcription.

You also have the option to select a preferred language from the set of language codes you provide. Adding a preferred language helps Amazon Transcribe identify the language in the first few seconds of your stream.

Streaming language identification can't be combined with custom language models or redaction.

**Note**

PCM and FLAC are the only supported audio formats for streaming language identification.

For a list of languages supported with streaming transcriptions, see [supported languages \(p. 3\)](#).

You can use automatic language identification in a streaming transcription using the [AWS Management Console](#), [HTTP/2](#), or [WebSockets](#); see the following for examples:

### AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Real-time transcription**. Scroll down to **Language settings** and expand this field if it is minimized.

**Real-time transcription** Info

See how Amazon Transcribe creates a text copy of speech in real time. Choose **Start streaming** and talk.

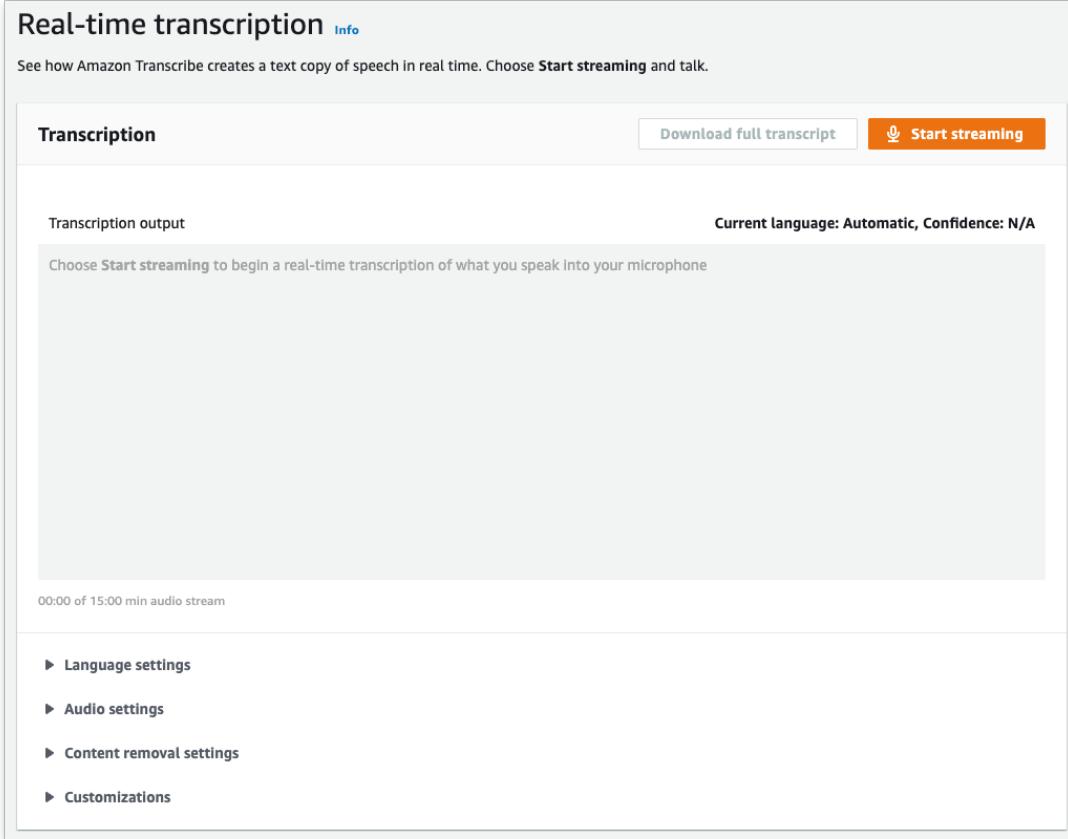
**Transcription** Download full transcript   Start streaming

Transcription output Current language: Automatic, Confidence: N/A

Choose Start streaming to begin a real-time transcription of what you speak into your microphone

00:00 of 15:00 min audio stream

► Language settings  
► Audio settings  
► Content removal settings  
► Customizations



### 3. Select Automatic language identification.

▼ Language settings

**Language settings**  
You can select a specific language for your transcription or have Amazon Transcribe identify the predominant language in your media and perform the transcription in that language.

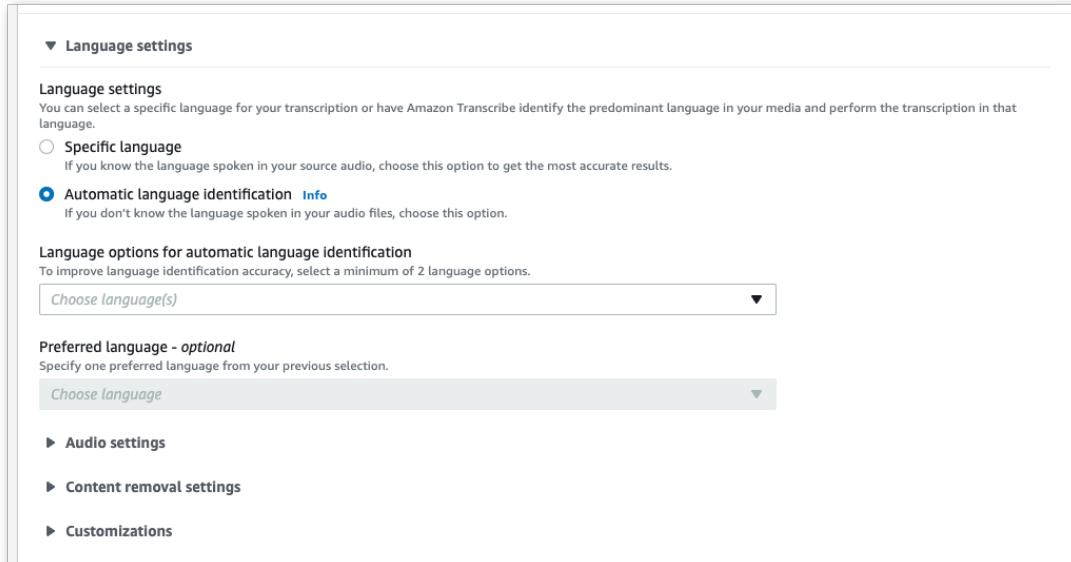
**Specific language**  
If you know the language spoken in your source audio, choose this option to get the most accurate results.

**Automatic language identification** Info  
If you don't know the language spoken in your audio files, choose this option.

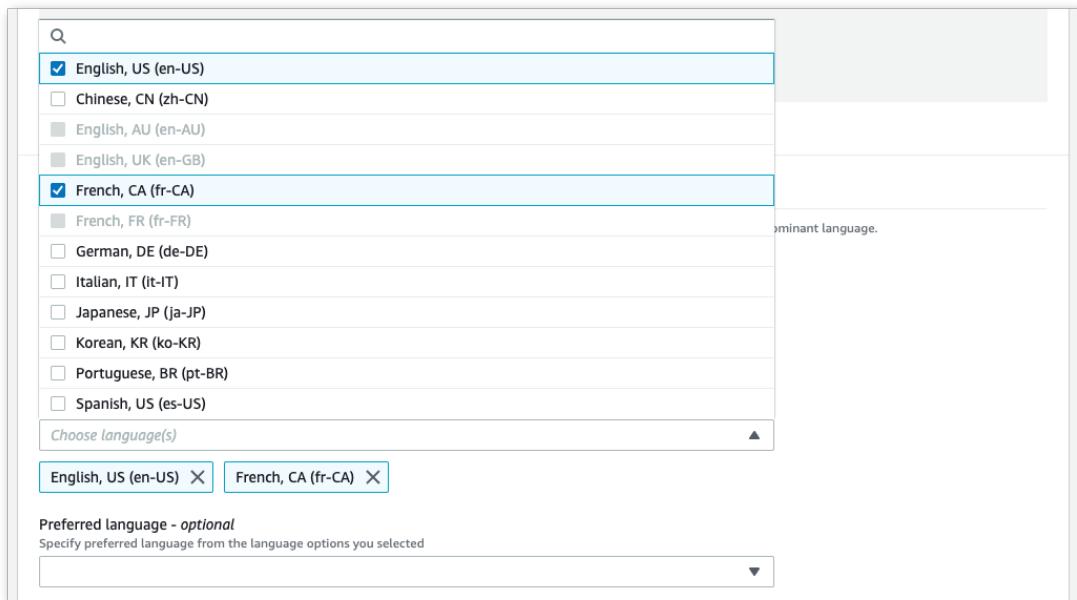
**Language options for automatic language identification**  
To improve language identification accuracy, select a minimum of 2 language options.

**Preferred language - optional**  
Specify one preferred language from your previous selection.

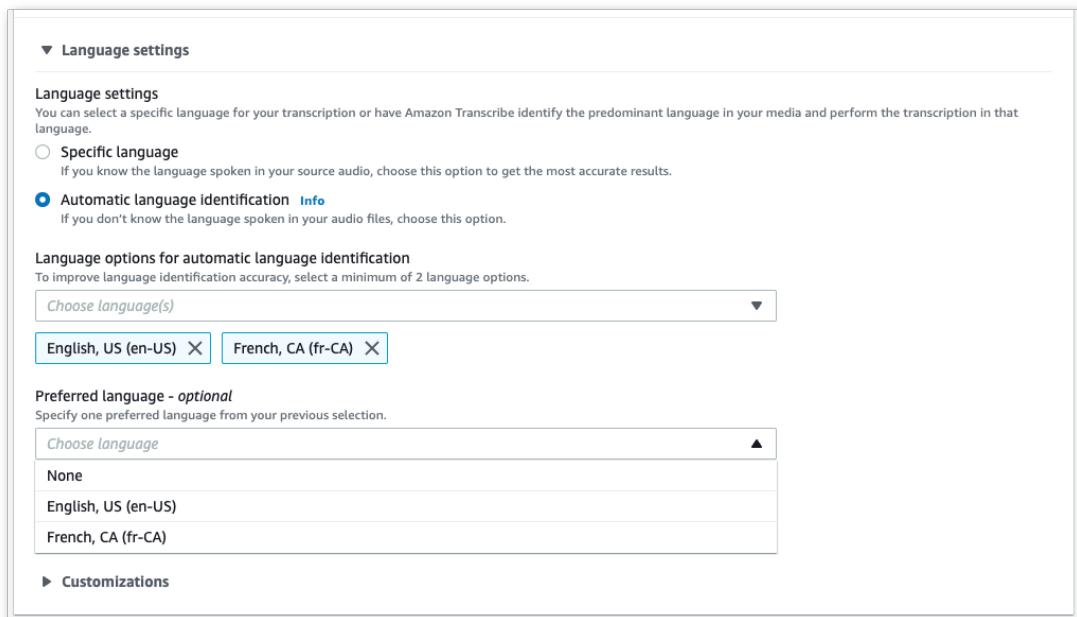
► Audio settings  
► Content removal settings  
► Customizations



### 4. Provide a minimum of 2 language codes for your transcription. Note that you can provide only one variant (locale) per language. For example, you cannot select both en-US and en-AU as language options for the same transcription.



5. (Optional) From the subset of languages you selected in the previous step, you can choose a preferred language for your transcript.



6. You're now ready to transcribe your stream. Select the **Start streaming** button and begin speaking. To end your dictation, select **Stop streaming**.

## HTTP/2 stream

This example creates an HTTP/2 request with language identification enabled. For more information on using HTTP/2 streaming with Amazon Transcribe, see [Setting up an HTTP/2 stream \(p. 50\)](#). For more detail on parameters and headers specific to Amazon Transcribe, see [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
```

If you use `identify-language` in your request, you must also include `language-options`. You cannot use both `language-code` and `identify-language` in the same request.

Parameter definitions can be found in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

## WebSocket stream

This example creates a pre-signed URL that uses language identification in a WebSocket stream. Line breaks have been added for readability. For more information on using WebSocket streams with Amazon Transcribe, see [Setting up a WebSocket stream \(p. 54\)](#). For more detail on parameters, see [StartStreamTranscription](#).

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=250
&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
```

If you use `identify-language` in your request, you must also include `language-options`. You cannot use both `language-code` and `identify-language` in the same request.

Parameter definitions can be found in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

# Identifying speakers (diarization)

To identify different speakers in Amazon Transcribe, use *speaker diarization*. When you enable speaker diarization, Amazon Transcribe labels each speaker utterance. You enable speaker diarization by using the batch transcription or real-time streaming APIs, or the AWS Management Console.

## Note

Speaker diarization is supported for all languages with batch transcription jobs, but is only supported for US English (en-US) with streaming transcriptions.

## Identifying speakers in audio

You can enable speaker diarization in a batch transcription job using either the [StartTranscriptionJob](#) API or the [AWS Management Console](#).

### AWS Management Console

To use the AWS Management Console to enable speaker diarization in your transcription job, you enable audio identification and then speaker diarization.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe, choose **Transcription jobs**.
3. Choose **Create job**.
4. On the **Specify job details** page, provide information about your transcription job.
5. Choose **Next**.
6. Enable **Audio identification**.
7. For **Audio identification type**, choose **Speaker identification**.
8. For **Maximum number of speakers**, specify the maximum number of speakers you think are speaking in your audio. For best results, match the number of speakers you ask Amazon Transcribe to identify to the number of speakers in the input audio. If you specify a value less than the number of speakers in your input audio, the transcription text of the most similar sounding speakers are attributed to a speaker label.
9. Choose **Create**.

### API

#### To identify speakers in audio using a batch transcription job (API)

- For the [StartTranscriptionJob](#) API, specify the following.
  - a. For **TranscriptionJobName**, specify a name unique to your AWS account.
  - b. For **LanguageCode**, specify the language code that corresponds to the language spoken in your media.
  - c. For the **MediaFileUri** parameter of the **Media** object, specify the name of the media file you want to transcribe.
  - d. For the **Settings** object, specify the following.

- i. `ShowSpeakerLabels - true.`
- ii. `MaxSpeakerLabels` - An integer between 2 and 10 that indicates the number of speakers you think are speaking in your audio. For best results, match the number of speakers you ask Amazon Transcribe to identify to the number of speakers in the input audio. If you specify a value less than the number of speakers in your input audio, the transcription text of the most similar sounding speakers are attributed to a speaker label.

The following syntax shows the request parameters to start a batch transcription job and their data types.

```
{  
    "ContentRedaction": {  
        "RedactionOutput": "string",  
        "RedactionType": "string"  
    },  
    "JobExecutionSettings": {  
        "AllowDeferredExecution": boolean,  
        "DataAccessRoleArn": "string"  
    },  
    "LanguageCode": "string",  
    "Media": {  
        "MediaFileUri": "string"  
    },  
    "MediaFormat": "string",  
    "MediaSampleRateHertz": number,  
    "OutputBucketName": "string",  
    "OutputEncryptionKMSKeyId": "string",  
    "Settings": {  
        "ChannelIdentification": boolean,  
        "MaxAlternatives": number,  
        "MaxSpeakerLabels": number,  
        "ShowAlternatives": boolean,  
        "ShowSpeakerLabels": boolean,  
        "VocabularyFilterMethod": "string",  
        "VocabularyFilterName": "string",  
        "VocabularyName": "string"  
    },  
    "TranscriptionJobName": "string"  
}
```

The following code shows an example output of a transcription job with speaker identification enabled.

```
{  
    "jobName": "job ID",  
    "accountId": "account ID",  
    "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

- Run the following code.

```
aws transcribe start-transcription-job \
--cli-input-json file://filepath/example-start-command.json
```

The following code shows the contents of example-start-command.json.

```
{  
    "TranscriptionJobName": "my-first-transcription-job",  
    "LanguageCode": "en-US",  
    "Media": {  
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"  
    },  
    "Settings":{  
        "MaxSpeakerLabels": 2,  
        "ShowSpeakerLabels":true  
    }  
}
```

The following is the response from running the preceding AWS CLI command.

```
{  
    "TranscriptionJob": {  
        "TranscriptionJobName": "my-first-transcription-job",  
        "TranscriptionJobStatus": "IN_PROGRESS",  
        "LanguageCode": "en-US",  
        "Media": {  
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"  
        },  
        "StartTime": "2020-07-29T17:45:09.826000+00:00",  
        "CreationTime": "2020-07-29T17:45:09.791000+00:00",  
        "Settings": {  
            "ShowSpeakerLabels": true,  
            "MaxSpeakerLabels": 2  
        }  
    }  
}
```

## Identifying speakers in real-time streams

You can identify different speakers in either HTTP/2 or WebSocket streams. Speaker diarization works best for identifying between two and five speakers. Although Amazon Transcribe can identify more than five speakers in a stream, the accuracy of speaker diarization decreases if you exceed that number. To start an HTTP/2 stream, you specify the `ShowSpeakerLabel` request parameter of the [StartStreamTranscription](#) API. To start a WebSocket request, you use a pre-signed URI, which is a URI that contains the information needed to start your stream. To use the AWS Management Console to transcribe speech spoken into your microphone, use the following procedure.

You can identify speakers in real-time streams that are in US English (en-US).

You can use the [AWS Management Console](#) to start a real-time stream and transcribe any speech picked up by your microphone.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Real-time transcription**.
3. In **Language**, choose the language of your real-time stream.
4. Under **Additional settings**, enable **Speaker identification**.
5. Choose **Start streaming**.
6. Speak into the microphone.

## HTTP/2 streaming

The following is the syntax for the parameters of an HTTP/2 request.

```
POST /stream-transcription HTTP/2
host: transcribestreaming.us-west-2.amazonaws.com
authorization: Generated value
x-amz-target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-EVENTS
x-amz-date: 20220208T235959Z
x-amzn-transcribe-session-id: my-first-http2-diarization-stream
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
content-type: application/vnd.amazon.eventstream
transfer-encoding: chunked
```

Parameter descriptions:

- **host:** Update the AWS Region ('us-west-2' in the preceding example) with the AWS Region you are calling. For a list of valid AWS Regions, see [AWS Regions and Endpoints](#).
- **authorization:** This is a generated field. To learn more about creating a signature, see [Signing AWS requests with Signature Version 4](#).
- **x-amz-target:** Don't alter this field; use the content shown in the preceding example.
- **x-amz-content-sha256:** This is a generated field. To learn more about calculating a signature, see [Signing AWS requests with Signature Version 4](#).
- **x-amz-date:** The date and time the signature is created. The format is YYYYMMDDTHHMMSSZ, where YYYY=year, MM=month, DD=day, HH=hour, MM=minute, SS=seconds, and 'T' and 'Z' are fixed characters. For more information, refer to [Handling Dates in Signature Version 4](#).
- **x-amzn-transcribe-session-id:** The name for your streaming session.
- **x-amzn-transcribe-language-code:** The encoding used for your input audio. Refer to [StartStreamTranscription](#) or [Supported languages and language-specific features \(p. 3\)](#) for a list of valid values.
- **x-amzn-transcribe-media-encoding:** The encoding used for your input audio. Valid values are `pcm`, `ogg-opus`, and `flac`.
- **x-amzn-transcribe-sample-rate:** The sample rate of the input audio (in Hertz). Amazon Transcribe supports a range from 8,000 Hz to 48,000 Hz. Low-quality audio, such as telephone audio, is typically around 8,000 Hz. High-quality audio typically ranges from 16,000 Hz to 48,000 Hz. Note that the sample rate you specify **must** match that of your audio.
- **x-amzn-transcribe-show-speaker-label:** To enable diarization, this value must be `true`.
- **content-type:** Don't alter this field; use the content shown in the preceding example.

## WebSocket streaming

To identify speakers in WebSocket streams, use the following format to create a pre-signed URI to start a WebSocket request and specify `show-speaker-label` as `true`. A pre-signed URI contains the information to set up bi-directional communication between your application and Amazon Transcribe.

```
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=250
&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
&show-speaker-label=true
```

Parameter definitions can be found in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

## Streaming transcription output

The following code shows the truncated example response of a streaming request.

```
{
    "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": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
"StartTime": 18.74
}
]
}
}

```

Amazon Transcribe breaks your incoming audio stream based on natural speech segments, such as a change in speaker or a pause in the audio. The transcription is returned progressively to your application, with each response containing more transcribed speech until the entire segment is transcribed. The preceding code is a truncated example of a fully-transcribed speech segment. Speaker labels only appear for entirely transcribed segments.

The following list shows the organization of the objects and parameters in a streaming transcription output.

#### **Transcript**

Each speech segment has its own `Transcript` object.

#### **Results**

Each `Transcript` object has its own `Results` object. This object contains the `isPartial` field. When its value is `false`, the results returned are for an entire speech segment.

### **Alternatives**

Each `Results` object has an `Alternatives` object.

### **Items**

Each `Alternatives` object has its own `Items` object that contains information about each word and punctuation mark in the transcription output. When you enable speaker identification, each word has a `Speaker` label for fully-transcribed speech segments. Amazon Transcribe uses this label to assign a unique integer to each speaker it identifies in the stream. The `Type` parameter having a value of `speaker-change` indicates that one person has stopped speaking and that another person is about to begin.

### **Transcript**

Each `Items` object contains a transcribed speech segment as the value of the `Transcript` field.

# Analyzing call center audio with Call Analytics

Use Amazon Transcribe Call Analytics to gain insight into customer-agent interactions. Call Analytics provides you with:

- [Call characteristics \(p. 155\)](#) (talk time, non-talk time, speaker loudness, interruptions, and talk speed)
- [Call summarization \(p. 155\)](#)
- [Custom categories \(p. 154\)](#) and rules that you can use to hone in on specific keywords and criteria
- [Redaction \(p. 156\)](#) of your text transcript and your audio file
- [Speaker sentiment \(p. 156\)](#)

Call Analytics can also help increase staff productivity by reducing the need for manual note-taking after each call. It also reduces the time required by supervisors to read transcripts or listen to audio recordings.

## Call Analytics use cases

- **Monitor issue frequency over time:** Use [call categorization \(p. 154\)](#) to identify recurring keywords within your transcripts.
- **Gain insight into your customer service experience:** Use [call characteristics \(p. 155\)](#) (non-talk time, talk time, interruptions, voice loudness, talk speed) and sentiment analysis to determine if customers' issues are being appropriately resolved during their call.
- **Ensure regulatory compliance or adherence to company policy:** Set [keywords and phrases \(p. 154\)](#) for company-specific greetings or disclaimers to verify your agents are meeting regulatory requirements.
- **Improve handling of customer data:** Use transcript and audio [redaction \(p. 156\)](#) to protect customer privacy.
- **Improve staff training:** Use criteria (sentiment, non-talk time, interruptions, talk speed) to flag transcripts that can be used as examples of positive or negative customer interactions.
- **Measure staff efficacy in creating a positive customer experience:** Use [sentiment analysis \(p. 156\)](#) to measure if your agents are able to turn a negative customer sentiment into a positive one as calls progress.
- **Improve data organization:** Label and sort calls based on [custom categories \(p. 154\)](#) (including keywords and phrases, sentiment, talk time, and interruptions).
- **Quickly summarize the important aspects of a call** without needing to review the entire transcript: Use automatic [call summarization \(p. 155\)](#) to get a succinct summary of all issues, action items, and outcomes identified in a given call.

To compare the features available with Call Analytics to those for Amazon Transcribe and Amazon Transcribe Medical, refer to the [Amazon Transcribe features \(p. 5\)](#) table.

To get started, see [Starting a Call Analytics job \(p. 163\)](#). Call Analytics output is similar to that of a standard transcription job, but contains additional analytics data. To view Call Analytics output, see [Example output \(p. 168\)](#).

### API operations specific to Call Analytics

[CreateCallAnalyticsCategory](#), [DeleteCallAnalyticsCategory](#), [DeleteCallAnalyticsJob](#),  
[GetCallAnalyticsCategory](#), [GetCallAnalyticsJob](#), [ListCallAnalyticsCategories](#),  
[ListCallAnalyticsJobs](#), [StartCallAnalyticsJob](#), [UpdateCallAnalyticsCategory](#)

## Considerations and additional information

Before using Call Analytics, note that:

- Call Analytics only supports stereo-channel audio; mono-channel audio cannot be used with this feature.
- [Job queuing \(p. 18\)](#) is always enabled, meaning that you are limited to 250 concurrent Call Analytics jobs.
- Input files for Call Analytics cannot be greater than 500 MB and must be less than 4 hours.
- If using categories, you must create all desired categories before starting a Call Analytics job. Any new categories cannot be applied to existing Call Analytics jobs. To learn how to create a new category, see [Creating categories \(p. 156\)](#).
- If you're using redaction, only US English (en-US) is supported. Refer to [Supported languages and language-specific features \(p. 3\)](#) for a complete list of languages supported with Call Analytics.
- Call summarization is only supported for the following English language variants: Australian (en-AU), British (en-GB), Indian (en-IN), Irish (en-IE), Scottish (en-AB), US (en-US), and Welsh (en-WL).
- Some Call Analytics quotas differ from Amazon Transcribe and Amazon Transcribe Medical; refer to [Guidelines and quotas \(p. 7\)](#) for specifics.

#### Tip

To learn more about post-Call Analytics options, see [Post Call Analytics for your contact center with Amazon language AI services](#).

To view sample Call Analytics output and features, check out our [GitHub demo](#). We also have a [JSON to Word document GitHub example](#) for Call Analytics.

## Call Analytics insights

The insights available with Call Analytics are separated into six main groups:

- [Call categorization \(p. 154\)](#)
- [Call characteristics \(p. 155\)](#)
- [Call summarization \(p. 155\)](#)
- [Sensitive data redaction \(p. 156\)](#)
- [Sentiment analysis \(p. 156\)](#)

## Call categorization

Use call categorization to flag keywords, phrases, sentiment, or actions during a call. Our categorization options can help you triage escalations, such as negative-sentiment calls with many interruptions, or organize calls into specific categories, such as company departments.

The criteria you can add to a category include:

- **Non-talk time:** Periods of time when neither the customer nor the agent is talking.

- **Interruptions:** When either the customer or the agent is interrupting the other person.
- **Customer or agent sentiment:** How either the customer or the agent is feeling during a specified time period. If at least 50 percent of the conversation turns (the back-and-forth between two speakers) in a specified time period match the specified sentiment, Amazon Transcribe considers the sentiment a match.
- **Keywords or phrases:** Matches part of the transcription based on an exact phrase. For example, if you set a filter for the phrase "I want to speak to the manager", Amazon Transcribe filters for that *exact* phrase.

Here's an [output example \(p. 168\)](#).

For more information on categories or to learn how to create a new category, see [Creating categories \(p. 156\)](#).

## Call characteristics

The call characteristics feature measures the quality of agent-customer interactions using these criteria:

- **Interruption**—Measures if and when one participant cuts off the other participant mid-sentence. Frequent interruptions may be associated with rudeness or anger, and could correlate to negative sentiment for one or both participants.
- **Loudness**—Measures the volume at which each participant is speaking. Use this metric to see if either the caller or the agent is speaking loudly or yelling, which is often indicative of being upset. This metric is represented as a normalized value (speech level per second of speech in a given segment) on a scale from 0 to 100, where a higher value indicates a louder voice.
- **Non-talk time**—Measures periods of time that do not contain speech. Use this metric to see if there are long periods of silence, such as an agent keeping a customer on hold for an excessive amount of time.
- **Talk speed**—Measures the speed at which both participants are speaking. Comprehension can be affected if one participant speaks too quickly. This metric is measured in words per minute.
- **Talk time**—Measures the amount of time (in milliseconds) each participant spoke during the call. Use this metric to help identify if one participant is dominating the call or if the dialogue is balanced.

Here's an [output example \(p. 168\)](#).

## Call summarization

Call summarization provides succinct summaries of the important components in agent-customer calls, including issues, action items, and outcomes for each participant.

Using call summarization, you can:

- Reduce the need for manual note-taking during and after calls
- Improve agent efficiency, allowing them to respond faster to customers
- Increase customer satisfaction through a reduction in hold times
- Streamline supervisor reviews, as call summaries are much quicker to digest than entire transcripts

Call summarization works across all industries and business sectors, and is context-based.

### Note

Call summarization is supported with these English language variants: Australian (en-AU), British (en-GB), Indian (en-IN), Irish (en-IE), Scottish (en-AB), US (en-US), and Welsh (en-WL).

Call summarization works out-of-the-box, and thus doesn't support customization, such as model training or custom categories.

Here's an [output example \(p. 169\)](#).

## Sensitive data redaction

Sensitive data redaction replaces personally identifiable information (PII) in the text transcript and the audio file. A redacted transcript replaces the original text with [PII]; a redacted audio file replaces spoken personal information with silence. This parameter is useful for protecting customer information.

**Note**

Redaction is supported with US English (en-US) only.

PII redaction works out-of-the-box, and thus doesn't support customization. To view the list of PII that is redacted using this feature, or to learn more about redaction with Amazon Transcribe, see [Redacting or identifying personally identifiable information \(p. 176\)](#).

Here's an [output example \(p. 170\)](#).

## Sentiment analysis

Sentiment analysis estimates how the customer and agent are feeling throughout the call. This metric is represented as both a quantitative value (with a range from 5 to -5) and a qualitative value (positive, neutral, mixed, or negative). Quantitative values are provided per quarter and per call; qualitative values are provided per turn.

Using this parameter, you can quantitatively evaluate the overall sentiment for each call participant and the sentiment for each participant during each quarter of the call. You can qualitatively evaluate turn-by-turn sentiment for each participant. This metric can help identify if your agent is able to delight an upset customer by the time the call ends.

Sentiment analysis works out-of-the-box, and thus doesn't support customization, such as model training or custom categories.

Here's an [output example \(p. 170\)](#).

## Creating categories

Call Analytics supports the creation of custom categories so you can tailor your transcript analysis to your specific business needs.

You can create as many categories as you'd like to cover a range of different scenarios. For each category you create, you must create between 1 and 20 rules. Each rule is based on one of four criteria: interruptions, keywords, non-talk time, or sentiment. For more details on using these criteria with the [CreateCallAnalyticsCategory](#) operation, refer to the [Rule criteria \(p. 161\)](#) section.

Here are a few examples of what you can do with custom categories:

- Isolate calls with specific characteristics, such as calls that end with a negative customer sentiment
- Identify trends in customer issues by flagging and tracking specific sets of keywords
- Monitor compliance, such as an agent speaking a specific phrase during the first few seconds of a call
- Gain insight into customer experience by flagging calls with many agent interruptions and negative customer sentiment

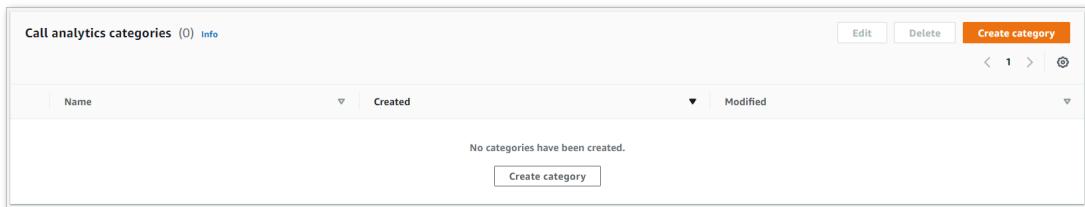
- Compare multiple categories to measure correlations, such as analyzing whether an agent using a welcome phrase correlates with positive customer sentiment

If you run a Call Analytics job and the call characteristics match the rules you've specified in a category, Amazon Transcribe labels your job with that category. See [call categorization output \(p. 168\)](#) for an example of a category match in JSON output.

To create a category, you can use the **AWS Management Console**, **AWS CLI**, or **AWS SDK**; see the following for examples:

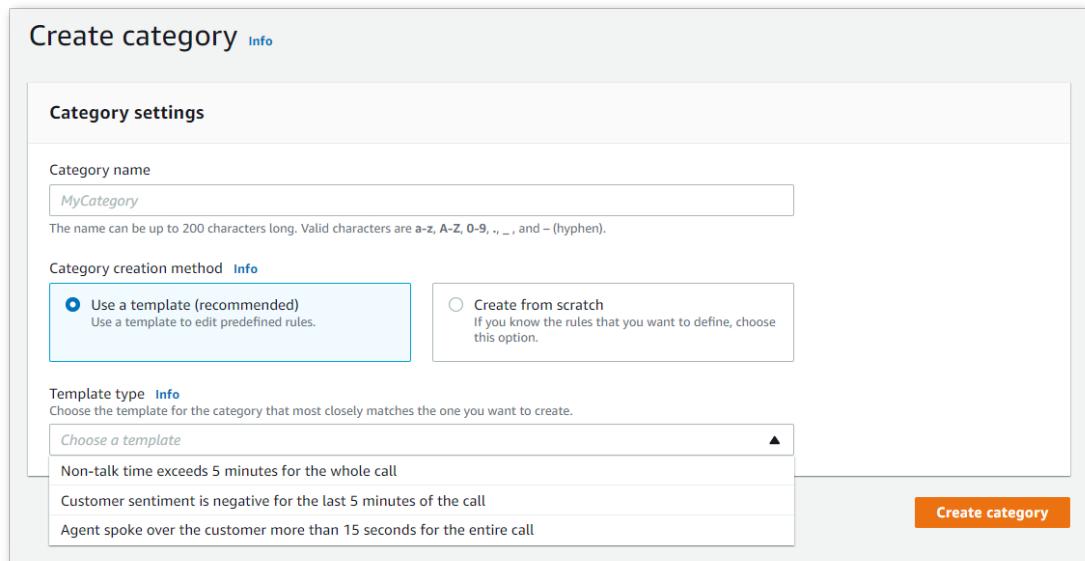
## AWS Management Console

1. In the navigation pane, under Amazon Transcribe, choose **Amazon Transcribe Call Analytics**.
2. Choose **Call analytics categories**, which takes you to the **Call analytics categories** page. Click the **Create category** button.



3. You're now on the **Create category page**. Enter a name for your category, then choose if you're going to use a template to create a category or make one from scratch.

If using a template: select **Use a template (recommended)**, choose the template you want, then select **Create category**.



4. If creating a custom category: select **Create from scratch**.

**Create category Info**

### Category settings

**Category name**  
MyCategory

The name can be up to 200 characters long. Valid characters are a-z, A-Z, 0-9, ., \_, and - (hyphen).

**Category creation method Info**

- Use a template (recommended)  
Use a template to edit predefined rules.
- Create from scratch  
If you know the rules that you want to define, choose this option.

### Rules

All the rule conditions must be met for a transcription job to be classified in this category.

**▼ Rule 1** Delete rule

**Rule type Info**  
Choose the rule that you want to define.

You can add up to 19 more rules.

5. Add rules to your category using the drop-down menu. You can add up to 20 rules per category.

**Rules**  
All the rule conditions must be met for a transcription job to be classified in this category.

**▼ Rule 1** Delete rule

**Rule type Info**  
Choose the rule that you want to define.

<input type="radio"/> Non-talk time	▲
<input type="radio"/> Non-talk time	Trigger the rule when nothing has been said for more than the time that you specify.
<input type="radio"/> Interruption time	Trigger the rule when the speaker has been interrupted for more than the time that you specify.
<input type="radio"/> Transcript content match	Trigger the rule when the speaker says the words or phrases that you specify.
<input type="radio"/> Transcript sentiment match	Trigger the rule when the sentiment of the speaker matches the sentiment that you specify.

You can add up to 19 more rules.

6. Here's an example of a category with two rules: an agent who interrupts a customer for more than 15 seconds during the call and there is negative sentiment by either the customer or the agent in the last two minutes of the call.

**Rules**

All the rule conditions must be met for a transcription job to be classified in this category.

**Rule 1** **Delete rule**

When the duration of the interruption was more than **15** second(s) during the **entire call** when the speaker was **agent**.

**Rule type** **Info**  
Choose the rule that you want to define.

**Logic** **Info**  
Define the conditions that must be met.

When the duration of the interruption was more than **15** second(s)

during the **entire call**

when the speaker was **agent**

**AND**

**Rule 2** **Delete rule**

When the sentiment is negative during the last 2 minute(s) when the speaker was either.

**Rule type** **Info**  
Choose the rule that you want to define.

**Logic** **Info**  
Define the conditions that must be met.

When the sentiment is **negative**

during the **last** **2** minute(s)

when the speaker was **either**

**Add rule**

You can add up to 18 more rules.

7. When you're finished adding rules to your category, choose **Create category**.

## AWS SDK for Python (Boto3)

This example uses the AWS SDK for Python (Boto3) to create a category using the `CategoryName` and `Rules` arguments for the [create\\_call\\_analytics\\_category](#) method. For more information, see [CreateCallAnalyticsCategory](#), [CategoryProperties](#), and [Rule](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
category_name = "example-call-analytics-category"
transcribe.create_call_analytics_category(
    CategoryName=category_name,
    Rules=[
        {
            'SentimentFilter': {
```

```
        'Sentiments': [
            'NEGATIVE'
        ]
    'ParticipantRole': 'AGENT',
}
]
)
result = transcribe.get_call_analytics_category(CategoryName=category_name)
print(result)
```

This example creates a category that filters for a negative agent sentiment.

## AWS CLI

This example uses the [create-call-analytics-category](#) command. For more information, see [CreateCallAnalyticsCategory](#), [CategoryProperties](#), and [Rule](#).

This example creates a category that filters for a negative agent sentiment:

```
aws transcribe create-call-analytics-category \
--category-name my-new-category \
--rules "SentimentFilter={ParticipantRole=AGENT,Sentiments=["NEGATIVE"]}"
```

Here's another example using the [create-call-analytics-category](#) command, and a request body that adds several rules to your category.

```
aws transcribe create-call-analytics-category \
--cli-input-json file://filepath/my-first-analytics-category.json
```

The file *my-first-analytics-category.json* contains the following request body.

```
{
    "CategoryName": "my-new-category",
    "Rules": [
        {
            "InterruptionFilter": {
                "AbsoluteTimeRange": {
                    "First": 60000
                },
                "Negate": true,
                "ParticipantRole": "CUSTOMER",
                "Threshold": 10000
            }
        },
        {
            "NonTalkTimeFilter": {
                "Negate": false,
                "RelativeTimeRange": {
                    "EndPercentage": 80,
                    "StartPercentage": 10
                },
                "Threshold": 20000
            }
        },
        {
            "SentimentFilter": {
                "ParticipantRole": "AGENT",
                "Sentiments": [
                    "NEGATIVE",
                    "POSITIVE"
                ]
            }
        }
    ]
}
```

```

        ],
    },
    {
        "TranscriptFilter": {
            "AbsoluteTimeRange": {
                "First": 10000
            },
            "Targets": [
                "welcome",
                "hello"
            ],
            "TranscriptFilterType": "EXACT"
        }
    }
]
}

```

The preceding example creates a category with the rules:

- The customer was not interrupted in the first 60,000 milliseconds.
- No one was speaking between 10% into the call through 80% into the call.
- The agent had a negative sentiment.
- The words "welcome" or "hello" were used in the first 10,000 milliseconds of the call.

## Rule criteria

This section outlines the types of custom rules you can create using the [CreateCallAnalyticsCategory](#) API operation.

### Interruption match

Rules using interruptions ([InterruptionFilter](#) data type) are designed to match:

- Instances where an agent interrupts a customer
- Instances where a customer interrupts an agent
- Either participant interrupting the other
- A lack of interruptions

Here's an example of the parameters available with [InterruptionFilter](#):

```

[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
},

```

Refer to [CreateCallAnalyticsCategory](#) and [InterruptionFilter](#) for more information on these parameters and the valid values associated with each.

## Keyword match

Rules using keywords ([TranscriptFilter](#) data type) are designed to match:

- Custom words or phrases
- Either the presence or absence of custom words or phrases
- Speech of the agent, the customer, or both
- Custom words or phrases that occur at a specific time frame

Here's an example of the parameters available with [TranscriptFilter](#):

```
"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  
}
```

Refer to [CreateCallAnalyticsCategory](#) and [TranscriptFilter](#) for more information on these parameters and the valid values associated with each.

## Non-talk time match

Rules using non-talk time ([NonTalkTimeFilter](#) data type) are designed to match:

- The presence of silence at specified periods throughout the call
- The presence of speech at specified periods throughout the call

Here's an example of the parameters available with [NonTalkTimeFilter](#):

```
"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  
},
```

Refer to [CreateCallAnalyticsCategory](#) and [NonTalkTimeFilter](#) for more information on these parameters and the valid values associated with each.

## Sentiment match

Rules using sentiment ([SentimentFilter](#) data type) are designed to match:

- Custom words or phrases
- Either the presence or absence of custom words or phrases
- Speech of the agent, the customer, or both
- Custom words or phrases that occur in a specified time frame

Here's an example of the parameters available with [SentimentFilter](#):

```
"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" ]  
},
```

Refer to [CreateCallAnalyticsCategory](#) and [SentimentFilter](#) for more information on these parameters and the valid values associated with each.

## Starting a Call Analytics job

Before running a Call Analytics transcription job, you must create all the [categories \(p. 156\)](#) you want Amazon Transcribe to match in your audio.

### Note

Call Analytics jobs can't be retroactively matched to new categories. Only the categories you create *before* running a Call Analytics job can be applied to that job.

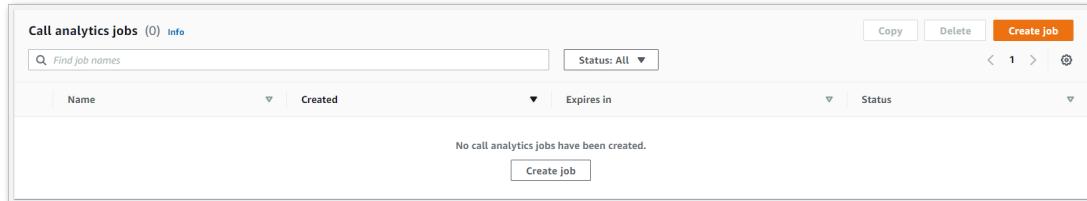
If you've created one or more categories, and your audio matches all the rules within at least one of your categories, Amazon Transcribe flags your output with the matching category. If you choose not to use categories, or if your audio doesn't match the rules specified in your categories, your transcript isn't flagged.

To start a Call Analytics job, you can use the [AWS Management Console](#), [AWS CLI](#), or [AWS SDK](#); see the following for examples:

## AWS Management Console

Use the following procedure to start a Call Analytics job. The calls that match all characteristics defined by a category are labeled with that category.

1. In the navigation pane, under Amazon Transcribe Call Analytics, choose **Call analytics jobs**.
2. Choose **Create job**.



3. On the **Specify job details** page, provide information about your Call Analytics job, including the location of your input data.

**Specify job details** [Info](#)

**Job settings**

Name  [Info](#)  
The name can be up to 200 characters long. Valid characters are a-z, A-Z, 0-9, . (period), \_ (underscore), and – (hyphen).

Model type [Info](#)  
Choose the type of model to use for the transcription job.

General model  
To use a model that is not specialized for a particular use case, choose this option. Configuration options vary between languages.

Custom language model  
To use a model that you trained for your specific use case, choose this option. This model has fewer configuration options than the general model.

Language settings  
You can transcribe your audio file in a language that you specify or have Amazon Transcribe identify and transcribe it in the predominant language.

Specific language [Info](#)  
If you know the language spoken in your source audio, choose this option to get the most accurate results. The options available for additional processing vary between languages.

Automatic language identification [Info](#)  
If you don't know the language spoken in your audio files, choose this option. You have access to fewer options for additional processing than if you choose **Specific language**.

Language  
Choose the language of the input audio.  
 [▼](#)

**Input data** [Info](#)

Input file location on S3  
Choose an input audio or video file in Amazon S3.  
 [Browse S3](#)

Valid file formats: MP3, MP4, WAV, FLAC, AMR, OGG, and WebM.

Agent audio channel identification [Info](#)  
Choose the channel that has the speech from the agent. The other channel is used for the customer's speech.  
 [▼](#)

Specify the desired Amazon S3 location of your output data and which IAM role to use.

### Output data

Output data location type [Info](#)

Service-managed S3 bucket  
The output will be removed after 90 days when the job expires.

Customer specified S3 bucket  
The output will not be removed from bucket even after the job expires.

### Access permissions

IAM role [Info](#)

Use an existing IAM role

Create an IAM role  
By choosing **Create job** you are authorizing creation of this role.

Permissions to access

Your role has access to these resources. The KMS key permission is used only if your input bucket is encrypted

Input S3 bucket and KMS decrypt permission to input bucket

Any S3 bucket and any KMS keys

Role name

Roles are prefixed with "AmazonTranscribeServiceRoleFullAccess-". Your newly created role has full access to the S3 bucket and KMS key for your account.

`MyTranscribeRole`

The name can be up to 64 characters long

**▼ Role permissions details**

Your new role has these permissions to give Amazon Transcribe access to the resources that you've specified.

Service	Access level	Resource
S3	List, Read, Write	All resources
Key Management Service	GenerateDataKey, Decrypt	All resources

[Cancel](#) [Next](#)

4. Choose **Next**.
5. For **Configure job**, turn on any optional features you want to include with your Call Analytics job.

**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.

**Automatic content redaction** [Info](#)  
Automatic content redaction removes personally identifiable information (PII) in your transcripts. Redactions in transcripts show up as [PII].

**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.

**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.

You haven't defined any categories. [Create a new category.](#)

[Cancel](#) [Previous](#) **Create job**

6. Choose **Create job**.

## AWS CLI

This example uses the [start-call-analytics-job](#) command and channel-definitions parameter. For more information, see [StartCallAnalyticsJob](#) and [ChannelDefinition](#).

```
aws transcribe start-call-analytics-job \
--region us-west-2 \
--call-analytics-job-name my-first-call-analytics-job-name \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-media-file.flac \
--output-location DOC-EXAMPLE-BUCKET \
--language-code en-US \
--data-access-role-arn "arn:aws:iam::111122223333:role/ExampleRole" \
--channel-definitions '[{"ChannelId": 0, "ParticipantRole": "AGENT"}, {"ChannelId": 1, "ParticipantRole": "CUSTOMER"}]'
```

Here's another example using the [start-call-analytics-job](#) command, and a request body that identifies audio channels for that job.

```
aws transcribe start-call-analytics-job \
--cli-input-json file://filepath/my-first-analytics-job.json
```

The file *my-first-analytics-job.json* contains the following request body.

```
{  
    "CallAnalyticsJobName": "my-first-call-analytics-job-name",  
    "OutputLocation": "s3://DOC-EXAMPLE-BUCKET/",  
    "DataAccessRoleArn": "arn:aws:iam::111122223333:role/ExampleRole",  
    "Media": {  
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"  
    },  
    "ChannelDefinitions": [  
        {  
            "ChannelId": 0,  
            "ParticipantRole": "AGENT"  
        },  
        {  
            "ChannelId": 1,  
            "ParticipantRole": "CUSTOMER"  
        }  
    ]  
}
```

## AWS SDK for Python (Boto3)

This example uses the AWS SDK for Python (Boto3) to start a Call Analytics job using the [start\\_call\\_analytics\\_job](#) method. For more information, see [StartCallAnalyticsJob](#) and [ChannelDefinition](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

```
from __future__ import print_function  
import time  
import boto3  
transcribe = boto3.client('transcribe', 'us-west-2')  
job_name = "my-first-call-analytics-job"  
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"  
output_location = "s3://DOC-EXAMPLE-BUCKET/"  
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']:br/>        break  
    print("Not ready yet...")  
    time.sleep(5)  
print(status)
```

## Example output

Call Analytics transcripts are displayed in a turn-by-turn format. Each segment includes data specific to that segment, which can include detected issues, action items, detected outcomes, loudness scores, redaction, and sentiment. Additionally, a summary of call characteristics is provided at the end of the transcript.

To increase accuracy and further customize your transcripts to your use case, such as including industry-specific terms, use [custom vocabularies \(p. 64\)](#) or [custom language models \(p. 107\)](#) with your Call Analytics job. To mask, remove, or tag words you don't want in your transcription results, such as profanity, add [vocabulary filtering \(p. 118\)](#).

The following sections show examples of redacted JSON output, separated by Call Analytics insight. An abridged, but otherwise complete, transcript is provided following the insight-specific examples.

## Call categorization

Here's what a category match looks like in your transcription output. This example shows that the audio from the 40040 millisecond time stamp to the 42460 millisecond timestamp is a match to the 'positive-resolution' category. In this case, the custom 'positive-resolution' category required a positive sentiment in last few seconds of speech.

```
"Categories": {  
    "MatchedDetails": {  
        "positive-resolution": {  
            "PointsOfInterest": [  
                {  
                    "BeginOffsetMillis": 40040,  
                    "EndOffsetMillis": 42460  
                }  
            ]  
        }  
    },  
    "MatchedCategories": [  
        "positive-resolution"  
    ]  
},
```

## Call characteristics

Here's what call characteristics look like in your transcription output. Note that loudness scores are provided for each conversation turn, while all other characteristics are provided at the end of the transcript.

```
"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
    }
},

```

## Call summarization

Here's what call summarization looks like in your transcription output:

- In the following example, **issues** are identified as starting at character 7 and ending at character 51, which refers to this section of the text: "*I would like to cancel my recipe subscription*".

```

    "Content": "Well, I would like to cancel my recipe subscription.",
    "IssuesDetected": [
        {
            "CharacterOffsets": {

```

```

        "Begin": 7,
        "End": 51
    }
],

```

- In the following example, **outcomes** are identified as starting at character 12 and ending at character 78, which refers to this section of the text: "*I made all changes to your account and now this discount is applied*".

```

"Content": "Wonderful. I made all changes to your account and now this discount is
applied, please check.",

"OutcomesDetected": [
{
    "CharacterOffsets": {
        "Begin": 12,
        "End": 78
    }
},
]

```

- In the following example, **action items** are identified as starting at character 0 and ending at character 103, which refers to this section of the text: "*I will send an email with all the details to you today, and I will call you back next week to follow up*".

```

"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
    }
},
]

```

## Sensitive data redaction

Here's what sensitive data redaction looks like in your transcription output:

```
"Content": "[PII], my name is [PII], how can I help?",
```

## Sentiment analysis

Here's what sentiment analysis looks like in your transcription output.

- Qualitative turn-by-turn sentiment values:

```

"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"

```

- Quantitative sentiment values for the entire call:

```

"Sentiment": {
    "OverallSentiment": {
        "AGENT": 2.5,
        "CUSTOMER": 2.1
    },
}

```

- Quantitative sentiment values per participant and per call quarter:

```

"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
    }
]
}
```

## Compiled output

For brevity, some content is replaced with ellipses in the below transcription output.

```
{
    "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
}
},
```

# Redacting or identifying personally identifiable information

Redaction is used to mask or remove sensitive content, in the form of personally identifiable information (PII), from your transcripts. Amazon Transcribe can redact information with batch jobs and streaming transcriptions. Additionally, if you're performing a streaming transcription, you also have the option to flag PII without redacting it; see [Example PII identification output \(p. 186\)](#) for an example.

## Types of PII Amazon Transcribe can recognize

PII type	Description
Bank Account Number	A number that uniquely identifies a bank account.
Bank Routing Number	A number that identifies the location of a bank account.
Credit Card Number or Debit Card Number	A value that uniquely defines a payment card issued by a bank.
Credit Card or Debit Card CVV Code	A 3-digit or 4-digit security code on each credit card.
Credit Card or Debit Card Expiration Date	The month and year a card expires.
Debit or credit card PIN	A security code issued by a bank or credit union. This number is used for bank accounts and payment cards.
Email address	The unique identifier of an email box where messages are delivered.
US mailing address	The United States mailing address of an individual.
Name	The first and last name of a person.
US phone number	A 10-digit phone number within the United States.
Social Security Number	A 9-digit number (or the last 4 digits of that number). Issued to US citizens, permanent residents, and temporary residents with employment.
All PII	Redact or identify all PII types listed in this table.

When redaction is enabled, you have the option to generate only a redacted transcript or both a redacted transcript and an unredacted transcript. If you choose to generate only a redacted transcript, note that your media is the only place where the complete conversation is stored. If you delete your original media, there is no record of the unredacted PII. Because of this, it may be prudent to generate an unredacted transcript in addition to a redacted one.

To learn more about PII redaction with batch transcriptions, refer to: [Redacting PII in your batch job \(p. 177\)](#).

To learn more about PII redaction or identification with streaming transcriptions, refer to: [Redacting or identifying PII in a real-time stream \(p. 180\)](#).

**Important**

The redaction feature is designed to identify and remove sensitive data. However, due to the predictive nature of machine learning, Amazon Transcribe may not identify and remove all instances of sensitive data in your transcript. We strongly recommend that you review any redacted output to ensure it meets your needs.

The redaction feature does not meet the requirements for de-identification under medical privacy laws, such as the U.S. Health Insurance Portability and Accountability Act of 1996 (HIPAA).

Redaction is supported with US English (en-US) only.

For a video walkthrough of the Amazon Transcribe's redaction feature, see [Use content redaction to identify & redact PII](#).

## Redacting PII in your batch job

When redacting personally identifiable information (PII) from a transcript during a batch transcription job, Amazon Transcribe replaces each identified instance of PII with [PII] in the main text body of your transcript. You can also view the type of PII that was redacted in the word-for-word portion of the transcription output. For an output sample, see [Example redacted batch output \(p. 183\)](#).

Both redacted and unredacted transcripts are stored in the same output Amazon S3 bucket. Amazon Transcribe stores them in either a bucket you specify or in the default Amazon S3 bucket managed by the service.

You can start a batch transcription job using the AWS Management Console, AWS CLI, or AWS SDK.

### AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Transcription jobs**, then select the **Create job** button (top right). This will open the **Specify job details** page.
3. After filling in your desired fields on the **Specify job details** page, click the **Next** button to go to the **Configure job - optional** page. Here you'll find the **Content removal** panel with the **PII redaction** toggle.

## Configure job - *optional* Info

### Audio settings

#### Audio identification Info

Choose to split multi-channel audio into separate channels for transcription, or identify speakers in the input audio.

#### Alternative results Info

Enable to view more transcription results

### 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].

- Once you select **PII redaction**, you have the option to select all PII types you want to redact. You can also choose to have an unredacted transcript if you select **Include unredacted transcript in job output** box.

### 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].

#### Include unredacted transcript in job output

Returns unredacted version of the transcript in addition to the redacted version.

#### Select PII entity types (11 of 11 selected)

##### Select All

##### Financial (6 of 6 selected)

<input checked="" type="checkbox"/> BANK_ACCOUNT_NUMBER	<input checked="" type="checkbox"/> BANK_ROUTING	<input checked="" type="checkbox"/> CREDIT_DEBIT_NUMBER
<input checked="" type="checkbox"/> CREDIT_DEBIT_CVV	<input checked="" type="checkbox"/> CREDIT_DEBIT_EXPIRY	<input checked="" type="checkbox"/> PIN

##### Personal (5 of 5 selected)

<input checked="" type="checkbox"/> NAME	<input checked="" type="checkbox"/> ADDRESS	<input checked="" type="checkbox"/> PHONE
<input checked="" type="checkbox"/> EMAIL	<input checked="" type="checkbox"/> SSN	

#### Vocabulary filtering Info

Vocabulary filtering can remove, mask or tag specified words in the final transcript.

5. Click the **Create job** button to run your transcription job.

## AWS CLI

This example uses the [start-transcription-job](#) command and `content-redaction` parameter. For more information, see [StartTranscriptionJob](#) and [ContentRedaction](#).

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--language-code en-US \
--content-redaction
    RedactionType=PII,RedactionOutput=redacted,PiiEntityTypes=NAME,ADDRESS,BANK_ACCOUNT_NUMBER
```

Here's another example using the [start-transcription-job](#) method, and the request body redacts PII for that job.

```
aws transcribe start-transcription-job \
--cli-input-json file://filepath/my-first-redaction-job.json
```

The file *my-first-redaction-job.json* contains the following request body.

```
{
    "TranscriptionJobName": "my-first-transcription-job",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
    },
    "OutputBucketName": "DOC-EXAMPLE-BUCKET",
    "LanguageCode": "en-US",
    "ContentRedaction": {
        "RedactionOutput": "redacted",
        "RedactionType": "PII",
        "PiiEntityTypes": [
            "NAME",
            "ADDRESS",
            "BANK_ACCOUNT_NUMBER"
        ]
    }
}
```

## AWS SDK for Python (Boto3)

This example uses the AWS SDK for Python (Boto3) to redact content using the `ContentRedaction` argument for the `start_transcription_job` method. For more information, see [StartTranscriptionJob](#) and [ContentRedaction](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac"
```

```
transcribe.start_transcription_job(
    TranscriptionJobName=job_name,
    Media={
        'MediaFileUri': job_uri
    },
    MediaFormat='flac',
    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

PII redaction for batch jobs is only supported in these AWS Regions: Asia Pacific (Hong Kong), Asia Pacific (Mumbai), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), GovCloud (US-West), Canada (Central), EU (Frankfurt), EU (Ireland), EU (London), EU (Paris), Middle East (Bahrain), South America (Sao Paulo), US East (N. Virginia), US East (Ohio), US West (Oregon), and US West (N. California).

## Redacting or identifying PII in a real-time stream

When redacting personally identifiable information (PII) from a streaming transcription, Amazon Transcribe replaces each identified instance of PII with [PII] in your transcript.

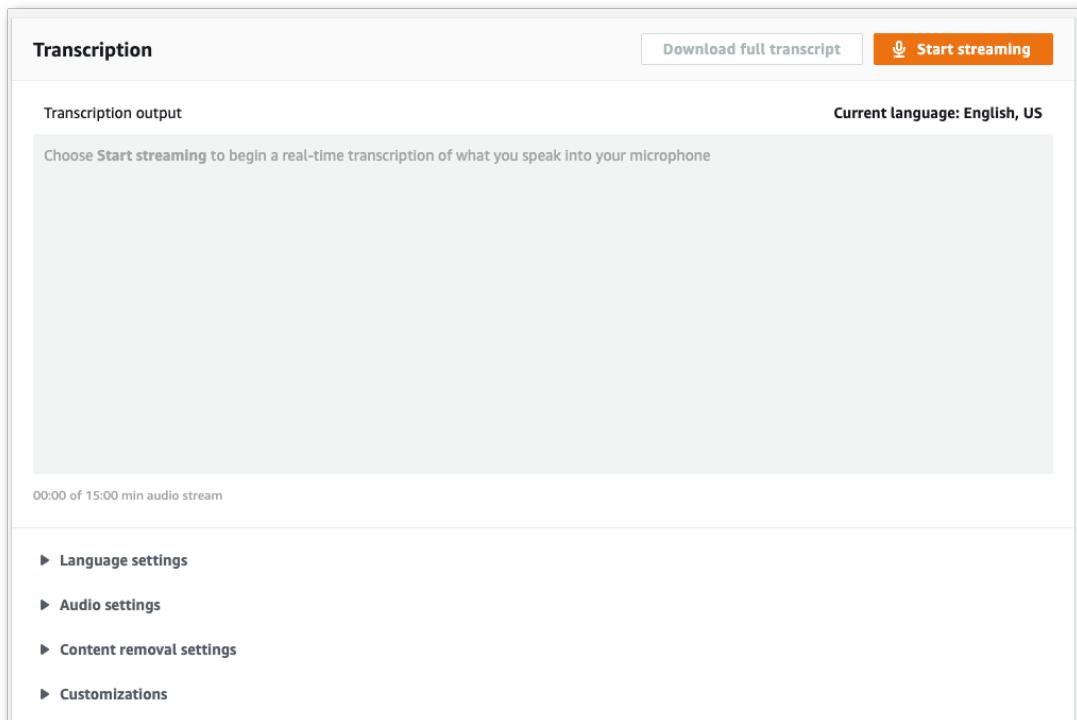
An additional option available for streaming audio is *PII identification*. When you activate PII Identification, Amazon Transcribe labels the PII in your transcription results under an `Entities` object. For an output sample, see [Example redacted streaming output \(p. 185\)](#) and [Example PII identification output \(p. 186\)](#).

PII identification and redaction for streaming jobs is performed only upon complete transcription of the audio segments.

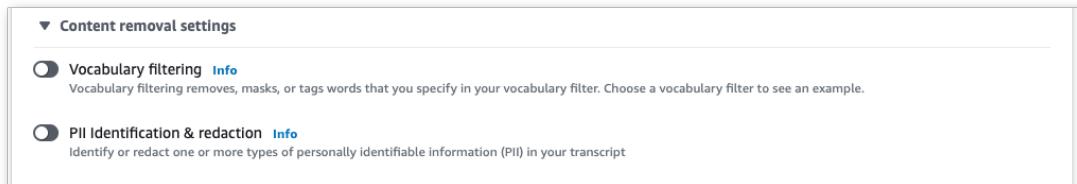
You can start a streaming transcription using the AWS Management Console, WebSocket, or HTTP/2.

### AWS Management Console

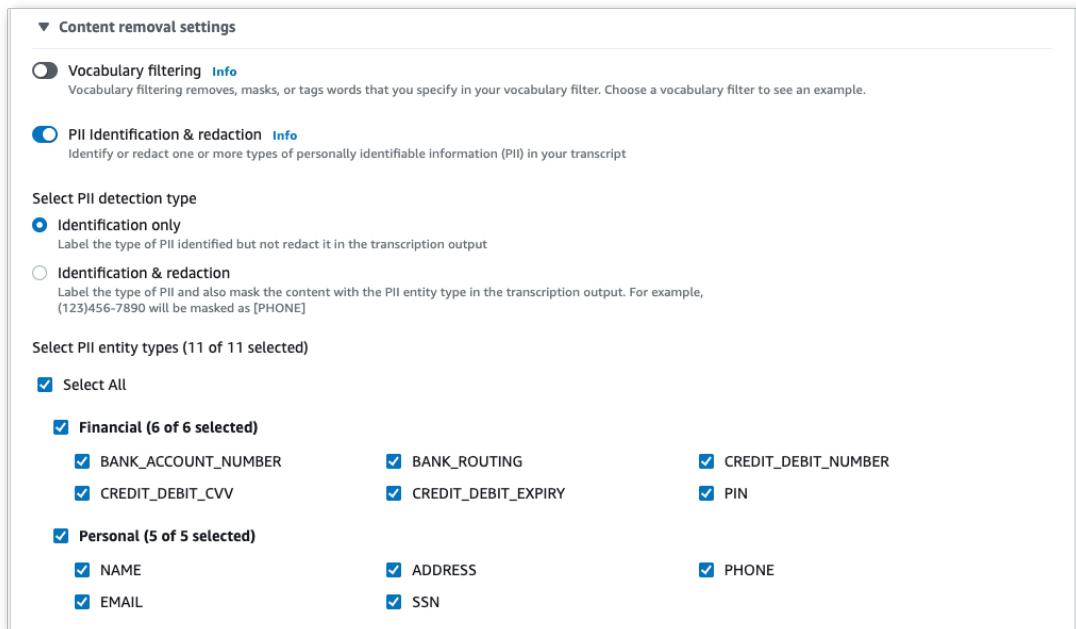
1. Sign into the [AWS Management Console](#).
2. In the navigation pane, choose **Real-time transcription**. Scroll down to **Content removal settings** and expand this field if it is minimized.



3. Toggle on **PII Identification & redaction**.



4. Select either **Identification only** or **Identification & redaction**, then select the PII entity types you want to identify or redact in your transcript.



5. You're now ready to transcribe your stream. Select the **Start streaming** button and begin speaking. To end your dictation, select **Stop streaming**.

## WebSocket stream

This example creates a pre-signed URL that uses either PII Identification or PII redaction in a WebSocket stream. Line breaks have been added for readability. For more information on using WebSocket streams with Amazon Transcribe, see [Setting up a WebSocket stream \(p. 54\)](#). For more detail on parameters, see [StartStreamTranscription](#).

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=250
&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)
```

You cannot use both `content-identification-type` and `content-redaction-type` in the same request.

Parameter definitions can be found in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

## HTTP/2 stream

This example creates an HTTP/2 request with PII identification or PII redaction enabled. For more information on using HTTP/2 streaming with Amazon Transcribe, see [Setting up an HTTP/2](#)

stream (p. 50). For more detail on parameters and headers specific to Amazon Transcribe, see [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
```

You cannot use both `content-identification-type` and `content-redaction-type` in the same request.

Parameter definitions can be found in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

**Note**

PII redaction for streaming is only supported in these AWS Regions: Asia Pacific (Seoul), Asia Pacific (Sydney), Asia Pacific (Tokyo), Canada (Central), EU (Frankfurt), EU (Ireland), EU (London), US East (N. Virginia), US East (Ohio), and US West (Oregon).

## Example PII redaction and identification output

The following examples show redacted output from batch and streaming jobs, and PII identification from a streaming job.

Transcription jobs using content redaction generate two types of confidence values. The Automatic Speech Recognition (ASR) confidence indicates the items that have the type of pronunciation or punctuation is a specific utterance. In the following transcript output, the word Good has a confidence of 1.0. This confidence value indicates that Amazon Transcribe is 100 percent confident that the word uttered in this transcript is 'Good'. The confidence value for a [ PII ] tag is the confidence that the speech it flagged for redaction is truly PII. In the following transcript output, the confidence of 0.9999 indicates that Amazon Transcribe is 99.99 percent confident that the entity it redacted in the transcript is PII.

## Example redacted batch output

```
{
    "jobName": "job id",
    "accountId": "account id",
    "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
```

```

[PII].           Security number [PII]. My credit card number is [PII] and my C V V code is
                I hope that Amazon Transcribe is doing a good job at redacting that
personal          information away. Let's check."
                }
],
"items": [
{
    "start_time": "2.86",
    "end_time": "3.35",
    "alternatives": [
        {
            "confidence": "1.0",
            "content": "Good"
        }
    ],
    "type": "pronunciation"
},
Items removed for brevity
{
    "start_time": "5.56",
    "end_time": "6.25",
    "alternatives": [
        {
            "content": "[PII]",
            "redactions": [
                {
                    "confidence": "0.9999",
                    "type": "NAME",
                    "category": "PII"
                }
            ]
        }
    ],
    "type": "pronunciation"
},
Items removed for brevity
],
},
"status": "COMPLETED"
}

```

Here's the unredacted transcript for comparison:

```

{
    "jobName": "job id",
    "accountId": "account id",
    "isRedacted": false,
    "results": {
        "transcripts": [
            {
                "transcript": "Good morning, everybody. My name is Mike, and today I feel
like
Social
job
                sharing a whole lot of personal information with you. Let's start with my
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
                at redacting that personal information away. Let's check."
            }
        ],
        "items": [
            {
                "start_time": "2.86",

```

```

    "end_time": "3.35",
    "alternatives": [
        {
            "confidence": "1.0",
            "content": "Good"
        }
    ],
    "type": "pronunciation"
},
Items removed for brevity
{
    "start_time": "5.56",
    "end_time": "6.25",
    "alternatives": [
        {
            "confidence": "0.9999",
            "content": "Mike",
            {
            },
            "type": "pronunciation"
},
Items removed for brevity
],
},
"status": "COMPLETED"
}

```

## Example redacted streaming output

```
{
    "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": "2db76dc8-d728-11e8-9f8b-f2801f1b9fd1",
    "StartTime": 0.0199375
}
]
}
}
```

## Example PII identification output

PII identification is an additional feature that you can use with your streaming transcription job.

```
{
    "TranscriptResultStream": {
        "TranscriptEvent": {
            "Transcript": {
                "Results": [
                    {
                        "Alternatives": [
                            {
                                "Transcript": "my name is janet smithy",
                                "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": "janet",
                                        "EndTime": 0.9199375,
                                        "StartTime": 0.7999375,
                                        "Type": "pronunciation"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    }
}
```

```
        "Content": "smithy",
        "EndTime": 1.0199375,
        "StartTime": 0.9299375,
        "Type": "pronunciation"
    }
],
"Entities": [
{
    "Content": "janet smithy",
    "Category": "PII",
    "Type": "NAME",
    "StartTime": 0.7999375,
    "EndTime": 1.0199375,
    "Confidence": 0.9989
}
]
},
"EndTime": 1.02,
"IsPartial": false,
"ResultId": "2db76dc8-d728-11e8-9f8b-f2801f1b9fd1",
"StartTime": 0.0199375
}
]
}
}
```

# Creating video subtitles

Amazon Transcribe supports WebVTT (\*.vtt) and SubRip (\*.srt) output for use as video subtitles. You can select one or both file types when setting up your batch video transcription job. When using the subtitle feature, your selected subtitle file(s) and a regular transcript file (containing additional information) are produced. Subtitle and transcription files are output to the same destination.

Subtitles are displayed at the same time text is spoken, and remain visible until there is a natural pause or the speaker finishes talking.

**Important**

Amazon Transcribe uses a default start index of 0 for subtitle output, which differs from the more widely used value of 1. If you require a start index of 1, you can specify this in the AWS Management Console or in your API request using the [OutputStartIndex](#) parameter.

Using the incorrect start index can result in compatibility errors with other services, so be sure to verify which start index you require before creating your subtitles. If you're uncertain which value to use, we recommend choosing 1. Refer to [Subtitles](#) for more information.

Features supported with subtitles:

- **Content redaction** — Any redacted content is reflected as 'PII' in both your subtitle and regular transcript output files. The audio is not altered.
- **Vocabulary filters** — Subtitle files are generated from the transcription file, so any words you filter in your standard transcription output are also filtered in your subtitles. Filtered content is displayed as whitespace or \*\*\* in your transcript and subtitle files. The audio is not altered.
- **Speaker diarization** — If there are multiple speakers in a given subtitle segment, dashes are used to distinguish each speaker. This applies to both WebVTT and SubRip formats; for example:
  - -- Text spoken by Person 1
  - -- Text spoken by Person 2

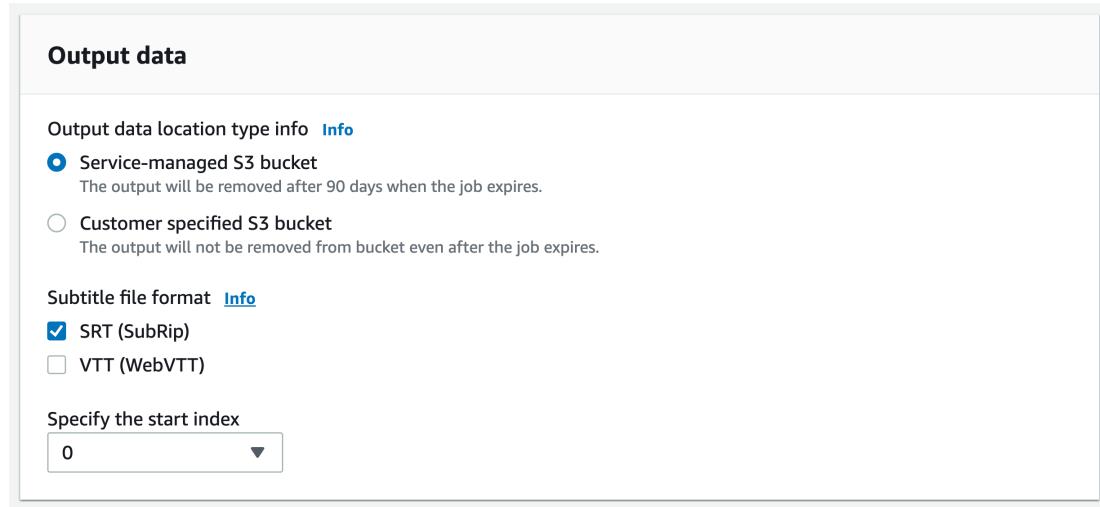
Subtitle files are stored in the same Amazon S3 location as your transcription output.

## Adding subtitles to your Amazon Transcribe video files

You can add subtitles using the [AWS Management Console](#), [AWS SDK](#), or [AWS CLI](#); see the following for examples:

### AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Transcription jobs**, then select the **Create job** button (top right). This opens the **Specify job details** page. Subtitle options are located in the **Output data** panel.
3. Select the formats you want for your subtitles files, then choose a value for your start index. Note that the Amazon Transcribe default is 0, but 1 is more widely used. If you're uncertain which value to use, we recommend choosing 1, as this may improve compatibility with other services.



4. Fill in any other fields you wish to include on the **Specify job details** page, then click the **Next** button. This takes you to the **Configure job - optional** page.
5. Click the **Create job** button to run your transcription job.

## AWS CLI

This example uses the `start-transcription-job` command and `Subtitles` parameter. For more information, see [StartTranscriptionJob](#) and [Subtitles](#).

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name your-transcription-job-name \
--media MediaFileUri=s3://DOC-EXAMPLE-BUCKET/my-media-file.flac \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--language-code en-US \
--subtitles Formats=vtt OutputStartIndex=1
```

Here's another example using the `start-transcription-job` command, and a request body that adds subtitles to that job.

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

The file *my-first-subtitle-job.json* contains the following request body.

```
{ "TranscriptionJobName": "your-transcription-job-name", "Media": { "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-media-file.flac" }, "OutputBucketName": "DOC-EXAMPLE-BUCKET", "LanguageCode": "en-US", "Subtitles": { "Formats": [ "vtt" ], "OutputStartIndex": 1 } }
```

```
}
```

## AWS SDK for Python (Boto3)

This example uses the AWS SDK for Python (Boto3) to add subtitles using the `Subtitles` argument for the `start_transcription_job` method. For more information, see [StartTranscriptionJob](#) and [Subtitles](#).

For additional examples using the AWS SDKs, including feature-specific, scenario, and cross-service examples, refer to the [Code examples for Amazon Transcribe using AWS SDKs \(p. 191\)](#) chapter.

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

# Code examples for Amazon Transcribe using AWS SDKs

The following code examples show how to use Amazon Transcribe with an AWS software development kit (SDK).

The examples are divided into the following categories:

## **Actions**

Code excerpts that show you how to call individual service functions.

## **Scenarios**

Code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## **Cross-service examples**

Sample applications that work across multiple AWS services.

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## **Code examples**

- [Actions for Amazon Transcribe using AWS SDKs \(p. 192\)](#)
  - [Create a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 192\)](#)
  - [Delete a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 193\)](#)
  - [Delete a Amazon Transcribe Medical transcription job using an AWS SDK \(p. 194\)](#)
  - [Delete an Amazon Transcribe transcription job using an AWS SDK \(p. 195\)](#)
  - [Get a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 196\)](#)
  - [Get an Amazon Transcribe transcription job using an AWS SDK \(p. 197\)](#)
  - [List custom Amazon Transcribe vocabularies using an AWS SDK \(p. 197\)](#)
  - [List Amazon Transcribe Medical transcription jobs using an AWS SDK \(p. 198\)](#)
  - [List Amazon Transcribe transcription jobs using an AWS SDK \(p. 199\)](#)
  - [Start an Amazon Transcribe Medical transcription job using an AWS SDK \(p. 201\)](#)
  - [Start an Amazon Transcribe transcription job using an AWS SDK \(p. 202\)](#)
  - [Update a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 203\)](#)
- [Scenarios for Amazon Transcribe using AWS SDKs \(p. 204\)](#)
  - [Create and refine an Amazon Transcribe custom vocabulary using an AWS SDK \(p. 204\)](#)
  - [Transcribe audio and get job data with Amazon Transcribe using an AWS SDK \(p. 210\)](#)
- [Cross-service examples for Amazon Transcribe using AWS SDKs \(p. 211\)](#)
  - [Build an Amazon Transcribe app \(p. 211\)](#)
  - [Build an Amazon Transcribe streaming app \(p. 211\)](#)

- Convert text to speech and back to text using an AWS SDK (p. 212)

## Actions for Amazon Transcribe using AWS SDKs

The following code examples demonstrate how to perform individual Amazon Transcribe actions with AWS SDKs. These excerpts call the Amazon Transcribe API and are not intended to be run in isolation. Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

The following examples include only the most commonly used actions. For a complete list, see the [Amazon Transcribe API Reference](#).

### Examples

- Create a custom Amazon Transcribe vocabulary using an AWS SDK (p. 192)
- Delete a custom Amazon Transcribe vocabulary using an AWS SDK (p. 193)
- Delete a Amazon Transcribe Medical transcription job using an AWS SDK (p. 194)
- Delete an Amazon Transcribe transcription job using an AWS SDK (p. 195)
- Get a custom Amazon Transcribe vocabulary using an AWS SDK (p. 196)
- Get an Amazon Transcribe transcription job using an AWS SDK (p. 197)
- List custom Amazon Transcribe vocabularies using an AWS SDK (p. 197)
- List Amazon Transcribe Medical transcription jobs using an AWS SDK (p. 198)
- List Amazon Transcribe transcription jobs using an AWS SDK (p. 199)
- Start an Amazon Transcribe Medical transcription job using an AWS SDK (p. 201)
- Start an Amazon Transcribe transcription job using an AWS SDK (p. 202)
- Update a custom Amazon Transcribe vocabulary using an AWS SDK (p. 203)

## Create a custom Amazon Transcribe vocabulary using an AWS SDK

The following code example shows how to create a custom Amazon Transcribe vocabulary.

Python

### SDK for Python (Boto3)

```
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
```

- Find instructions and more code on [GitHub](#).
- For API details, see [CreateVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Delete a custom Amazon Transcribe vocabulary using an AWS SDK

The following code example shows how to delete a custom Amazon Transcribe vocabulary.

Python

### SDK for Python (Boto3)

```
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
```

- Find instructions and more code on [GitHub](#).
- For API details, see [DeleteVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

# Delete a Amazon Transcribe Medical transcription job using an AWS SDK

The following code example shows how to delete an Amazon Transcribe Medical transcription job.

JavaScript

## SDK for JavaScript V3

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create anAmazon EC2 service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Delete a medical transcription job.

```
// 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_transcription_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();
```

- Find instructions and more code on [GitHub](#).
- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteMedicalTranscriptionJob](#) in [AWS SDK for JavaScript API Reference](#).

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

# Delete an Amazon Transcribe transcription job using an AWS SDK

The following code examples show how to delete an Amazon Transcribe transcription job.

JavaScript

## SDK for JavaScript V3

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create anAmazon EC2 service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Delete a transcription job.

```
// 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, 'transcription_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();
```

- Find instructions and more code on [GitHub](#).
- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteTranscriptionJob](#) in [AWS SDK for JavaScript API Reference](#).

Python

## SDK for Python (Boto3)

```
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
```

- Find instructions and more code on [GitHub](#).
- For API details, see [DeleteTranscriptionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Get a custom Amazon Transcribe vocabulary using an AWS SDK

The following code example shows how to get a custom Amazon Transcribe vocabulary.

Python

### SDK for Python (Boto3)

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a customer 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
```

- Find instructions and more code on [GitHub](#).
- For API details, see [GetVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Get an Amazon Transcribe transcription job using an AWS SDK

The following code example shows how to get an Amazon Transcribe transcription job.

Python

### SDK for Python (Boto3)

```
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
```

- Find instructions and more code on [GitHub](#).
- For API details, see [GetTranscriptionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## List custom Amazon Transcribe vocabularies using an AWS SDK

The following code example shows how to list custom Amazon Transcribe vocabularies.

Python

### SDK for Python (Boto3)

```
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)
vocab = response['Vocabularies']
next_token = response.get('NextToken')
while next_token is not None:
    response = transcribe_client.list_vocabularies(
        NameContains=vocabulary_filter, NextToken=next_token)
    vocab += response['Vocabularies']
    next_token = response.get('NextToken')
logger.info(
    "Got %s vocabularies with filter %s.", len(vocab), vocabulary_filter)
except ClientError:
    logger.exception(
        "Couldn't list vocabularies with filter %s.", vocabulary_filter)
    raise
else:
    return vocab
```

- Find instructions and more code on [GitHub](#).
- For API details, see [ListVocabularies](#) in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## List Amazon Transcribe Medical transcription jobs using an AWS SDK

The following code example shows how to list Amazon Transcribe Medical transcription jobs.

JavaScript

### SDK for JavaScript V3

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create anAmazon EC2 service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

List medical transcription jobs.

```
// 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();
```

- Find instructions and more code on [GitHub](#).
- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListMedicalTranscriptionJobs](#) in [AWS SDK for JavaScript API Reference](#).

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## List Amazon Transcribe transcription jobs using an AWS SDK

The following code examples show how to list Amazon Transcribe transcription jobs.

JavaScript

### SDK for JavaScript V3

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create anAmazon EC2 service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

List transcription jobs.

```
// 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();
```

- Find instructions and more code on [GitHub](#).
- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTranscriptionJobs](#) in [AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

```
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
```

- Find instructions and more code on [GitHub](#).
- For API details, see [ListTranscriptionJobs](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

# Start an Amazon Transcribe Medical transcription job using an AWS SDK

The following code example shows how to start an Amazon Transcribe Medical transcription job.

JavaScript

## SDK for JavaScript V3

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create anAmazon EC2 service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Start a medical transcription job.

```
// 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();
```

- Find instructions and more code on [GitHub](#).
- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [StartMedicalTranscriptionJob](#) in [AWS SDK for JavaScript API Reference](#).

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Start an Amazon Transcribe transcription job using an AWS SDK

The following code examples show how to start an Amazon Transcribe transcription job.

JavaScript

### SDK for JavaScript V3

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create anAmazon EC2 service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Start a transcription job.

```
// 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"
    },
};

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();
```

- Find instructions and more code on [GitHub](#).
- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [StartTranscriptionJob in AWS SDK for JavaScript API Reference](#).

Python

**SDK for Python (Boto3)**

```
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
```

- Find instructions and more code on [GitHub](#).
- For API details, see [StartTranscriptionJob in AWS SDK for Python \(Boto3\) API Reference](#).

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Update a custom Amazon Transcribe vocabulary using an AWS SDK

The following code example shows how to update a custom Amazon Transcribe vocabulary.

Python

**SDK for Python (Boto3)**

```
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
```

- Find instructions and more code on [GitHub](#).
- For API details, see [UpdateVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Scenarios for Amazon Transcribe using AWS SDKs

The following code examples show you how to implement common scenarios in Amazon Transcribe with AWS SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within Amazon Transcribe. Each scenario includes a link to GitHub, where you can find instructions on how to set up and run the code.

### Examples

- [Create and refine an Amazon Transcribe custom vocabulary using an AWS SDK \(p. 204\)](#)
- [Transcribe audio and get job data with Amazon Transcribe using an AWS SDK \(p. 210\)](#)

## Create and refine an Amazon Transcribe custom vocabulary using an AWS SDK

The following code example shows how to create and refine an Amazon Transcribe custom vocabulary to improve transcription accuracy.

## Python

### SDK for Python (Boto3)

Transcribe an audio file that contains a reading of Jabberwocky by Lewis Carroll. Start by creating functions that wrap Amazon Transcribe actions.

```
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 customer 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

```

Call the wrapper functions to transcribe audio without a custom vocabulary and then with different versions of a custom vocabulary to see improved results.

```

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'][0]['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_vocab = list_vocabularies('Jabber', transcribe_client)
print(f"Found {len(jabber_vocab)} vocabularies:")
for vocab_sum in jabber_vocab:
    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!")

```

- Find instructions and more code on [GitHub](#).

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Transcribe audio and get job data with Amazon Transcribe using an AWS SDK

The following code example shows how to transcribe audio and get job data with Amazon Transcribe.

For more information, see [Getting started with Amazon Transcribe](#).

Python

### SDK for Python (Boto3)

```
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()
```

- Find instructions and more code on [GitHub](#).
- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [GetTranscriptionJob](#)

- [StartTranscriptionJob](#)

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Cross-service examples for Amazon Transcribe using AWS SDKs

The following sample applications use AWS SDKs to combine Amazon Transcribe with other AWS services. Each example includes a link to GitHub, where you can find instructions on how to set up and run the application.

### Examples

- [Build an Amazon Transcribe app \(p. 211\)](#)
- [Build an Amazon Transcribe streaming app \(p. 211\)](#)
- [Convert text to speech and back to text using an AWS SDK \(p. 212\)](#)

## Build an Amazon Transcribe app

The following code example shows how to use Amazon Transcribe to transcribe and display voice recordings in the browser.

JavaScript

### SDK for JavaScript V3

Create an app that uses Amazon Transcribe to transcribe and display voice recordings in the browser. The app uses two Amazon Simple Storage Service (Amazon S3) buckets, one to host the application code, and another to store transcriptions. The app uses an Amazon Cognito user pool to authenticate your users. Authenticated users have AWS Identity and Access Management (IAM) permissions to access the required AWS services.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon Transcribe

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Build an Amazon Transcribe streaming app

The following code example shows how to build an app that records, transcribes, and translates live audio in real-time, and emails the results.

## JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Transcribe to build an app that records, transcribes, and translates live audio in real-time, and emails the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

## Convert text to speech and back to text using an AWS SDK

The following code example shows how to:

- Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file.
- Upload the audio file to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Transcribe to convert the audio file to text.
- Display the text.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file, upload the audio file to an Amazon Simple Storage Service bucket, use Amazon Transcribe to convert that audio file to text, and display the text.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Polly
- Amazon S3
- Amazon Transcribe

For a complete list of AWS SDK developer guides and code examples, see [Using this service with an AWS SDK \(p. 42\)](#). This topic also includes information about getting started and details about previous SDK versions.

# Security in Amazon Transcribe

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security of the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to Amazon Transcribe, see [AWS services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This section helps you understand how to apply the shared responsibility model when using Amazon Transcribe. The following topics show you how to configure Amazon Transcribe to meet your security and compliance objectives. You also learn how to use other AWS services to monitor and secure your Amazon Transcribe resources.

## Topics

- [Data protection in Amazon Transcribe \(p. 213\)](#)
- [Identity and access management for Amazon Transcribe \(p. 219\)](#)
- [Logging and monitoring in Amazon Transcribe \(p. 233\)](#)
- [Compliance validation for Amazon Transcribe \(p. 239\)](#)
- [Resilience in Amazon Transcribe \(p. 239\)](#)
- [Infrastructure security in Amazon Transcribe \(p. 240\)](#)

## Data protection in Amazon Transcribe

Amazon Transcribe conforms to the AWS [shared responsibility model](#), which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all of the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Amazon Transcribe or other AWS services using the AWS Management Console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon Transcribe or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the [AWS Security Blog](#).

#### Topics

- [Encryption at rest \(p. 214\)](#)
- [Encryption in transit \(p. 214\)](#)
- [Key management \(p. 214\)](#)
- [Opting out of using your data for service improvement \(p. 216\)](#)
- [Amazon Transcribe and interface VPC endpoints \(AWS PrivateLink\) \(p. 217\)](#)

## Encryption at rest

Amazon Transcribe uses the default Amazon S3 key (SSE-S3) for server-side encryption of transcripts placed in your Amazon S3 bucket.

When you use the [StartTranscriptionJob](#) API, you can specify your own KMS key to encrypt the output from a transcription job.

Amazon Transcribe uses an Amazon EBS volume encrypted with the default key.

## Encryption in transit

Amazon Transcribe uses TLS 1.2 with AWS certificates to encrypt data in transit. This includes streaming transcription.

## Key management

Amazon Transcribe works with KMS key to provide enhanced encryption for your data. Amazon S3 already enables you to encrypt your input audio when creating a transcription job. Integration with AWS KMS enables you to encrypt the output of the [StartTranscriptionJob](#) API.

If you don't specify a KMS key, the output of the transcription job is encrypted with the default Amazon S3 key (SSE-S3).

For more information on AWS KMS, see the [AWS Key Management Service Developer Guide](#).

## AWS KMS encryption with the AWS Management Console

To encrypt the output of your transcription job, you can choose between using a KMS key for the account that is making the request, or you can use a KMS key from another account.

If you don't specify a KMS key, the output of the transcription job is encrypted with the default Amazon S3 key (SSE-S3).

## To enable output result encryption

- Under **Output data** choose **Encryption**.



- Choose whether the KMS key is from the account you're currently using or from a different account. If you want to use a key from the current account, choose the key from **KMS key ID**. If you're using a key from a different account, you need to enter the key ARN. To use a key from a different account, the caller must have `kms:Encrypt` permissions for the KMS key.

## AWS KMS encryption with the API

To use output encryption with the API, you set the `OutputEncryptionKMSKeyId` parameter of the [StartTranscriptionJob](#) API. You can use a KMS key from the current account, or you can use a key from another account. The account that you are using to create the job must have `kms:Encrypt` permissions for the KMS key.

You can use either of the following to identify a KMS key in the current account:

- KMS key ID: "1234abcd-12ab-34cd-56ef-1234567890ab"
- KMS key Alias: "alias/ExampleAlias"

You can use either of the following to identify a KMS key in the current account or another account:

- Amazon Resource Name (ARN) of a KMS key: "arn:aws:kms:region:account-ID:key/1234abcd-12ab-34cd-56ef-1234567890ab"
- ARN of a KMS key alias: "arn:aws:kms:region:account-ID:alias/ExampleAlias"

## AWS KMS encryption context

AWS KMS encryption context is a map of plain text, non-secret key:value pairs. This map represents additional authenticated data, known as encryption context pairs, which provide an added layer of security for your data. Amazon Transcribe requires a symmetric key to encrypt transcription output into a customer-specified Amazon S3 bucket. To learn more, see [Using symmetric and asymmetric keys](#).

When creating your encryption context pairs, **do not** use sensitive information. Encryption context is not secret—it is visible in plain text within your CloudTrail logs (so you can use it to identify and categorize your cryptographic operations).

Your encryption context pair can include special characters, such as underscores (\_), dashes (-), slashes (/), \ and colons (:).

### Tip

It can be useful to relate the values in your encryption context pair to the data being encrypted. Although not required, we recommend you use non-sensitive metadata related to your encrypted content, such as file names, header values, or unencrypted database fields.

To use output encryption with the API, set the `KMSEncryptionContext` parameter in the [StartTranscriptionJob](#) operation. In order to provide encryption context for the output encryption operation, the `OutputEncryptionKMSKeyId` parameter must reference a symmetric KMS key ID.

You can use [AWS KMS condition keys](#) with IAM policies to control access to a symmetric KMS key based on the encryption context that was used in the request for a [cryptographic operation](#). For example, the following policy grants the IAM role “ExampleRole” permission to use the AWS KMS `Decrypt` and `Encrypt` operations for this particular KMS key. This policy works **only** for requests with at least one encryption context pair, in this case “color:indig0Blu3”.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
            },
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:Encrypt",
                "kms:GenerateDataKey*",
                "kms:ReEncrypt*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:color": "indig0Blu3"
                }
            }
        }
    ]
}
```

Using encryption context is optional, but recommended. For more information, see [Encryption context](#).

## Opting out of using your data for service improvement

By default, Amazon Transcribe stores and uses voice inputs that it has processed to develop the service and continuously improve your experience. You can opt out of having your content used to develop and improve Amazon Transcribe by using an AWS Organizations opt-out policy. For information about how to opt out, see [AI services opt-out policies](#).

Opting out has the following effect:

- Amazon Transcribe deletes all of the transcripts stored in service-managed buckets that were generated before you opted out.
- When you use the [StartTranscriptionJob](#) API, you must specify where you want to store output with the `OutputBucketName` parameter. Otherwise, you get a `BadRequestException` error.
- If the transcripts were stored in a service-managed bucket, the [GetTranscriptionJob](#) API returns `null` as the value of the `TranscriptFileUri` or `RedactedTranscriptFileUri` parameters.

If you store transcripts in service-managed buckets, we highly recommend backing them up. To back up your transcripts, store them in an Amazon S3 bucket that you manage before you opt out. To see which of your transcription jobs uses Amazon Transcribe to store its outputs, see the `OutputLocationType` response parameter of the [ListTranscriptionJobs](#) API.

### To move transcripts to your own Amazon S3 buckets

1. In the `TranscriptionJobName` parameter of the [GetTranscriptionJob](#) API, specify the name of the transcription job whose output you want to back up.
2. Use the link provided in the `TranscriptFileUri` or `RedactedTranscriptFileUri` response parameters to download your transcript.
3. Sign in to the [AWS Management Console](#).
4. In the **Bucket name** list, choose the name of the bucket that you want to upload your files to.
5. Choose **Upload**.
6. In the **Upload** dialog box, choose **Add files**.
7. Choose one or more files to upload, and then choose **Open**.
8. Choose **Upload**.

## Amazon Transcribe and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and Amazon Transcribe by creating an *interface VPC endpoint*. Interface endpoints are powered by [AWS PrivateLink](#), a technology that enables you to privately access Amazon Transcribe APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with Amazon Transcribe APIs. Traffic between your VPC and Amazon Transcribe does not leave the Amazon network.

Each interface endpoint is represented by one or more [Elastic Network Interfaces](#) in your subnets.

For more information, see [Interface VPC endpoints \(AWS PrivateLink\)](#) in the *Amazon VPC User Guide*.

### Considerations for Amazon Transcribe VPC endpoints

Before you set up an interface VPC endpoint for Amazon Transcribe, ensure that you review [Interface endpoint properties and limitations](#) in the *Amazon VPC User Guide*.

Amazon Transcribe supports making calls to all of its API actions from your VPC.

### Creating an interface VPC endpoint for Amazon Transcribe

You can create a VPC endpoint for the Amazon Transcribe service using either the Amazon VPC AWS Management Console or AWS CLI. For more information, see [Creating an interface endpoint](#) in the *Amazon VPC User Guide*.

For batch transcription in Amazon Transcribe, create a VPC endpoint using the following service name:

- com.amazonaws.**us-west-2**.transcribe

For streaming transcription in Amazon Transcribe, create a VPC endpoint using the following service name:

- com.amazonaws.**us-west-2**.transcribestreaming

If you enable private DNS for the endpoint, you can make API requests to Amazon Transcribe using its default DNS name for the AWS Region, for example, `transcribestreaming.us-east-2.amazonaws.com`.

For more information, see [Accessing a service through an interface endpoint](#) in the *Amazon VPC User Guide*.

## Creating a VPC endpoint policy for Amazon Transcribe

You can attach an endpoint policy to your VPC endpoint that controls access to either the streaming service or the batch transcription service of Amazon Transcribe. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see [Controlling access to services with VPC endpoints](#) in the *Amazon VPC User Guide*.

### Example: VPC endpoint policy for Amazon Transcribe streaming transcription actions

The following is an example of an endpoint policy for streaming transcription in Amazon Transcribe. When attached to an endpoint, this policy grants access to the listed Amazon Transcribe actions for all principals on all resources.

```
{  
    "Statement": [  
        {  
            "Principal": "*",  
            "Effect": "Allow",  
            "Action": [  
                "transcribe:StartStreamTranscription",  
                "transcribe:StartStreamTranscriptionWebSocket"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

### Example: VPC endpoint policy for Amazon Transcribe batch transcription actions

The following is an example of an endpoint policy for batch transcription in Amazon Transcribe. When attached to an endpoint, this policy grants access to the listed Amazon Transcribe actions for all principals on all resources.

```
{  
    "Statement": [  
        {  
            "Principal": "*",  
            "Effect": "Allow",  
            "Action": [  
                "transcribe:StartTranscriptionJob",  
                "transcribe>ListTranscriptionJobs"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

# Identity and access management for Amazon Transcribe

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Transcribe resources. IAM is an AWS service that you can use with no additional charge.

## Topics

- [Audience \(p. 219\)](#)
- [Authenticating with identities \(p. 219\)](#)
- [Managing access using policies \(p. 221\)](#)
- [How Amazon Transcribe works with IAM \(p. 223\)](#)
- [Amazon Transcribe identity-based policy examples \(p. 226\)](#)
- [Cross-service confused deputy prevention \(p. 230\)](#)
- [Troubleshooting Amazon Transcribe identity and access \(p. 231\)](#)

## Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon Transcribe.

**Service user** – If you use the Amazon Transcribe service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon Transcribe features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon Transcribe, see [Troubleshooting Amazon Transcribe identity and access \(p. 231\)](#).

**Service administrator** – If you're in charge of Amazon Transcribe resources at your company, you probably have full access to Amazon Transcribe. It's your job to determine which Amazon Transcribe features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon Transcribe, see [How Amazon Transcribe works with IAM \(p. 223\)](#).

**IAM administrator** – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon Transcribe. To view example Amazon Transcribe identity-based policies that you can use in IAM, see [Amazon Transcribe identity-based policy examples \(p. 226\)](#).

## Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see [Signing in to the AWS Management Console as an IAM user or root user](#) in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the [AWS Management Console](#), use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM

users access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 signing process](#) in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see [Using multi-factor authentication \(MFA\) in AWS](#) in the *IAM User Guide*.

## AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the *AWS account root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

## IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see [Managing access keys for IAM users](#) in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to create an IAM user \(instead of a role\)](#) in the *IAM User Guide*.

## IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by [switching roles](#). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Using IAM roles](#) in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an [identity provider](#). For more information about federated users, see [Federated users and roles](#) in the *IAM User Guide*.

- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see [Actions, Resources, and Condition Keys for Amazon Transcribe](#) in the *Service Authorization Reference*.
  - **Service role** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role \(instead of a user\)](#) in the *IAM User Guide*.

## Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

## Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choosing between managed policies and inline policies](#) in the *IAM User Guide*.

## Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

## Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

## Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [How SCPs work](#) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies.

Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session policies](#) in the *IAM User Guide*.

## Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

For more information about identity and access management for Amazon Transcribe, continue to the following pages:

- [How Amazon Transcribe works with IAM \(p. 223\)](#)
- [Troubleshooting Amazon Transcribe identity and access \(p. 231\)](#)

## How Amazon Transcribe works with IAM

Before you use IAM to manage access to Amazon Transcribe, you should understand which IAM features are available to use with Amazon Transcribe. To get a high-level view of how Amazon Transcribe and other AWS services work with IAM, see [AWS services That Work with IAM](#) in the *IAM User Guide*.

### Topics

- [Amazon Transcribe identity-based policies \(p. 223\)](#)
- [Amazon Transcribe IAM roles \(p. 226\)](#)

## Amazon Transcribe identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources and the conditions under which actions are allowed or denied. Amazon Transcribe supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see [IAM JSON Policy Elements Reference](#) in the *IAM User Guide*.

### Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The **Action** element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon Transcribe use the following prefix before the action: `transcribe:`. For example, to grant someone permission to run an Amazon EC2 instance with the Amazon Transcribe [StartTranscriptionJob](#) API, you include the `transcribe:StartTranscriptionJob` action in their policy. Policy statements must include either an **Action** or **NotAction** element. Amazon Transcribe defines actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows.

```
"Action": [
```

```
"transcribe:action1",  
"transcribe:action2"
```

You can specify multiple actions using wildcards (\*). For example, to specify all actions that begin with the word List, include the following action.

```
"Action": "transcribe>List*"
```

To see a list of Amazon Transcribe actions, see [Actions Defined by Amazon Transcribe](#) in the *IAM User Guide*.

**Note**

You can also control access to your resources using tags. For information on tag-based access control, see [Controlling access to AWS resources using tags](#).

## Resources

Amazon Transcribe doesn't support specifying resource ARNs in a policy.

### Condition keys

Use the Condition element (or a Condition block) to specify conditions in which a statement is in effect. The Condition element is optional. You can build conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](#) in the *IAM user guide*.

Amazon Transcribe defines its own set of condition keys and also supports using some global condition keys. For a list of all AWS global condition keys, see [AWS global condition context keys](#) in the *AWS Identity and Access Management User Guide*.

The following table lists the Amazon Transcribe condition keys that apply to Amazon Transcribe resources. You can include these keys in the Condition element in an IAM permissions policy.

Amazon Transcribe condition key	Description	Value type	Action
transcribe:OutputBucketName	Filters access by the output bucket used to start a transcription job.	String	<a href="#">StartTranscriptionJob</a>
transcribe:OutputEncryptionKey	Filters access by the KMS key used to start a transcription job.	String	<a href="#">StartTranscriptionJob</a>
transcribe:Outputkey	Filters access by the output key used to start a transcription job.	String	<a href="#">StartTranscriptionJob</a>

For examples of how you can use condition keys to control access to the resources of Amazon Transcribe, see the following.

If you want your users to always use a specific output bucket when they use the [StartTranscriptionJob](#) API, you can use the following policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": [  
                "transcribe:StartTranscriptionJob",  
                ],  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "transcribe:OutputBucketName": "DOC-EXAMPLE-BUCKET"  
                }  
            }  
        }  
    ]  
}
```

If you want users to always use a KMS key when they use the [StartTranscriptionJob](#) API, you can use the following policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": [  
                "transcribe:StartTranscriptionJob",  
                ],  
            "Resource": "*",  
            "Condition": {  
                "Null": {  
                    "transcribe:OutputEncryptionKMSKeyId": "false"  
                }  
            }  
        }  
    ]  
}
```

For information on KMS keys, see [Key management \(p. 214\)](#).

If you want your users to always use a specific output key when they use the [StartTranscriptionJob](#) API, you can use the following policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": [  
                "transcribe:StartTranscriptionJob",  
                ],  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "transcribe:OutputEncryptionKMSKeyId": "arn:aws:kms:  
                }  
            }  
        }  
    ]  
}
```

```
        "Effect": "Allow",
        "Action": [
            "transcribe:StartTranscriptionJob",
            ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "transcribe:Outputkey": "DOC-EXAMPLE-BUCKET/prefix"
            }
        }
    ]
}
```

For more information, see the [OutputKey](#) parameter description of the [StartTranscriptionJob](#) API.

## Examples

For examples of Amazon Transcribe identity-based policies, see [Amazon Transcribe identity-based policy examples \(p. 226\)](#).

## Amazon Transcribe resource-based policies

Amazon Transcribe doesn't support resource-based policies.

## Amazon Transcribe IAM roles

An [IAM role](#) is an entity within your AWS account that has specific permissions.

### Using temporary credentials with Amazon Transcribe

You can use temporary credentials to sign in with federation, to assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS Security Token Service (AWS STS) APIs, such as [AssumeRole](#) or [GetFederationToken](#).

Amazon Transcribe supports using temporary credentials.

### Service-linked roles

[Service-linked roles](#) allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but can't edit, the permissions for service-linked roles.

Amazon Transcribe doesn't support service-linked roles.

### Service roles

You can allow a service to assume a [service role](#) on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might prevent the service from functioning as expected.

Refer to [Cross-service confused deputy prevention \(p. 230\)](#) to learn about preventing unwanted services from assuming roles and performing actions with Amazon Transcribe.

## Amazon Transcribe identity-based policy examples

By default, IAM users and roles don't have permission to create or modify Amazon Transcribe resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM

administrator must create IAM policies that grant users and roles permission to perform specific API operations on the resources that they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see [Creating policies on the JSON tab](#) in the *IAM User Guide*.

#### Topics

- [Policy best practices \(p. 227\)](#)
- [Using the AWS Management Console \(p. 227\)](#)
- [AWS managed \(predefined\) policies for Amazon Transcribe \(p. 228\)](#)
- [Permissions required for IAM user roles \(p. 228\)](#)
- [Permissions required for Amazon S3 encryption keys \(p. 229\)](#)
- [Allow users to view their own permissions \(p. 229\)](#)

## Policy best practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Amazon Transcribe resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started using AWS managed policies** – To start using Amazon Transcribe quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see [Get started using permissions with AWS managed policies](#) in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see [Grant least privilege](#) in the *IAM User Guide*.
- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see [Using multi-factor authentication \(MFA\) in AWS](#) in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.

## Using the AWS Management Console

To access the [AWS Management Console](#), you must have a minimum set of permissions for the AWS Management Console. These permissions must allow you to list and view details about the Amazon Transcribe resources in your AWS account. If you create an identity-based policy that applies permissions that are more restrictive than the minimum required permissions, the AWS Management Console won't function as intended for entities (IAM users or roles) with that policy.

To ensure that those entities can use the [AWS Management Console](#), attach the following AWS managed policy to them.

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

```
        "Action": [
            "transcribe:*"
        ],
        "Resource": "*",
        "Effect": "Allow"
    }
}
```

You don't need to allow minimum AWS Management Console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

For more information, see [Adding Permissions to a User](#) in the *IAM User Guide*:

## AWS managed (predefined) policies for Amazon Transcribe

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These policies are called AWS managed policies. Managed policies make it easier for you to assign appropriate permissions to users, groups, and roles than if you had to write the policies yourself. For more information, see [AWS Managed Policies](#) in the *AWS Identity and Access Management User Guide*.

The following AWS managed policies, which you can attach to users, roles, and groups in your account, are specific to Amazon Transcribe:

- **AmazonTranscribeReadOnly** – Grants read-only access to Amazon Transcribe resources so that you can get and list transcription jobs and custom vocabularies.
- **AmazonTranscribeFullAccess** – Grants full access to create, read, update, delete, and run all Amazon Transcribe resources. It also allows access to Amazon S3 buckets with `transcribe` in the bucket name.

### Note

You can review the managed permission policies by signing in to the IAM AWS Management Console and searching by policy name. A search for "transcribe" returns both policies listed above (`AmazonTranscribeReadOnly` and `AmazonTranscribeFullAccess`).

You can also create your own custom IAM policies to allow permissions for Amazon Transcribe API actions. You can attach these custom policies to the IAM users, roles, or groups that require those permissions.

## Permissions required for IAM user roles

When you create an IAM user to call Amazon Transcribe, the identity must have permission to access the Amazon S3 bucket and the KMS key used to encrypt the contents of the bucket, if you provided one.

The user must have the following IAM policy for decrypt permissions on the AWS KMS Amazon Resource Name (ARN).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-west-2:111122223333:key/KMS-Example-KeyId",
            "Effect": "Allow"
        }
    ]
}
```

```
        ]
    }
}
```

The user's IAM policy must have Amazon S3 permissions to access the Amazon S3 bucket where audio files are stored and transcriptions are saved.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/"
        }
    ]
}
```

## Permissions required for Amazon S3 encryption keys

If you are using a KMS key to encrypt an Amazon S3 bucket, include the following in the KMS key policy. This gives Amazon Transcribe access to the contents of the bucket.

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

For more information about allowing access to KMS keys, see [Allowing external AWS accounts to access an KMS key in the AWS KMS developer guide](#).

## Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "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": "*"
}
]
}

```

## Cross-service confused deputy prevention

A confused deputy is an entity (a service or an account) that is coerced by a different entity to perform an action. This type of impersonation can happen cross-account and cross-service.

To prevent confused deputies, AWS provides tools that help you protect your data for all services using service principals that have been given access to resources in your account. This section focuses on cross-service confused deputy prevention specific to Amazon Transcribe; however, you can learn more about this topic in the [confused deputy problem](#) section of the IAM User Guide.

To limit the permissions IAM gives to Amazon Transcribe to access your resources, we recommend using the global condition context keys `aws:SourceArn` and `aws:SourceAccount` in your resource policies. Note that if you use both of these global condition context keys, and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement.

If you want only one resource to be associated with the cross-service access, use `aws:SourceArn`. If you want to associate any resource in that account with cross-service access, use `aws:SourceAccount`.

### Note

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the **full ARN** of the resource. If you don't know the full ARN, or if you're specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (\*) for the unknown portions of the ARN. For example, `arn:aws:transcribe::123456789012:*`.

Here's an example of an assume role policy that shows how you can use `aws:SourceArn` and `aws:SourceAccount` with Amazon Transcribe to prevent a confused deputy issue.

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

```

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

## Troubleshooting Amazon Transcribe identity and access

Use the following information to diagnose and fix common issues that you might encounter when working with Amazon Transcribe and AWS Identity and Access Management (IAM).

### Topics

- [I am not authorized to perform an action in Amazon Transcribe \(p. 231\)](#)
- [I am not authorized to perform iam:PassRole \(p. 231\)](#)
- [I want to view my access keys \(p. 232\)](#)
- [I'm an administrator and want to allow others to access Amazon Transcribe \(p. 232\)](#)
- [I want to allow people outside of my AWS account to access my Amazon Transcribe resources \(p. 232\)](#)

## I am not authorized to perform an action in Amazon Transcribe

If you are using the AWS Management Console, and you get a message that you're not authorized to perform an action, contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

For example, the following error occurs when an IAM user named `mateojackson` tries to use the AWS Management Console to view details about a transcription job but doesn't have `transcribe:GetTranscriptionJob` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:  
transcribe:GetTranscriptionJob
```

Mateo must ask his administrator to update his policies to allow him to access the `GetTranscriptionJob` API using the `transcribe:GetTranscriptionJob` action.

## I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to Amazon Transcribe.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon Transcribe. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

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

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

## I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

**Important**

Do not provide your access keys to a third party, even to help [find your canonical user ID](#). By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see [Managing access keys](#) in the *IAM User Guide*.

## I'm an administrator and want to allow others to access Amazon Transcribe

To allow others to access Amazon Transcribe, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Amazon Transcribe.

To get started right away, see [Creating your first IAM delegated user and group](#) in the *IAM User Guide*.

## I want to allow people outside of my AWS account to access my Amazon Transcribe resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon Transcribe supports these features, see [How Amazon Transcribe works with IAM](#) (p. 223).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.

- To learn the difference between using roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

## Logging and monitoring in Amazon Transcribe

Monitoring is an important part of maintaining the reliability, availability, and performance of your Amazon Transcribe applications. To monitor Amazon Transcribe API calls, you can use AWS CloudTrail. To monitor the status of your jobs, use Amazon EventBridge.

### Topics

- [Monitoring Amazon Transcribe API calls with AWS CloudTrail \(p. 233\)](#)
- [Using Amazon EventBridge with Amazon Transcribe \(p. 235\)](#)
- [Using Amazon CloudWatch metrics and dimensions with Amazon Transcribe \(p. 239\)](#)

## Monitoring Amazon Transcribe API calls with AWS CloudTrail

Amazon Transcribe is integrated with AWS CloudTrail, a service that provides a record of actions taken in Amazon Transcribe by an AWS Identity and Access Management (IAM) user or role, or by an AWS service. CloudTrail captures all API calls for Amazon Transcribe, including calls from the AWS Management Console and from code calls to the Amazon Transcribe APIs, as events. By creating a trail, you can enable continuous delivery of CloudTrail events, including events for Amazon Transcribe, to an Amazon S3 bucket. If you don't create a trail, you can still view the most recent events in the CloudTrail AWS Management Console in **Event history**. Using the information collected by CloudTrail, you can see each request that is made to Amazon Transcribe, the IP address from which the request is made, who made the request, when it is made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

### Amazon Transcribe information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon Transcribe, that activity is recorded in a CloudTrail event along with other AWS service events in the CloudTrail **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing Events with CloudTrail Event History](#).

For an ongoing record of events in your AWS account, including events for Amazon Transcribe, create a trail. A *trail* is a configuration that enables CloudTrail to deliver events as log files to a specified Amazon S3 bucket. By default, when you create a trail in the AWS Management Console, the trail applies to all AWS Regions. The trail logs events from all AWS Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#)

CloudTrail logs all Amazon Transcribe actions, which are documented in the [Reference](#). For example, calls to the [CreateVocabulary](#), [GetTranscriptionJob](#), and [StartTranscriptionJob](#) APIs generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. This information helps you determine the following:

- Whether the request is made with root or IAM user credentials
- Whether the request is made with temporary security credentials for an IAM role or federated user
- Whether the request is made by another AWS service

For more information, see the [CloudTrail userIdentity Element](#).

You can also aggregate Amazon Transcribe log files from multiple AWS Regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#).

## Example: Amazon Transcribe log file entries

A *trail* is a configuration that enables delivery of events as log files to a specified Amazon S3 bucket. CloudTrail log files contain one or more log entries. An *event* represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

Calls to the [StartTranscriptionJob](#) and [GetTranscriptionJob](#) APIs create the following entry.

```
{  
    "Records": [  
        {  
            "eventVersion": "1.05",  
            "userIdentity": {  
                "type": "AssumedRole | FederatedUser | IAMUser | Root | SAMLUser | WebIdentityUser",  
                "principalId": "principal ID",  
                "arn": "ARN",  
                "accountId": "account ID",  
                "accessKeyId": "access key",  
                "userName": "user name"  
            },  
            "eventTime": "timestamp",  
            "eventSource": "transcribe.amazonaws.com",  
            "eventName": "StartTranscriptionJob",  
            "awsRegion": "us-west-2",  
            "sourceIPAddress": "192.0.2.0/24",  
            "userAgent": "user agent",  
            "requestParameters": {  
                "mediaFormat": "flac",  
                "languageCode": "en-US | es-US",  
                "transcriptionJobName": "my-first-transcription-job",  
                "media": {  
                    "mediaFileUri": ""  
                }  
            },  
            "responseElements": {  
                "transcriptionJob": {  
                    "transcriptionJobStatus": "IN_PROGRESS",  
                    "mediaFormat": "flac",  
                    "creationTime": "timestamp",  
                    "transcriptionJobName": "my-first-transcription-job",  
                    "languageCode": "en-US | es-US",  
                    "media": {  
                        "mediaFileUri": ""  
                    }  
                }  
            }  
        }  
    ]  
}
```

```

        },
        "requestID": "request ID",
        "eventID": "event ID",
        "eventType": "AwsApiCall",
        "recipientAccountId": "account id"
    },
    {
        "eventVersion": "1.05",
        "userIdentity": {
            "type": "AssumedRole | FederatedUser | IAMUser | Root | SAMLUser | WebIdentityUser",
            "principalId": "principal ID",
            "arn": "ARN",
            "accountId": "account ID",
            "accessKeyId": "access key",
            "userName": "user name"
        },
        "eventTime": "timestamp",
        "eventSource": "transcribe.amazonaws.com",
        "eventName": "GetTranscriptionJob",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "192.0.2.0/24",
        "userAgent": "user agent",
        "requestParameters": {
            "transcriptionJobName": "my-first-transcription-job"
        },
        "responseElements": {
            "transcriptionJob": {
                "settings": {

                },
                "transcriptionJobStatus": "COMPLETED | FAILED | IN_PROGRESS",
                "mediaFormat": "flac",
                "creationTime": "timestamp",
                "transcriptionJobName": "my-first-transcription-job",
                "languageCode": "en-US | es-US",
                "media": {
                    "mediaFileUri": ""
                },
                "transcript": {
                    "transcriptFileUri": ""
                }
            }
        },
        "requestID": "request ID",
        "eventID": "event ID",
        "eventType": "AwsApiCall",
        "recipientAccountId": "account id"
    }
]
}

```

## Using Amazon EventBridge with Amazon Transcribe

With Amazon EventBridge, you can respond to state changes in your Amazon Transcribe jobs by triggering events in other AWS services. When a transcription job changes state, EventBridge automatically sends an event to an event stream. You create rules that define the events that you want to monitor in the event stream and the action that EventBridge should take when those events occur. For example, routing the event to another service (or target), which can then take an action. You could, for example, configure a rule to route an event to an AWS Lambda function when a transcription job has completed successfully.

Before using EventBridge, you should understand the following concepts:

- **Event** – An event indicates a change in the state of one of your transcription jobs. For example, when the `TranscriptionJobStatus` of a job changes from `IN_PROGRESS` to `COMPLETED`.
- **Target** – A target is another AWS service that processes an event. For example, AWS Lambda or Amazon Simple Notification Service (Amazon SNS). A target receives events in JSON format.
- **Rule** – A rule matches incoming events that you want EventBridge to watch for and routes them to a target or targets for processing. If a rule routes an event to multiple targets, all of the targets process the event in parallel. A rule can customize the JSON sent to the target.

Amazon EventBridge events are emitted on a best-effort basis. For more information about creating and managing events in EventBridge, see [Amazon EventBridge events](#) in the *Amazon EventBridge User Guide*.

## Defining EventBridge rules

To define EventBridge rules, use the [AWS Management Console](#). When you define a rule, use Amazon Transcribe as the service name. For an example of how to create an EventBridge rule, see [Amazon EventBridge rules](#).

The following is an example of an EventBridge rule for Amazon Transcribe. It's triggered when a transcription job's status changes to `COMPLETED` or `FAILED`.

```
{  
    "source": [  
        "aws.transcribe"  
    ],  
    "detail-type": [  
        "Transcribe Job State Change"  
    ],  
    "detail": {  
        "TranscriptionJobStatus": [  
            "COMPLETED",  
            "FAILED"  
        ]  
    }  
}
```

The rule contains the following fields:

- `source` – The source of the event. For Amazon Transcribe, this is always `aws.transcribe`.
- `detail-type` – An identifier for the details of the event. For Amazon Transcribe, this is always `Transcribe Job State Change`.
- `detail` – The new job status of the transcription job. In this example, the rule triggers an event when the job status changes to `COMPLETED` or `FAILED`. For a list of status values, see the [TranscriptionJob API](#).

## Amazon Transcribe events

Amazon EventBridge logs three kinds of Amazon Transcribe events: transcription job events, automatic language identification events, and Call Analytics events.

Definitions for all fields are provided below the event examples.

### Transcription job events

When a job's state changes from `IN_PROGRESS` to either `COMPLETED` or `FAILED`, Amazon Transcribe generates an event. To identify the job that changed state and triggered the event in your target, use the

event's `TranscriptionJobName` field. An Amazon Transcribe event contains the following information. A `FailureReason` field is added under `detail` if your transcription job status is `FAILED`.

```
{  
    "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"  
    }  
}
```

## Language identification events

When you enable automatic language identification, Amazon Transcribe generates an event when the language identification state is either `COMPLETED` or `FAILED`. To identify the job that changed state and triggered the event in your target, use the event's `JobName` field. An Amazon Transcribe event contains the following information. A `FailureReason` field is added under `detail` if your language identification status is `FAILED`.

```
{  
    "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": "job-name",  
        "LanguageIdentificationStatus": "status"  
    }  
}
```

## Call Analytics events

When a Call Analytics job state changes from `IN_PROGRESS` to either `COMPLETED` or `FAILED`, Amazon Transcribe generates an event. To identify the Call Analytics job that changed state and triggered the event in your target, use the event's `JobName` field. An Amazon Transcribe event contains the following information. A `FailureReason` field is added under `detail` if your Call Analytics job status is `FAILED`.

```
{  
    "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": "job-name",  
    }  
}
```

```
        "JobStatus": "status"
    }
```

## Vocabulary events

When a vocabulary's state changes from PENDING to either READY or FAILED, Amazon Transcribe generates an event. To identify the vocabulary that changed state and triggered the event in your target, use the event's VocabularyName field. An Amazon Transcribe event contains the following information. A FailureReason field is added under detail if your vocabulary state is FAILED.

```
{
    "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": "state"
    }
}
```

Descriptions for the preceding fields:

- **version** – The version of the event data. This value is always 0.
- **id** – A unique identifier generated by EventBridge for the event.
- **detail-type** – An identifier for the details of the event. For Amazon Transcribe, this is one of Transcribe Job State Change, Language Identification State Change, Call Analytics Job State Change, or Vocabulary State Change.
- **source** – The source of the event. For Amazon Transcribe this is always aws.transcribe.
- **account** – The AWS account ID of the account that generated the API call.
- **timestamp** – The date and time the event is delivered.
- **region** – The AWS Region where the API call is made.
- **resources** – The resources used by the API call. For Amazon Transcribe, this field is always empty.
- **detail** – Details about the event. The fields within the detail tag may vary depending on the type of event, but may contain the following fields:
  - **JobType** – For transcription jobs, this value must be TranscriptionJob.
  - **JobName** – The unique name for your transcription job.
  - **JobStatus** – The status of your Call Analytics transcription job. It can be either COMPLETED or FAILED.
  - **TranscriptionJobName** – The unique name you chose for your transcription job.
  - **TranscriptionJobStatus** – The status of the transcription job. For a list of status values, see the TranscriptionJobStatus field of the [TranscriptionJob](#) data type.
  - **LanguageIdentificationStatus** – The status of language identification in a transcription job. It can be either COMPLETED or FAILED.
  - **VocabularyName** – The unique name for your vocabulary.
  - **VocabularyState** – The processing state of your vocabulary. For a list of status values, see the VocabularyState field of the [VocabularyInfo](#) data type.
  - **FailureReason** – This field is present if the state or status changes to FAILED, and describes the reason for the FAILED state or status.

## Using Amazon CloudWatch metrics and dimensions with Amazon Transcribe

Amazon Transcribe supports CloudWatch metrics and dimensions, which are data that can help you monitor performance; supported metrics categories include traffic, errors, data transfer, and latency associated with your transcription jobs. Supported metrics are located through CloudWatch in the **AWS/Transcribe** namespace.

**Note**

CloudWatch monitoring metrics are free of charge and don't count against CloudWatch service quotas.

For more information on CloudWatch metrics, see [Using Amazon CloudWatch metrics](#).

## Compliance validation for Amazon Transcribe

Third-party auditors assess the security and compliance of Amazon Transcribe as part of multiple AWS compliance programs. These include PCI, FedRAMP, HIPAA, and others. You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using Amazon Transcribe is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and compliance quick start guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS Compliance resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Config](#) – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

For a list of AWS services in scope of specific compliance programs, see [AWS services in Scope by Compliance Program](#). For general information, see [AWS Compliance Programs](#).

## Resilience in Amazon Transcribe

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

## Infrastructure security in Amazon Transcribe

As a managed service, Amazon Transcribe is protected by the AWS global network security procedures that are described in the [Amazon Web Services: Overview of Security Processes](#) whitepaper.

To access Amazon Transcribe through the network, you use AWS published API calls. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems, such as Java 7 and later, support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

# Amazon Transcribe Medical

Amazon Transcribe Medical is an automatic speech recognition (ASR) service designed for medical professionals who wish to transcribe medical-related speech, such as physician-dictated notes, drug safety monitoring, telemedicine appointments, or physician-patient conversations. Amazon Transcribe Medical is available through either real-time streaming (via microphone) or transcription of an uploaded file (batch).

## Important

Amazon Transcribe Medical is not a substitute for professional medical advice, diagnosis, or treatment. Identify the right confidence threshold for your use case, and use high confidence thresholds in situations that require high accuracy. For certain use cases, results should be reviewed and verified by appropriately trained human reviewers. Amazon Transcribe Medical transcriptions should only be used in patient care scenarios after review for accuracy and sound medical judgment by trained medical professionals.

Amazon Transcribe Medical operates under a shared responsibility model, whereby AWS is responsible for protecting the infrastructure that runs Amazon Transcribe Medical and you are responsible for managing your data. For more information, see [Shared Responsibility Model](#).

Amazon Transcribe Medical is available in US English (en-US).

For analysis of your transcripts, you can use other AWS services, such as [Amazon Comprehend Medical](#).

## Supported specialties

Specialty	Sub-specialty	Audio input
Cardiology	none	streaming only
Neurology	none	streaming only
Oncology	none	streaming only
Primary Care	Family Medicine	batch, streaming
Primary Care	Internal Medicine	batch, streaming
Primary Care	Obstetrics and Gynecology (OB-GYN)	batch, streaming
Primary Care	Pediatrics	batch, streaming
Radiology	none	streaming only
Urology	none	streaming only

## How Amazon Transcribe Medical works

Amazon Transcribe Medical enables you to transcribe individual audio content with medical information into text.

Transcription methods can be separated into two main categories:

- Batch transcription jobs: Transcribing media that have been uploaded into an Amazon S3 bucket.
- Streaming transcriptions: Transcribing media streams in real time.

Batch transcription jobs can be created with the [AWS CLI \(p. 31\)](#), [AWS Management Console \(p. 23\)](#), and various [AWS SDKs \(p. 34\)](#).

Streaming transcriptions can be created with the [AWS Management Console \(p. 23\)](#), [HTTP/2 \(p. 50\)](#), [WebSockets \(p. 54\)](#), and various [AWS SDKs \(p. 34\)](#).

For more information on medical transcriptions with streaming audio, see [Streaming transcription overview \(p. 243\)](#) and [Establish a bi-directional connection using the WebSocket protocol \(p. 267\)](#). For more information on batch transcriptions, see [Batch transcription overview \(p. 243\)](#).

## Speech input

Amazon Transcribe Medical can transcribe speech as either an audio file or a real-time stream. Your input audio must use the encodings and formats described in the following sections.

### Topics

- [Containers and formats for batch transcription \(p. 242\)](#)
- [Audio containers and formats for streaming transcription \(p. 242\)](#)

## Containers and formats for batch transcription

When you transcribe audio using the [StartMedicalTranscriptionJob](#) API or the AWS Management Console, make sure that the file is:

- In FLAC, MP3, MP4, Ogg, WebM, AMR, or WAV file format
- Less than 4 hours in length and less than 2 GB in size
- Encoded at a sample rate of 16,000 Hz or higher

### Note

For AMR, Amazon Transcribe Medical supports both Adaptive Multi-Rate Wideband (AMR-WB) and Adaptive Multi-Rate Narrowband (AMR-NB) codecs.

For the Ogg and WebM file formats, Amazon Transcribe Medical supports the Opus codec.

For best results:

- Use a lossless format. You can choose either FLAC, or WAV with PCM 16-bit encoding.

## Audio containers and formats for streaming transcription

When you transcribe a real-time stream using the [StartMedicalStreamTranscription](#) API or a WebSocket request, make sure that your stream is encoded in:

- PCM signed 16-bit little-endian
- FLAC
- OPUS encoded audio in the Ogg container

Your stream must use a sample rate of 16,000 Hz or higher.

For best results:

- Use a lossless format, such as FLAC or PCM encoding.

For more information on using a WebSocket request to transcribe your streaming audio, see [Establish a bi-directional connection using the WebSocket protocol \(p. 267\)](#).

## Streaming transcription overview

Streaming transcription takes a stream of your audio data and transcribes it in real time. It uses a bidirectional WebSocket connection so that the results of the transcription are returned to your application while you send more audio to Amazon Transcribe Medical. You can also use it when you have an audio file that you want to process as it is transcribed.

Streaming transcription is available in US English (en-US). It can produce transcriptions of accented English, spoken by non-native speakers. Streaming audio transcription comes with the following features:

- Transcribe 16 kHz audio in real time.
- Transcribe up to 4 hours of audio streams.
- Word-level timestamp in transcripts.
- Word-level confidence in transcripts.
- [Number normalization \(p. 244\)](#).
- Punctuation and true casing in transcripts.
- Supports both dictation and conversation speech types.

For more information, see [Establish a bi-directional connection using the WebSocket protocol \(p. 267\)](#).

## Batch transcription overview

Amazon Transcribe Medical batch transcription is available in US English. It has the ability to transcribe accented English from non-native speakers. It supports the transcription of individual audio files. You start a transcription job with either the AWS Management Console or by direct API call.

You interact with Amazon Transcribe Medical using four main API resources. To start a medical transcription job, use the [StartMedicalTranscriptionJob](#) API. To retrieve information on a medical transcription job, use [GetMedicalTranscriptionJob](#). You list medical transcription jobs with [ListMedicalTranscriptionJobs](#). You delete a medical transcription job with [DeleteMedicalTranscriptionJob](#).

To transcribe an audio file, you use a transcription job. You store the file as an object in an Amazon S3 bucket. The input file must:

- Be in FLAC, MP3, MP4, or WAV file format.
- Use 16-bit Linear PCM encoding.
- Be less than 4 hours in length and less than 2 GB in size
- Use a sample rate of 16,000 Hz or higher

For best results:

- Use a lossless format, such as FLAC or WAV.

When creating a medical transcription job, you specify the language, the medical specialty, and the audio type of the source file. You input US English (en-US) as the language and PRIMARYCARE as the medical specialty. Entering primary care as the value enables you to generate transcriptions from source audio in the following medical specialties:

- Family Medicine
- Internal Medicine
- Obstetrics and Gynecology (OB-GYN)
- Pediatrics

You have the choice between dictation and conversation for your audio type. Choose dictation for audio files where the physician is giving a report about a patient visit or procedure. Choose conversation for audio files that involve a conversation between a physician and a patient or a conversation between physicians.

To store the output of your transcription job, select an Amazon S3 bucket that you've already created. For more information on Amazon S3 buckets see [Getting Started with Amazon Simple Storage Service](#).

The following are the minimum number of request parameters to enter in the sample JSON.

```
{
    "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 enables you to generate alternative transcriptions. For more information, see [Generating alternative transcriptions \(p. 315\)](#).

You can also identify different speakers or channels in your audio. For more information, see [Identifying speakers and labeling their speech \(p. 280\)](#) and [Transcribing multi-channel audio \(p. 287\)](#).

## Transcribing numbers

Amazon Transcribe Medical transcribes digits as numbers instead of words. For example, the spoken number "one thousand two hundred forty-two" is transcribed as 1242.

Numbers are transcribed according to the following rules.

Rule	Description
Convert cardinal numbers greater than 10 to numbers.	<ul style="list-style-type: none"> <li>• "Fifty five" &gt; 55</li> <li>• "a hundred" &gt; 100</li> <li>• "One thousand and thirty one" &gt; 1031</li> <li>• "One hundred twenty-three million four hundred fifty six thousand seven hundred eight nine" &gt; 123,456,789</li> </ul>
Convert cardinal numbers followed by "million" or "billion" to numbers followed by a word when "million" or "billion" is not followed by a number.	<ul style="list-style-type: none"> <li>• "one hundred million" &gt; 100 million</li> <li>• "one billion" &gt; 1 billion</li> <li>• "two point three million" &gt; 2.3 million</li> </ul>
Convert ordinal numbers greater than 10 to numbers.	<ul style="list-style-type: none"> <li>• "Forty third" &gt; 43rd</li> </ul>

Rule	Description
	<ul style="list-style-type: none"> <li>"twenty sixth avenue" &gt; 26th avenue</li> </ul>
Convert fractions to their numeric format.	<ul style="list-style-type: none"> <li>"a quarter" &gt; 1/4</li> <li>"three sixteenths" &gt; 3/16</li> <li>"a half" &gt; 1/2</li> <li>"a hundredth" &gt; 1/100</li> </ul>
Convert numbers less than 10 to digits if there are more than one in a row.	<ul style="list-style-type: none"> <li>"three four five" &gt; 345</li> <li>"My phone number is four two five five five five one two one two" &gt; 4255551212</li> </ul>
Decimals are indicated by "dot" or "point."	<ul style="list-style-type: none"> <li>"three hundred and three dot five" &gt; 303.5</li> <li>"three point twenty three" &gt; 3.23</li> <li>"zero point four" &gt; 0.4</li> <li>"point three" &gt; 0.3</li> </ul>
Convert the word "percent" after a number to a percent sign.	<ul style="list-style-type: none"> <li>"twenty three percent" &gt; 23%</li> <li>"twenty three point four five percent" &gt; 23.45%</li> </ul>
Convert the words "dollar," "US dollar," "Australian dollar," "AUD," or "USD" after a number to a dollar symbol before the number.	<ul style="list-style-type: none"> <li>"one dollar and fifteen cents" &gt; \$1.15</li> <li>"twenty three USD" &gt; \$23</li> <li>"twenty three Australian dollars" &gt; \$23</li> </ul>
Convert the words "pounds," or "milligrams" to "lbs" or "mg".	<ul style="list-style-type: none"> <li>"twenty three pounds" &gt; 23 lbs</li> <li>"forty-five milligrams" &gt; 45 mg</li> </ul>
Convert the words "rupees," "Indian rupees," or "INR" after a number to rupee sign (₹) before the number.	<ul style="list-style-type: none"> <li>"twenty three rupees" &gt; ₹23</li> <li>"fifty rupees thirty paise" &gt; ₹50.30</li> </ul>
Convert times to numbers.	<ul style="list-style-type: none"> <li>"seven a m eastern standard time" &gt; 7 a.m. eastern standard time</li> <li>"twelve thirty p m" &gt; 12:30 p.m.</li> </ul>
Combine years expressed as two digits into four.  Only valid for the 20th, 21st, and 22nd centuries.	<ul style="list-style-type: none"> <li>"nineteen sixty two" &gt; 1962</li> <li>"the year is twenty twelve" &gt; the year is 2012</li> <li>"twenty nineteen" &gt; 2019</li> <li>"twenty one thirty" &gt; 2130</li> </ul>
Convert dates to numbers.	<ul style="list-style-type: none"> <li>"May fifth twenty twelve" &gt; May 5th 2012</li> <li>"May five twenty twelve" &gt; May 5 2012</li> <li>"five May twenty twelve" &gt; 5 May 2012</li> </ul>
Separate spans of numbers by the word "to."	<ul style="list-style-type: none"> <li>"twenty three to thirty seven" &gt; 23 to 37</li> </ul>

## Transcribing medical terms and measurements

Amazon Transcribe Medical can transcribe medical terms and measurements. Amazon Transcribe Medical outputs abbreviations for spoken terms. For example, "blood pressure" is transcribed as BP. You can find a list of conventions that Amazon Transcribe Medical uses for medical terms and measurements in the table on this page. The *Spoken Term* column refers to the term spoken in the source audio. The *Output* column refers to the abbreviation you see in your transcription results.

You can see how the terms spoken in source audio correspond to the transcription output here.

<b>Term spoken in source audio</b>	<b>Abbreviation used in output</b>	<b>Example output</b>
Centigrade	C	The patient's temperature is 37.4 C.
Celsius	C	The patient's temperature is 37.4 C.
Fahrenheit	F	The patient's temperature is 101 F.
grams	g	A mass of 100 g was extracted from the patient.
meters	m	The patient is 1.8 m tall.
feet	ft	The patient is 6 ft tall.
kilos	kg	The patient weighs 80 kg.
kilograms	kg	The patient weighs 80 kg.
c c	cc	Patient received 100 cc of saline solution.
cubic centimeter	cc	Patient received 100 cc of saline solution.
milliliter	mL	Patient excreted 100 mL urine.
blood pressure	BP	Patient BP was elevated.
b p	BP	Patient BP was elevated.
X over Y	X/Y	Patient BP was 120/80.
beats per min	BPM	Patient had atrial fibrillation with heart rate of 160 BPM.
beats per minute	BPM	Patient had atrial fibrillation with heart rate of 160 BPM.
O 2	O2	Patient O2 saturation was 98%.
CO2	CO2	Patient required respiratory support for elevated CO2.
post operation	POSTOP	Patient came for POSTOP evaluation.
post op	POSTOP	Patient came for POSTOP evaluation.
cat scan	CT Scan	Patient indication of cerebral hemorrhage required use of CT Scan.
Pulse 80	P 80	Patient vitals were P 80, R 17,...

Term spoken in source audio	Abbreviation used in output	Example output
Respiration 17	R 17	Patient vitals were P 80, R 17,...
in and out	I/O	Patient was I/O sinus rhythm
L five	L5	Lumbar puncture was performed between L4 and L5

## Getting started with Amazon Transcribe Medical

To get started using Amazon Transcribe Medical, set up an AWS account and create an AWS Identity and Access Management (IAM) user.

### Topics

- [Set up an AWS account and create an administrator user \(p. 247\)](#)
- [Getting started with the AWS Management Console \(streaming\) \(p. 248\)](#)
- [Getting started with batch transcription \(p. 249\)](#)

## Set up an AWS account and create an administrator user

Before you use Amazon Transcribe Medical for the first time, complete the following tasks:

1. [Sign up for Amazon Web Services \(p. 247\)](#)
2. [Create an IAM user \(p. 248\)](#)

## Sign up for Amazon Web Services

When you sign up for Amazon Web Services, your AWS account is automatically signed up for all AWS services, including Amazon Transcribe Medical. You are charged only for the services that you use.

With Amazon Transcribe Medical, you pay only for the resources that you use. If you are a new AWS customer, you can get started with Amazon Transcribe Medical for free. For more information, see [AWS Free Usage Tier](#).

If you already have an AWS account, skip to the next section.

### To create an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Record your AWS account ID because you'll need it for the next task.

## Create an IAM user

Services in AWS, such as Amazon Transcribe Medical, require that you provide credentials when you access them. This allows the service to determine whether you have permissions to access the service's resources.

We strongly recommend that you access AWS using AWS Identity and Access Management (IAM), not the credentials for your AWS account. To use IAM to access AWS, create an IAM user, add the user to an IAM group with administrative permissions, and then grant administrative permissions to the IAM user. You can then access AWS using a special URL and the IAM user's credentials.

The Getting Started exercises in this guide assume that you have a user with administrator privileges, `adminuser`.

### To create an administrator user and sign in to the AWS Management Console

1. Create an administrator user called `adminuser` in your AWS account. For instructions, see [Creating Your First IAM User and Administrators Group](#) in the *IAM User Guide*.
2. Sign in to the AWS Management Console using a special URL. For more information, see [How Users Sign In to Your Account](#) in the *IAM User Guide*.

For more information about IAM, see the following:

- [AWS Identity and Access Management \(IAM\)](#)
- [Getting started](#)
- [IAM User Guide](#)

## Next step

To get started with medical transcription using the AWS Management Console, see [Getting Started \(AWS Management Console\) \(p. 248\)](#).

## Getting started with the AWS Management Console (streaming)

Get started with Amazon Transcribe Medical streaming transcription by using the AWS Management Console to transcribe up to 15 minutes of medical speech into text. You use the microphone on your computer for the audio source.

### To start a streaming medical transcription job

1. Open the [AWS Management Console](#).
2. From the left menu, look under **Amazon Transcribe Medical** and choose **Real-time transcription**.
3. Choose **Start streaming** and then speak into your microphone. Amazon Transcribe Medical will transcribe the speech and display the results on the AWS Management Console.

## Next step

To learn more about WebSocket streaming, go to [WebSocket Streaming \(p. 267\)](#).

## Getting started with batch transcription

Get started with Amazon Transcribe Medical batch transcription in the AWS Management Console to transcribe your audio file. To start a transcription job, your audio file must be stored in Amazon S3 bucket and its file format must be either FLAC, WAV, MP3, or MP4.

### To start a batch medical transcription job

1. Open the [AWS Management Console](#).
2. From the left menu, look under **Amazon Transcribe Medical** and choose **Transcription jobs**.
3. Choose **Create job**
4. Specify the details of your job. Choose the **Info** links if you need help
5. When you finished specifying the details of your job, choose **Create** to start a medical transcription job.

[Batch transcription overview \(p. 243\)](#)

## Streaming transcription

Amazon Transcribe Medical streaming transcription enables you to send an audio stream and receive a stream of text in real time. The API makes it easy for developers to add real-time speech-to-text capability to their applications.

The following table shows which languages are available for streaming transcription and how you can access them.

Language	Sample rate	Available in
US English (en-US)	16 kHz, 8 kHz	<a href="#">AWS Management Console</a> , <a href="#">StartMedicalStreamTranscription API</a> , and <a href="#">WebSocket request</a>

If you are using HTTP/2, we provide an HTTP/2 streaming client that handles retrying the connection when there are transient problems on the network. You can use this client as a starting point for your own applications. To use Amazon Transcribe Medical streaming with the WebSocket protocol, you can create your own client.

Streaming transcription takes a stream of your audio data and transcribes it in real time. The transcription is returned to your application in a stream of transcription events.

Amazon Transcribe Medical breaks your incoming audio stream based on natural speech segments, such as a change in speaker or a pause in the audio. The transcription is returned progressively to your application, with each response containing more transcribed speech until the entire segment is transcribed.

In the following example, each line is a partial result transcription output of an audio segment being streamed.

```
The  
The PE.
```

```
The pain.  
The patient.  
The patient was  
The patient was in  
The patient was entered.  
The patient was entered, and, uh  
The patient was I/O.  
The patient was I/O of  
The patient was I/O of some  
The patient was I/O sign.  
The patient was I/O of Sinus.  
The patient was I/O of Sinus rhyth.  
The patient was I/O of Sinus rhythm.  
The patient was I/O of Sinus rhythm with
```

Each [Result](#) object in the response contains a field called `IsPartial` that indicates whether the response is a partial response containing the transcription results so far or if it is a complete transcription of the audio segment.

Each [Result](#) object also contains the start time and end time of the term from the audio stream so that you can, for example, synchronize the transcription with the video.

The following example is a partial transcription response.

```
{
    "Transcript": {
        "Results": [
            {
                "Alternatives": [
                    {
                        "Items": [
                            {
                                "Content": "The",
                                "EndTime": 1.07,
                                "StartTime": 1.04,
                                "Type": "pronunciation",
                                "VocabularyFilterMatch": false
                            },
                            {
                                "Content": "patient",
                                "EndTime": 1.5,
                                "StartTime": 1.08,
                                "Type": "pronunciation",
                                "VocabularyFilterMatch": false
                            },
                            {
                                "Content": "was",
                                "EndTime": 1.61,
                                "StartTime": 1.51,
                                "Type": "pronunciation",
                                "VocabularyFilterMatch": false
                            },
                            {
                                "Content": "I/O",
                                "EndTime": 2.25,
                                "StartTime": 2.06,
                                "Type": "pronunciation",
                                "VocabularyFilterMatch": false
                            },
                            {
                                "Content": "of",
                                "EndTime": 2.34,
                                "StartTime": 2.26,
                                "Type": "pronunciation",
                                "VocabularyFilterMatch": false
                            }
                        ]
                    }
                ]
            }
        ]
    }
}
```

```

        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Content": "Sinus",
        "EndTime": 2.71,
        "StartTime": 2.35,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Content": "rhythm",
        "EndTime": 3.07,
        "StartTime": 2.72,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Content": "with",
        "EndTime": 3.68,
        "StartTime": 3.49,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    }
],
"Transcript": "The patient was I/O of Sinus rhythm with"
}
],
"EndTime": 3.75,
"IsPartial": true,
"ResultId": "93b1df2b-8702-4c91-892a-ace3b65a6477",
"StartTime": 1.04
}
]
}
}

```

The following example shows the transcription results for a fully transcribed speech segment.

```

{
    "Transcript": {
        "Results": [
            {
                "Alternatives": [
                    {
                        "Items": [
                            {
                                "Confidence": 0.99,
                                "Content": "The",
                                "EndTime": 1.12,
                                "StartTime": 1.04,
                                "Type": "pronunciation",
                                "VocabularyFilterMatch": false
                            },
                            {
                                "Confidence": 0.99,
                                "Content": "patient",
                                "EndTime": 1.53,
                                "StartTime": 1.13,
                                "Type": "pronunciation",
                                "VocabularyFilterMatch": false
                            },
                            {
                                "Confidence": 1,
                                "Content": "was",
                                "EndTime": 1.53,
                                "StartTime": 1.13,
                                "Type": "pronunciation",
                                "VocabularyFilterMatch": false
                            }
                        ]
                    }
                ]
            }
        ]
    }
}

```

```
        "EndTime": 1.73,
        "StartTime": 1.54,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 1,
        "Content": "I/O",
        "EndTime": 2.3,
        "StartTime": 2.12,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 1,
        "Content": "of",
        "EndTime": 2.39,
        "StartTime": 2.31,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 1,
        "Content": "Sinus",
        "EndTime": 2.82,
        "StartTime": 2.4,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 0.99,
        "Content": "rhythm",
        "EndTime": 3.32,
        "StartTime": 2.83,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 0.99,
        "Content": "with",
        "EndTime": 3.72,
        "StartTime": 3.49,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 0.99,
        "Content": "a",
        "EndTime": 3.78,
        "StartTime": 3.73,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 1,
        "Content": "heart",
        "EndTime": 4.02,
        "StartTime": 3.79,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 1,
        "Content": "rate",
        "EndTime": 4.19,
        "StartTime": 4.03,
```

```

        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 0.99,
        "Content": "of",
        "EndTime": 4.26,
        "StartTime": 4.2,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 1,
        "Content": "75",
        "EndTime": 4.81,
        "StartTime": 4.27,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Confidence": 0.97,
        "Content": "bpm",
        "EndTime": 5.47,
        "StartTime": 4.82,
        "Type": "pronunciation",
        "VocabularyFilterMatch": false
    },
    {
        "Content": ".",
        "EndTime": 5.47,
        "StartTime": 5.47,
        "Type": "punctuation",
        "VocabularyFilterMatch": false
    }
],
    "Transcript": "The patient was I/O of Sinus rhythm with a heart
rate of 75 bpm."
}
],
    "EndTime": 5.53,
    "IsPartial": false,
    "ResultId": "93b1df2b-8702-4c91-892a-ace3b65a6477",
    "StartTime": 1.04
}
]
}
}

```

Each word, phrase or punctuation mark in the transcription output is an *item*. Each word or phrase has a *confidence* score. The confidence score is a value between 0 and 1 that indicates how confident Amazon Transcribe Medical is that it correctly transcribed the item. A confidence score with a larger value indicates that Amazon Transcribe Medical is more confident that it transcribed the item correctly.

The preceding example shows "I/O" in the transcription output, which is an abbreviation for "in and out". For more information on how Amazon Transcribe Medical uses abbreviations in the transcription output, see [Transcribing medical terms and measurements \(p. 245\)](#).

## Topics

- [Event stream encoding \(p. 254\)](#)
- [Using Amazon Transcribe Medical streaming with HTTP/2 \(p. 255\)](#)
- [Establish a bi-directional connection using the WebSocket protocol \(p. 267\)](#)

## Event stream encoding

Event stream encoding provides bidirectional communication using messages between a client and a server. Data frames sent to the Amazon Transcribe Medical streaming service are encoded in this format. The response from Amazon Transcribe Medical also uses this encoding.

Each message consists of two sections: the prelude and the data. The prelude consists of:

1. The total byte length of the message
2. The combined byte length of all of the headers

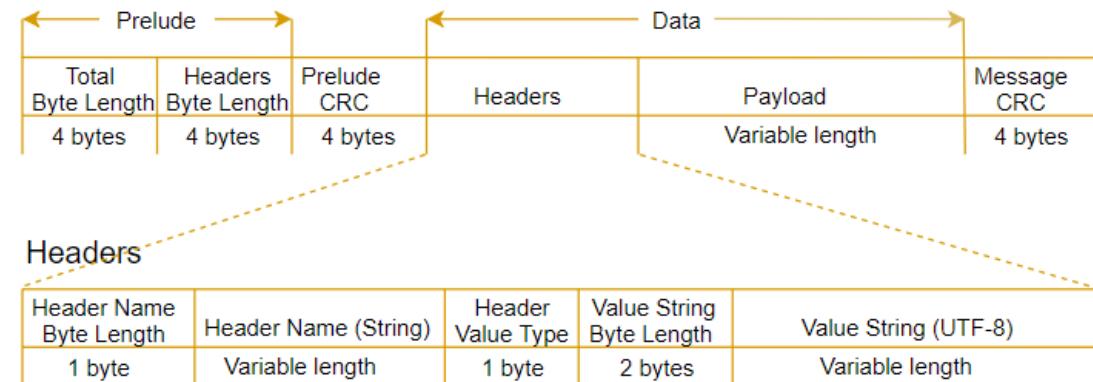
The data section consists of:

1. The headers
2. A payload

Each section ends with a 4-byte big-endian integer CRC checksum. The message CRC checksum is for both the prelude section and the data section. Amazon Transcribe Medical uses CRC32 (often referred to as GZIP CRC32) to calculate both CRCs. For more information about CRC32, see [GZIP file format specification version 4.3](#).

Total message overhead, including the prelude and both checksums, is 16 bytes.

The following diagram shows the components that make up a message and a header. There are multiple headers per message.



Each message contains the following components:

- **Prelude:** Always a fixed size of 8 bytes, two fields of 4 bytes each.
  - *First 4 bytes:* The total byte-length. This is the big-endian integer byte-length of the entire message, including the 4-byte length field itself.
  - *Second 4 bytes:* The headers byte-length. This is the big-endian integer byte-length of the headers portion of the message, excluding the headers length field itself.
- **Prelude CRC:** The 4-byte CRC checksum for the prelude portion of the message, excluding the CRC itself. The prelude has a separate CRC from the message CRC to ensure that Amazon Transcribe Medical can detect corrupted byte-length information immediately without causing errors such as buffer overruns.
- **Headers:** Metadata annotating the message, such as the message type, content type, and so on. Messages have multiple headers. Headers are key-value pairs where the key is a UTF-8 string. Headers can appear in any order in the headers portion of the message and any given header can appear only once. For the required header types, see the following sections.

- **Payload:** The audio content to be transcribed.
- **Message CRC:** The 4-byte CRC checksum from the start of the message to the start of the checksum. That is, everything in the message except the CRC itself.

Each header contains the following components. There are multiple headers per frame.

- **Header name byte-length:** The byte-length of the header name.
- **Header name:** The name of the header indicating the header type. For valid values, see the following frame descriptions.
- **Header value type:** An enumeration indicating the header value.

The following shows the possible values for the header and what they indicate.

- 0 – TRUE
- 1 – FALSE
- 2 – BYTE
- 3 – SHORT
- 4 – INTEGER
- 5 – LONG
- 6 – BYTE ARRAY
- 7 – STRING
- 8 – TIMESTAMP
- 9 – UUID
- **Value string byte length:** The byte-length of the header value string.
- **Header value:** The value of the header string. Valid values for this field depend on the type of header. For valid values, see the following frame descriptions.

## Using Amazon Transcribe Medical streaming with HTTP/2

Amazon Transcribe Medical uses a format called *event stream encoding* for streaming-med transcription. This format encodes binary data with header information that describes the contents of each event. For more information, see [Event stream encoding \(p. 48\)](#). You can use this information for applications that call the Amazon Transcribe Medical endpoint without using an SDK.

When Amazon Transcribe Medical uses the [HTTP/2 protocol](#) for streaming medical transcriptions, the key components for a streaming medical request are:

- A header frame. This contains the HTTP/2 headers for the request, and a signature in the authorization header that Amazon Transcribe Medical uses as a seed signature to sign the following data frames.
- One or more message frames in event stream encoding. The frame contains metadata and the raw audio bytes.
- An end frame. This is a signed message in event stream encoding with an empty body.

### Streaming request

To make a streaming request, you use the [StartMedicalStreamTranscription](#) API.

## Header frame

The header frame is the authorization frame for the streaming transcription. Amazon Transcribe Medical uses the value of the `authorization` header as the seed for generating a chain of authorization headers for the data frames in the request.

### Required headers

The header frame of a request to Amazon Transcribe Medical requires the following HTTP/2 headers.

```
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-MEDAWS4-HMAC-SHA256-EVENTS
x-amz-date: 20220208T235959Z
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
Content-type: application/vnd.amazon.eventstream
transfer-encoding: chunked
```

In the request, use the following values for the `host`, `authorization`, and `x-amz-date` headers:

- **host:** Use the AWS Region where you are calling Amazon Transcribe Medical. For a list of valid AWS Regions, see [AWS Regions and Endpoints](#) in the [AWS General Reference](#).
- **authorization:** This is a generated field. To learn more about creating a signature, see [Signing AWS requests with Signature Version 4](#).
- **x-amz-date:** The date and time the signature is created. The format is YYYYMMDDTHHMMSSZ, where YYYY=year, MM=month, DD=day, HH=hour, MM=minute, SS=seconds, and 'T' and 'Z' are fixed characters. For more information, refer to [Handling Dates in Signature Version 4](#).

For more information about the headers specific to Amazon Transcribe Medical, see the [StartMedicalStreamTranscription API](#).

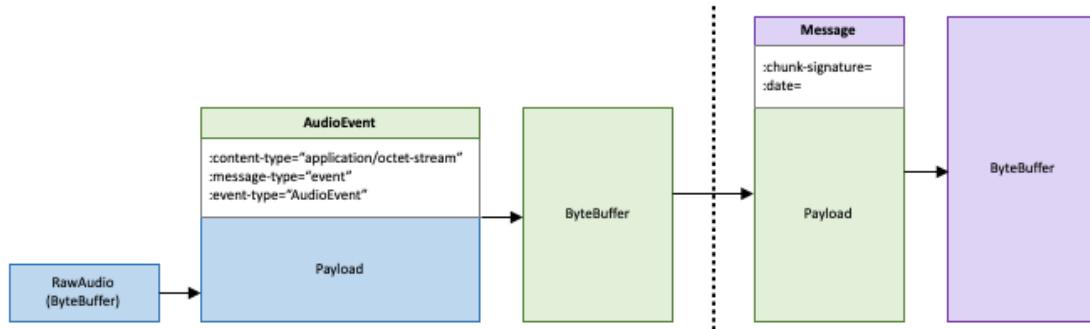
## Data frames

Each request contains one or more data frames. The data frames use event stream encoding. The encoding supports bidirectional data transmission between a client and a server.

There are two steps to creating a data frame:

1. Combine the raw audio data with metadata to create the payload of the request.
2. Combine the payload with a signature to form the event message that is sent to Amazon Transcribe Medical.

The following diagram shows how this works.



### Create the audio event

To create the message to send to Amazon Transcribe Medical, create the audio event. Combine the headers described in the following table with a chunk of audio bytes into an event-encoded message.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	<code>:content-type</code>	7	24	application/octet-stream
11	<code>:event-type</code>	7	10	AudioEvent
13	<code>:message-type</code>	7	5	event

To create the payload for the event message, use a buffer in raw-byte format.

### Create the message

Create a data frame using the audio event payload to send to Amazon Transcribe Medical. The data frame contains event-encoding headers that include the current date and a signature for the audio chunk and the audio event. To indicate to Amazon Transcribe Medical that the audio stream is complete, send an empty data frame that contains only the date and signature.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value
16	<code>:chunk-signature</code>	6	varies	generated signature
5	<code>:date</code>	8	8	timestamp

To create the signature for the data frame, first create a string to sign, and then calculate the signature for the event. Construct the string to sign as follows.

```

String stringToSign =
    "AWS4-HMAC-SHA256-PAYLOAD" +
    "\n" +
    DATE +
    "\n" +
    KEYPATH +
    "\n" +
    Hex(priorSignature) +
    "\n" +
  
```

```
HexHash(nonSignatureHeaders) +
"\n" +
HexHash(payload);
```

- **DATE:** The current date and time in Universal Time Coordinated (UTC) and using the [ISO 8601 format](#). Don't include milliseconds in the date. For example, 20220127T223754Z is 22:37:54 on 1/27/2022.
- **KEYPATH:** The signature scope in the format `date/region/service/aws4_request`. For example, `20220127/us-west-2/transcribe/aws4_request`.
- **priorSignature:** The signature for the previous frame. For the first data frame, use the signature of the header frame.
- **nonSignatureHeaders:** The DATE header encoded as a string.
- **payload:** The byte buffer containing the audio event data.
- **Hex:** A function that encodes its input into a hexadecimal representation.
- **HexHash:** A function that first creates a SHA-256 hash of its input and then uses the Hex function to encode the hash.

After you have constructed the string to sign, sign it using the key that you derived for Signature Version 4, as follows. For details, see [Examples of How to Derive a Signing Key for Signature Version 4](#) in the [AWS General Reference](#).

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

- **HMACSHA256:** A function that creates a signature using the SHA-256 hash function.
- **derivedSigningKey:** The Signature Version 4 signing key.
- **stringToSign:** The string that you calculated for the data frame.

After you have calculated the signature for the data frame, construct a byte buffer containing the date, the signature, and the audio event payload. Send the byte array to Amazon Transcribe Medical for transcription.

## End frame

To indicate that the audio stream is complete, send an end frame to Amazon Transcribe Medical. The *end frame* is a data frame with an empty payload. You construct the end frame the same way that you construct a data frame.

## Streaming response

Responses from Amazon Transcribe Medical are also sent using event stream encoding. Use this information to decode a response from the [StartMedicalStreamTranscription](#) API.

### Transcription response

A transcription response is event stream encoded. It contains the standard prelude and the following headers.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	15	TranscriptEvent

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:message-type	7	5	event

For details, see [Event stream encoding \(p. 48\)](#).

When the response is decoded, it contains the following information:

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

JSON transcription information
```

For an example of the JSON structure returned by Amazon Transcribe Medical, see [Using Amazon Transcribe Medical streaming with HTTP/2 \(p. 255\)](#).

## Exception response

If there is an error in processing your transcription stream, Amazon Transcribe Medical sends an exception response. The response is event stream encoded. For details, see [Event stream encoding \(p. 48\)](#).

The response contains the standard prelude and the following headers.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	16	application/json
11	:event-type	7	19	BadRequestException
13	:message-type	7	9	exception

When the exception response is decoded, it contains the following information.

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

Exception message
```

## Example request and response

The following is an end-to-end example of a streaming transcription request. In this example, binary data are represented as base64-encoded strings. In an actual response, the data are raw bytes.

### Step 1: Start the session with Amazon Transcribe Medical

To start the session, send an HTTP/2 request to Amazon Transcribe Medical.

```
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-language-code: en-US
x-amzn-transcribe-media-encoding: flac
x-amzn-transcribe-sample-rate: 16000
content-type: application/vnd.amazon.eventstream
transfer-encoding: chunked
```

## Step 2: Send authentication information to Amazon Transcribe Medical

Amazon Transcribe Medical sends the following response.

```
HTTP/2.0 200
x-amzn-transcribe-language-code: en-US
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
x-amzn-transcribe-media-encoding: flac
x-amzn-RequestId: 8a08df7d-5998-48bf-a303-484355b4ab4e
content-type: application/vnd.amazon.eventstream
```

## Step 3: Create an audio event

Create an audio event containing the audio data to send. For details, see [Event stream encoding \(p. 254\)](#). The binary data in this request are base64-encoded. In an actual request, the data are raw bytes.

```
:content-type: "application/octet-stream"
:event-type: "AudioEvent"
:message-type: "event"

Uk1GRjzxPQBXQVFZm10IBAAAAABAAEAgD4AAAB9AACABAAZGF0YVTwPQAAAAAAAAAAAAAAD//wIA/f8EAA==
```

## Step 4: Create an audio event message

Create an audio message that contains the audio data to send to Amazon Transcribe Medical. For details, see [Event stream encoding \(p. 254\)](#). The audio event data in this example are base64-encoded. In an actual request, the data are raw bytes.

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

AAAA0gAAAIKVorFcTTcjb250ZW50LXR5cGUHABhhcHBsaWNhdGlvbi9vY3RldC1zdHJ1YW0LOmV2ZW50LXR5
cGUHAApBdWRpb0V2ZW50DTptZXNzYWdlLXR5cGUHAAVldmVudAxDb256ZW50LVR5cGUHABphcHBsaWNhdGlv
bi94LWFtei1qc29uLTEuMVJJRky88T0AV0FWRWZtdCAQAAAAQABAIA+AAAAfQAAAAGAQAGRhdGFU8D0AAAAAA
AAAAAAAAAA//8CAP3/BAC7QLFF
```

## Step 5: Use the response from Amazon Transcribe Medical

Amazon Transcribe Medical creates a stream of transcription events that it sends to your application. The events are sent in raw-byte format. In this example, the bytes are base64-encoded.

Amazon Transcribe Medical sends the following response.

```
AAAAUwAAAEP1RHpYBTpkYXR1CAAAAWiXUkMLEDpjaHVuay1zaWduYXR1cmUGACt6Zy+uymwEK2SrLp/zVBI
5eGn83jdBwCaRUBJA+eaDafqjqi=
```

To see the transcription results, decode the raw bytes using event stream encoding.

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

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

For an example of the JSON structure returned by Amazon Transcribe Medical, see [Event stream encoding \(p. 254\)](#).

## Step 6: End the transcription stream

Finally, send an empty audio event to Amazon Transcribe Medical to end the transcription stream. Create the audio event exactly like any other, except with an empty payload. Sign the event and include the signature in the `:chunk-signature` header, as follows.

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

## HTTP/2 streaming retry client

You can use the following code in your applications to handle retry logic for Amazon Transcribe Medical streaming transcription. The code provides tolerance for intermittent failures in the connection to Amazon Transcribe Medical. There are two parts of the client: an interface that you implement for your application, and the retry client itself.

### Streaming retry client code

This code implements a streaming retry client. It manages the connection to Amazon Transcribe Medical and retries sending data when there are errors on the connection. For example, if there is a transient error on the network, this client resends the request that failed.

The retry client has two properties that control the behavior of the client. You can set:

- The maximum number of times that the client should attempt before failing. Reduce this value to make your application stop retrying sooner when there are network issues. The default is 10.
- The time in milliseconds that the client should wait between retries. Longer times raise the risk of losing data, shorter times raise the risk of your application being throttled. The default is 100 milliseconds.

The following is the client. You can copy this code to your application or use it as a starting point for your own client.

```
public class TranscribeStreamingRetryClient {  
  
    private static final int DEFAULT_MAX_RETRIES = 10;  
    private static final int DEFAULT_MAX_SLEEP_TIME_MILLS = 100;
```

```
private static final Logger log =
LoggerFactory.getLogger(TranscribeStreamingRetryClient.class);
private final TranscribeStreamingAsyncClient client;
List<Class<?>> nonRetriableExceptions = Arrays.asList(BadRequestException.class);
private int maxRetries = DEFAULT_MAX_RETRIES;
private int sleepTime = DEFAULT_MAX_SLEEP_TIME_MILLS;

/**
 * Create a TranscribeStreamingRetryClient with given credential and configuration
 */
public TranscribeStreamingRetryClient(AwsCredentialsProvider creds,
                                       String endpoint, Region region) throws
URISyntaxException {
    this(TranscribeStreamingAsyncClient.builder()
        .overrideConfiguration(
            c -> c.putAdvancedOption(
                SdkAdvancedClientOption.SIGNER,
                EventStreamAws4Signer.create())))
    .credentialsProvider(creds)
    .endpointOverride(new URI(endpoint))
    .region(region)
    .build());
}

/**
 * Initiate TranscribeStreamingRetryClient with TranscribeStreamingAsyncClient
 */
public TranscribeStreamingRetryClient(TranscribeStreamingAsyncClient client) {
    this.client = client;
}

/**
 * Get Max retries
 */
public int getMaxRetries() {
    return maxRetries;
}

/**
 * Set Max retries
 */
public void setMaxRetries(int maxRetries) {
    this.maxRetries = maxRetries;
}

/**
 * Get sleep time
 */
public int getSleepTime() {
    return sleepTime;
}

/**
 * Set sleep time between retries
 */
public void setSleepTime(int sleepTime) {
    this.sleepTime = sleepTime;
}

/**
 * Initiate a Stream Transcription with retry.
 */
public CompletableFuture<Void> startStreamTranscription(final
StartStreamTranscriptionRequest request,
```

```

final Publisher<AudioStream>
publisher,
final
StreamTranscriptionBehavior responseHandler) {

    CompletableFuture<Void> finalFuture = new CompletableFuture<>();

    recursiveStartStream(rebuildRequestWithSession(request), publisher,
responseHandler, finalFuture, 0);

    return finalFuture;
}

/**
 * Recursively call startStreamTranscription() until the request is completed or we run
out of retries.
*/
private void recursiveStartStream(final StartStreamTranscriptionRequest request,
final Publisher<AudioStream> publisher,
final StreamTranscriptionBehavior responseHandler,
final CompletableFuture<Void> finalFuture,
final int retryAttempt) {
    CompletableFuture<Void> result = client.startStreamTranscription(request,
publisher,
    getResponseHandler(responseHandler));
    result.whenComplete((r, e) -> {
        if (e != null) {
            log.debug("Error occurred:", e);

            if (retryAttempt <= maxRetries && isExceptionRetriable(e)) {
                log.debug("Retriable error occurred and will be retried.");
                log.debug("Sleeping for sometime before retrying...");
                try {
                    Thread.sleep(sleepTime);
                } catch (InterruptedException e1) {
                    log.debug("Unable to sleep. Failed with exception: ", e);
                    e1.printStackTrace();
                }
                log.debug("Making retry attempt: " + (retryAttempt + 1));
                recursiveStartStream(request, publisher, responseHandler, finalFuture,
retryAttempt + 1);
            } else {
                log.error("Encountered unretrievable exception or ran out of retries. ");
                responseHandler.onError(e);
                finalFuture.completeExceptionally(e);
            }
        } else {
            responseHandler.onComplete();
            finalFuture.complete(null);
        }
    });
}

private StartStreamTranscriptionRequest
rebuildRequestWithSession(StartStreamTranscriptionRequest request) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(request.languageCode())
        .mediaEncoding(request.mediaEncoding())
        .mediaSampleRateHertz(request.mediaSampleRateHertz())
        .sessionId(UUID.randomUUID().toString())
        .build();
}

/**
 * StartStreamTranscriptionResponseHandler implements subscriber of transcript stream

```

```

        * Output is printed to standard output
    */
private StartStreamTranscriptionResponseHandler getResponseHandler(
    StreamTranscriptionBehavior transcriptionBehavior) {
    final StartStreamTranscriptionResponseHandler build =
StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            transcriptionBehavior.onResponse(r);
        })
        .onError(e -> {
            //Do nothing here. Don't close any streams that shouldn't be cleaned up
yet.
        })
        .onComplete(() -> {
            //Do nothing here. Don't close any streams that shouldn't be cleaned up
yet.
        })

        .subscriber(event -> transcriptionBehavior.onStream(event))
        .build();
    return build;
}

/**
 * Check if the exception can be retried.
 */
private boolean isExceptionRetriable(Throwable e) {
    e.printStackTrace();

    return nonRetriableExceptions.contains(e.getClass());
}

public void close() {
    this.client.close();
}
}

```

## Streaming retry client interface code

This interface is similar to the response handler used in the getting started example. It implements the same event handlers. Implement this interface to use the streaming-med retry client.

```

package com.amazonaws.transcribestreaming;

import
    software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponse;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptResultStream;

/**
 * Defines how a stream response should be handled.
 * You should build a class implementing this interface to define the behavior.
 */
public interface StreamTranscriptionBehavior {
    /**
     * Defines how to respond when encountering an error on the stream transcription.
     */
    void onError(Throwable e);

    /**
     * Defines how to respond to the Transcript result stream.
     */
}

```

```
 */
void onStream(TranscriptResultStream e);

/**
 * Defines what to do on initiating a stream connection with the service.
 */
void onResponse(StartStreamTranscriptionResponse r);

/**
 * Defines what to do on stream completion
 */
void onComplete();
}
```

The following is an example implementation of the `StreamTranscriptionBehavior` interface. You can use this implementation or use it as a starting point for your own implementation.

```
package com.amazonaws.transcribestreaming.retryclient;

import com.amazonaws.transcribestreaming.retryclient.StreamTranscriptionBehavior;
import software.amazon.awssdk.services.transcribestreaming.model.Result;
import software.amazon.awssdk.services.transcribestreaming.model.StartStreamTranscriptionResponse;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptResultStream;

import java.util.List;

/**
 * Implementation of StreamTranscriptionBehavior to define how a stream response should be
 * handled.
 *
 * COPYRIGHT:
 *
 * Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 */
public class StreamTranscriptionBehaviorImpl implements StreamTranscriptionBehavior {

    @Override
    public void onError(Throwable e) {
        System.out.println("==== Failure Encountered ====");
        e.printStackTrace();
    }

    @Override
    public void onStream(TranscriptResultStream e) {
        // EventResultStream has other fields related to the timestamp of the transcripts
        // in it.
    }
}
```

```
// Please refer to the javadoc of TranscriptResultStream for more details
List<Result> results = ((TranscriptEvent) e).transcript().results();
if (results.size() > 0) {
    if (results.get(0).alternatives().size() > 0)
        if (!results.get(0).alternatives().get(0).transcript().isEmpty())
            System.out.println(results.get(0).alternatives().get(0).transcript());
}
}

@Override
public void onResponse(StartStreamTranscriptionResponse r) {

    System.out.println(String.format("==> Received Initial response. Request Id: %s
==>", r.requestId()));
}

@Override
public void onComplete() {
    System.out.println("==> All records stream successfully ==>");
}
```

## Next step

[Using the HTTP/2 retry client \(p. 266\)](#)

## Using the HTTP/2 retry client

The following is a sample application that uses the retry client to transcribe audio from either a file or your microphone. You can use this application to test the client, or you can use it as a starting point for your own applications.

To run the sample, do the following:

- Copy the retry client to your workspace. See [Streaming retry client code \(p. 261\)](#).
- Copy the retry client interface to your workspace. Implement the interface, or you can use the sample implementation. See [Streaming retry client interface code \(p. 264\)](#).
- Copy the sample application to your workspace. Build and run the application.

```
public class StreamingRetryApp {
    private static final String endpoint = "endpoint";
    private static final Region region = Region.US_EAST_1;
    private static final int sample_rate = 28800;
    private static final String encoding = " ";
    private static final String language = LanguageCode.EN_US.toString();

    public static void main(String args[]) throws URISyntaxException, ExecutionException,
    InterruptedException, LineUnavailableException, FileNotFoundException {
        /**
         * Create Amazon Transcribe streaming retry client.
         */
        TranscribeStreamingRetryClient client = new
        TranscribeStreamingRetryClient(EnvironmentVariableCredentialsProvider.create() ,endpoint,
        region);

        StartStreamTranscriptionRequest request = StartStreamTranscriptionRequest.builder()
            .languageCode(language)
            .mediaEncoding(encoding)
            .mediaSampleRateHertz(sample_rate)
```

```

        .build();

    /**
     * Start real-time speech recognition. The Amazon Transcribe streaming java client
     uses the Reactive-streams
     * interface. For reference on Reactive-streams:
     * https://github.com/reactive-streams/reactive-streams-jvm
     */
    CompletableFuture<Void> result = client.startStreamTranscription(
        /**
         * Request parameters. Refer to API documentation for details.
         */
        request,
        /**
         * Provide an input audio stream.
         * For input from a microphone, use getStreamFromMic().
         * For input from a file, use getStreamFromFile().
         */
        new AudioStreamPublisher(
            new FileInputStream(new File("FileName"))),
        /**
         * Object that defines the behavior on how to handle the stream
         */
        new StreamTranscriptionBehaviorImpl());

    /**
     * Synchronous wait for stream to close, and close client connection
     */
    result.get();
    client.close();
}

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

```

## Establish a bi-directional connection using the WebSocket protocol

You can use the [WebSocket protocol](#) to open a bi-directional connection that is designed to be secure with Amazon Transcribe Medical. You encode the audio stream with event stream encoding, and Amazon Transcribe Medical returns a stream of text in JSON blobs that are also encoded using event stream

encoding. For more information, see [Event stream encoding \(p. 254\)](#). You can use the information in this section to create applications using the WebSocket library of your choice.

### Topics

- [Adding a policy for WebSocket requests to your IAM role \(p. 268\)](#)
- [Creating a pre-signed URI \(p. 268\)](#)
- [Handling the WebSocket upgrade response \(p. 272\)](#)
- [Making a WebSocket streaming request \(p. 273\)](#)
- [Handling a WebSocket streaming response \(p. 273\)](#)
- [Handling WebSocket streaming errors \(p. 274\)](#)

## Adding a policy for WebSocket requests to your IAM role

To use the WebSocket protocol to call Amazon Transcribe Medical, you need to attach the following policy to the IAM role that makes the request.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "transcribemedicalstreaming",  
            "Effect": "Allow",  
            "Action": "transcribe:StartMedicalStreamTranscription",  
            "Resource": "*"  
        }  
    ]  
}
```

## Creating a pre-signed URI

You need to construct a URI for your WebSocket request that contains the information needed to set up communication between your application and Amazon Transcribe Medical. To open a bi-directional connection, you must use port 8443. WebSocket streaming-med uses the Amazon Signature Version 4 process for signing requests. Signing the request helps to verify the identity of the requester and to protect your audio data in transit. It also protects against potential replay attacks. For more information about Signature Version 4, see [Signing AWS API Requests](#) in the *Amazon Web Services General Reference*.

The URI has the following format. Line breaks have been added for readability.

```
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=250  
&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=specialty  
&type=type
```

For TYPE, it is best to select CONVERSATION if your use case involves multiple speakers engaged in a discussion. An example would be a conversation between a clinician and a patient. Select DICTATION

if your use case involves a single speaker where a person is dictating speech. An example would be if a physician is dictating medical notes for data entry purposes after a patient encounter.

Use the following values for the URI parameters:

- **language-code** – The language code for the input audio. The valid value is en-US.
- **media-encoding** – The encoding used for the input audio. The valid value is pcm.
- **sample-rate** – The sample rate of the input audio in hertz; sample rates of 16,000 Hz or higher are accepted.
- **sessionId** – Optional. An identifier for the transcription session. If you don't provide a session ID, Amazon Transcribe Medical generates one for you and returns it in the response.
- **specialty** – The specialty of the medical domain. Valid values are PRIMARYCARE, CARDIOLOGY, NEUROLOGY, ONCOLOGY, RADIOLOGY, and UROLOGY.
- **type** – The type of audio. Must be DICTATION or CONVERSATION.

The remaining parameters are Signature Version 4 parameters:

- **X-Amz-Algorithm** – The algorithm you're using in the signing process. The only valid value is AWS4-HMAC-SHA256.
- **X-Amz-Credential** – A string formed by concatenating your access key ID, the date, your AWS Region, the AWS service you're calling, and a standard termination term. Each of these components is separated by '%2F'. This is the general form:

*your-access-key-id%2Fdate%2FAWS-Region%2FAWS-service%2Faws4\_request*

For more information, refer to [Handling Dates in Signature Version 4](#).

- **X-Amz-Date** – The date and time the signature is created. The format is YYYYMMDDTHHMMSSZ, where YYYY=year, MM=month, DD=day, HH=hour, MM=minute, SS=seconds, and 'T' and 'Z' are fixed characters. For more information, refer to [Handling Dates in Signature Version 4](#).
- **X-Amz-Expires** – The length of time in seconds until the credentials expire. The maximum value is 300 seconds (5 minutes).
- **X-Amz-Security-Token** – Optional. A Signature Version 4 token for temporary credentials. If you specify this parameter, include it in the canonical request. For more information, see [Requesting Temporary Security Credentials in the AWS Identity and Access Management User Guide](#).
- **X-Amz-Signature** – The Signature Version 4 signature that you generated for the request.
- **X-Amz-SignedHeaders** – The headers that are signed when creating the signature for the request. The only valid value is host.

To construct the URI for the request and create the Signature Version 4 signature, use the following steps. The examples are in pseudocode.

### Task 1: Create a canonical request

Create a string that includes information from your request in a standardized format. This ensures that when AWS receives the request, it can calculate the same signature that you calculate in Task 3. For more information, see [Create a Canonical Request for Signature Version 4](#) in the *AWS General Reference*.

1. Define variables for the request in your application.

```
# HTTP verb
method = "GET"
# Service name
service = "transcribe"
# Region
region = "us-west-2"
```

```
# Amazon Transcribe streaming-med endpoint
endpoint = "https://transcribestreaming.us-west-2.amazonaws.com:8443"
# Host
host = "transcribestreaming.us-west-2.amazonaws.com:8443"
# Date and time of request
amz-date = YYYYMMDD'T'HHMMSS'Z'
# Date without time for credential scope
datestamp = YYYYMMDD
```

2. Create a canonical URI. The canonical URI is the part of the URI between the domain and the query string.

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

3. Create the canonical headers and signed headers. Note the trailing "\n" in the canonical headers.
  - Append the lowercase header name followed by a colon.
  - Append a comma-separated list of values for that header. Do not sort the values in headers that have multiple values.
  - Append a new line (\n).

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

4. Match the algorithm to the hashing algorithm. You must use SHA-256.

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

5. Create the credential scope, which scopes the derived key to the date, AWS Region, and service to which the request is made.

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

6. Create the canonical query string. Query string values must be URI-encoded and sorted by name.

- Sort the parameter names by character code point in ascending order. Parameters with duplicate names should be sorted by value. For example, a parameter name that begins with the uppercase letter F precedes a parameter name that begins with a lowercase letter b.
- Do not URI-encode any of the unreserved characters that [RFC 3986](#) defines: A-Z, a-z, 0-9, hyphen (-), underscore (\_), period (.), and tilde (~).
- Percent-encode all other characters with %XY, where X and Y are hexadecimal characters (0-9 and uppercase A-F). For example, the space character must be encoded as %20 (not using '+', as some encoding schemes do) and extended UTF-8 characters must be in the form %XY%ZA%BC.
- Double-encode any equals (=) characters in parameter values.

```
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=250"
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"
canonical_querystring += "&specialty=PRIMARYCARE"
canonical_querystring += "&type=DICTATION"
```

7. Create a hash of the payload. For a GET request, the payload is an empty string.

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

8. Combine all of the elements to create the canonical request.

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

### Task 2: Create the string to sign

The string to sign contains meta information about your request. You use the string to sign in the next step when you calculate the request signature. For more information, see [Create a String to Sign for Signature Version 4](#) in the *AWS General Reference*.

- Create the string.

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

### Task 3: Calculate the signature

You derive a signing key from your AWS secret access key. For a greater degree of protection, the derived key is specific to the date, service, and AWS Region. You use the derived key to sign the request. For more information, see [Calculate the Signature for AWS Signature Version 4](#) in the *AWS General Reference*.

The code assumes that you have implemented the `GetSignatureKey` function to derive a signing key. For more information and example functions, see [Examples of How to Derive a Signing Key for Signature Version 4](#) in the *AWS General Reference*.

The function `HMAC(key, data)` represents an HMAC-SHA256 function that returns the results in binary format.

- Create the signing key and sign the string to sign.

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

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

### Task 4: Add signing information to the request and create the request URI

After you calculate the signature, add it to the query string. For more information, see [Add the Signature to the HTTP Request](#) in the *AWS General Reference*.

1. Add the authentication information to the query string.

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

2. Create the URI for the request.

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

You use the request URI with your WebSocket library to make the request to the Amazon Transcribe Medical service.

## Including WebSocket request headers

The request to Amazon Transcribe Medical must include the following headers. Typically these med headers are managed by your WebSocket client library.

```
Host: transcribestreaming.us-west-2.amazonaws.com:8443
Connection: Upgrade
Upgrade: websocket
Origin: request source
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: random key
```

Use the following values for the headers:

- **Connection** – Always Upgrade.
- **Upgrade** – Always websocket.
- **Origin** – The URI of the WebSocket client.
- **Sec-WebSocket-Version** – The version of the WebSocket protocol to use.
- **Sec-WebSocket-Key** – A base-64 encoded randomly generated string that identifies the request.

## Handling the WebSocket upgrade response

When Amazon Transcribe Medical receives your WebSocket request, it responds with a WebSocket upgrade response. Typically, your WebSocket library manages this response and sets up a socket for communications with Amazon Transcribe Medical.

The following is the response from Amazon Transcribe Medical. Line breaks have been added to the `websocket-location` header for readability.

```
HTTP/1.1 101 WebSocket Protocol Handshake
Connection: upgrade
Upgrade: websocket
websocket-origin: https://transcribestreaming.us-west-2.amazonaws.com:8443
websocket-location: 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=250
&X-Amz-SignedHeaders=host
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
&type=dictation or conversation
&specialty=medical specialty (must be Primary Care)
```

```
&X-Amz-Signature=signature
x-amzn-RequestId: RequestId
x-amzn-SessionId: SessionId
Strict-Transport-Security: max-age=31536000
sec-websocket-accept: token
```

The response has the following values:

- **Connection** – Always Upgrade.
- **Upgrade** – Always websocket.
- **websocket-origin** – The URI of the WebSocket server that responded to the request.
- **websocket-location** – The contents of the request URI that was sent to the server.
- **x-amzn-RequestId** – An identifier for the request.
- **x-amzn-SessionId** – An identifier for a transcription session.
- **Strict-Transport-Security** – A header that informs browsers to access the endpoint using only HTTPS.
- **sec-websocket-accept** – The hash of the Sec-WebSocket-Key header sent in the request.

## Making a WebSocket streaming request

After the WebSocket connection is established, the client can start sending a sequence of audio frames. Each frame contains one data frame that is encoded in event stream encoding. For more information, see [Event stream encoding \(p. 48\)](#).

Each data frame contains three headers combined with a chunk of raw audio bytes. The following table lists and describes the headers.

Header name Byte Length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	24	application/ octet-stream
11	:event-type	7	10	AudioEvent
13	:message-type	7	5	event

To end the audio data stream, send an empty audio chunk in an event stream encoded message.

## Handling a WebSocket streaming response

The response contains event stream encoded raw bytes in the payload. It contains the standard prelude and the following headers.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	24	application/octet- stream
11	:event-type	7	15	TranscriptEvent

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:message-type	7	5	event

When you decode the binary response, you end up with a JSON structure with the results of the transcription. For an example of the JSON response, see [Streaming transcription \(p. 249\)](#).

## Handling WebSocket streaming errors

If an exception occurs while processing your request, Amazon Transcribe Medical responds with a terminal WebSocket frame containing an event stream encoded response. The response has the headers described in the following table, and the body of the response contains a descriptive error message. After sending the exception response, Amazon Transcribe Medical sends a close frame.

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	24	application/octet-stream
15	:exception-type	7	varies	varies, see below
13	:message-type	7	9	exception

The exception-type header contains one of the following values:

- **BadRequestException** – There was a client error when the stream was created, or an error occurred while streaming-med data. Make sure that your client is ready to accept data and try your request again.
- **InternalFailureException** – Amazon Transcribe Medical had a problem during the handshake with the client. Try your request again.
- **LimitExceededException** – The client exceeded the concurrent stream limit. For more information, see [Amazon Transcribe Service Quotas](#) in the *Amazon Web Services General Reference*. Reduce the number of streams that you are transcribing.
- **UnrecognizedClientException** – The WebSocket upgrade request was signed with an incorrect access key or secret key. Make sure that you are correctly creating the access key and try your request again.

In addition, Transcribe Medical can return any of the common service errors. For a list, see [Common Errors](#).

## Transcribing a medical conversation

You can use Amazon Transcribe Medical to transcribe a medical conversation between a clinician and a patient using either a batch transcription job or a real-time stream. Batch transcription jobs enable you to transcribe audio files. To ensure that Amazon Transcribe Medical produces transcription results with the highest possible accuracy, you must specify the medical specialty of the clinician in your transcription job or stream.

You can transcribe a clinician-patient visit in the following medical specialities:

- Cardiology – available in streaming transcription only
- Neurology – available in streaming transcription only
- Oncology – available in streaming transcription only
- Primary Care – includes the following types of medical practice:
  - Family medicine
  - Internal medicine
  - Obstetrics and Gynecology (OB-GYN)
  - Pediatrics
- Urology – available in streaming transcription only

You can improve transcription accuracy by using medical custom vocabularies. For information on how medical custom vocabularies work, see [Improving transcription accuracy with medical custom vocabularies \(p. 299\)](#).

By default, Amazon Transcribe Medical returns the transcription with the highest confidence level. If you'd like to configure it to return alternative transcriptions, see [Generating alternative transcriptions \(p. 315\)](#).

For information about how numbers and medical measurements appear in the transcription output, see [Transcribing numbers \(p. 244\)](#) and [Transcribing medical terms and measurements \(p. 245\)](#).

#### Topics

- [Transcribing an audio file of a medical conversation \(p. 275\)](#)
- [Transcribing a medical conversation in a real-time stream \(p. 279\)](#)
- [Identifying speakers and labeling their speech \(p. 280\)](#)
- [Transcribing multi-channel audio \(p. 287\)](#)

## Transcribing an audio file of a medical conversation

Use a batch transcription job to transcribe audio files of medical conversations. You can use this to transcribe a clinician-patient dialogue. You can start a batch transcription job in either the [StartMedicalTranscriptionJob](#) API or the AWS Management Console.

When you start a medical transcription job with the [StartMedicalTranscriptionJob](#) API, you specify `PRIMARYCARE` as the value of the `Specialty` parameter.

### AWS Management Console

#### To transcribe a clinician-patient dialogue (AWS Management Console)

To use the AWS Management Console to transcribe a clinician-patient dialogue, create a transcription job and choose **Conversation** for **Audio input type**.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Transcription jobs**.
3. Choose **Create job**.
4. On the **Specify job details** page, under **Job settings**, specify the following.
  - a. **Name** – the name of the transcription job.

b. **Audio input type – Conversation**

5. For the remaining fields, specify the Amazon S3 location of your audio file and where you want to store the output of your transcription job.
6. Choose **Next**.
7. Choose **Create**.

## API

### To transcribe a medical conversation using a batch transcription job (API)

- For the [StartMedicalTranscriptionJob](#) API, specify the following.
  - a. For `MedicalTranscriptionJobName`, specify a name unique in your AWS account.
  - b. For `LanguageCode`, specify the language code that corresponds to the language spoken in your audio file and the language of your vocabulary filter.
  - c. For the `MediaFileUri` parameter of the `Media` object, specify the name of the audio file that you want to transcribe.
  - d. For `Specialty`, specify the medical specialty of the clinician speaking in the audio file as `PRIMARYCARE`.
  - e. For `Type`, specify `CONVERSATION`.
  - f. For `OutputBucketName`, specify the Amazon S3 bucket to store the transcription results.

The following is an example request that uses the AWS SDK for Python (Boto3) to transcribe a medical conversation of a clinician in the `PRIMARYCARE` specialty and a patient.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-med-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    OutputBucketName = 's3://DOC-EXAMPLE-BUCKET'
)
while True:
    status =
        transcribe.get_medical_transcription_job(MedicalTranscriptionJobName=job_name)
        if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
        'FAILED']:
            break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

The following example code shows the transcription results of a clinician-patient conversation.

```
{
```

```
"jobName": "conversation-medical-transcription-job",
"accountId": "123456789012",
"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

### To transcribe a medical conversation using a batch transcription job (AWS CLI)

- Run the following code.

```
aws transcribe start-medical-transcription-job \
--cli-input-json file://example-start-command.json
```

The following code shows the contents of `example-start-command.json`.

```
{
    "MedicalTranscriptionJobName": "my-first-med-transcription-job",
    "LanguageCode": "en-US",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName": "s3://DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
    }
}
```

The following is the response from running the preceding AWS CLI command.

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "my-first-med-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "en-US",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
        },
        "StartTime": "2020-10-05T20:43:39.583000+00:00",
        "CreationTime": "2020-10-05T20:43:39.547000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

## Transcribing a medical conversation in a real-time stream

You can transcribe an audio stream of a medical conversation using either the HTTP/2 or [WebSocket](#) protocols. For information on how to start a stream using the WebSocket protocol, see [Establish a bi-directional connection using the WebSocket protocol \(p. 267\)](#). To start an HTTP/2 stream, use the [StartMedicalStreamTranscription](#) API.

You can transcribe streaming audio in the following medical specialties:

- Cardiology
- Neurology
- Oncology
- Primary Care
- Urology

Each medical specialty includes many types of procedures and appointments. Clinicians therefore dictate many different types of notes. Use the following examples as guidance to help you specify the value of the `specialty` URL parameter of the WebSocket request, or the `Specialty` parameter of the [StartMedicalStreamTranscription](#) API:

- For electrophysiology or echocardiography consultations, choose `CARDIOLOGY`.
- For medical oncology, surgical oncology, or radiation oncology consultations, choose `ONCOLOGY`.
- For a physician providing a consultation to a patient who had a stroke, either a transient ischemic attack or a cerebrovascular attack, choose `NEUROLOGY`.
- For a consultation around urinary incontinence, choose `UROLOGY`.
- For yearly checkup or urgent care visits, choose `PRIMARYCARE`.
- For inpatient hospitalist visits, choose `PRIMARYCARE`.
- For consultations regarding fertility, tubal ligation, IUD insertion, or abortion, choose `PRIMARYCARE`.

### AWS Management Console

#### To transcribe a streaming medical conversation (AWS Management Console)

To use the AWS Management Console to transcribe a clinician-patient dialogue in real-time stream, choose the option to transcribe a medical conversation, start the stream, and begin speaking into the microphone.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Real-time transcription**.
3. Choose **Conversation**.
4. For **Medical specialty**, choose the clinician's specialty.
5. Choose **Start streaming**.
6. Speak into the microphone.

### Transcribing a medical conversation in an HTTP/2 stream

The following is the syntax for the parameters of an HTTP/2 request.

To transcribe an HTTP/2 stream of a medical conversation, use the [StartMedicalStreamTranscription](#) API and specify the following:

- **LanguageCode** – The language code. The valid value is `en-US`
- **MediaEncoding** – The encoding used for the input audio. Valid values are `pcm`, `ogg-opus`, and `flac`.
- **Specialty** – The specialty of the medical professional.
- **Type** – `CONVERSATION`

To improve transcription accuracy of specific terms in a real-time stream, use a custom vocabulary. To enable a custom vocabulary, set the value of `VocabularyName` parameter to the name of the custom vocabulary that you want to use. For more information, see [Improving transcription accuracy with medical custom vocabularies \(p. 299\)](#).

To label the speech from different speakers, set the `ShowSpeakerLabel` parameter to `true`. For more information, see [Identifying speakers and labeling their speech \(p. 280\)](#).

For more information on setting up an HTTP/2 stream to transcribe a medical conversation, see [Streaming request \(p. 255\)](#).

## Transcribing a medical conversation in a WebSocket stream

You can use a WebSocket request to transcribe a medical conversation. When you make a WebSocket request, you create a pre-signed URI. This URI contains the information needed to set up the audio stream between your application and Amazon Transcribe Medical. For more information on creating WebSocket requests, see [Establish a bi-directional connection using the WebSocket protocol \(p. 267\)](#).

Use the following template to create your pre-signed URI.

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/medical-stream-transcription-  
websocket  
?language-code=languageCode  
&X-Amz-Algorithm=AWS4-HMAC-SHA256  
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-west-2%2Ftranscribe%2Faws4_request  
&X-Amz-Date=20220208T235959Z  
&X-Amz-Expires=250  
&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
```

To improve transcription accuracy of specific terms in a real-time stream, use a custom vocabulary. To enable a custom vocabulary, set the value of `vocabulary-name` to the name of the custom vocabulary that you want to use. For more information, see [Improving transcription accuracy with medical custom vocabularies \(p. 299\)](#).

To label the speech from different speakers, set the `show-speaker-label` parameter in to `true`. For more information, see [Identifying speakers and labeling their speech \(p. 280\)](#).

For more information on creating pre-signed URIs, see [Creating a pre-signed URI \(p. 268\)](#).

## Identifying speakers and labeling their speech

To identify and label the speech from different speakers in Amazon Transcribe Medical, use *speaker diarization*. This enables you to see what the patient said and what the clinician said in the transcription output.

When you enable speaker diarization, Amazon Transcribe Medical labels each speaker *utterance* with a unique identifier for each speaker. An *utterance* is a unit of speech that is typically separated from other utterances by silence. In batch transcription, an utterance from the clinician could receive a label of `spk_0` and an utterance from the patient could receive a label of `spk_1`.

If an utterance from one speaker overlaps with an utterance from another speaker, Amazon Transcribe Medical orders them in the transcription by their start times. Utterances that overlap in the input audio don't overlap in the transcription output.

You can enable speaker diarization when you transcribe an audio file using batch transcription job, or in a real-time stream.

### Topics

- [Identifying speakers and labeling their speech in audio files \(p. 281\)](#)
- [Identifying speakers and labeling their speech in real-time streams \(p. 284\)](#)

## Identifying speakers and labeling their speech in audio files

You can enable speaker identification in a batch transcription job using either the [StartMedicalTranscriptionJob](#) API or the AWS Management Console. This enables you to identify the speakers in a clinician-patient conversation and determine who said what in the transcription output.

### AWS Management Console

To use the AWS Management Console to enable speaker diarization in your transcription job, you enable audio identification and then speaker identification.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Transcription jobs**.
3. Choose **Create job**.
4. On the **Specify job details** page, provide information about your transcription job.
5. Choose **Next**.
6. Enable **Audio identification**.
7. For **Audio identification type**, choose **Speaker identification**.
8. For **Maximum number of speakers**, enter the maximum number of speakers that you think are speaking in your audio file. For best results, match the number of speakers that you ask to identify to the number of speakers in the input audio. If you specify a value less than the number of speakers in your input audio, the transcription text of the most similar sounding speakers are attributed to a speaker label.
9. Choose **Create**.

### API

#### To identify speakers in an audio file using a batch transcription job (API)

- For the [StartMedicalTranscriptionJob](#) API, specify the following.
  - a. For `MedicalTranscriptionJobName`, specify a name that is unique in your AWS account.
  - b. For `LanguageCode`, specify the language code that corresponds to the language spoken in the audio file.
  - c. For the `MediaFileUri` parameter of the `Media` object, specify the name of the audio file that you want to transcribe.
  - d. For `Specialty`, specify the medical specialty of the clinician speaking in the audio file.
  - e. For `Type`, specify `CONVERSATION`.

- f. For `OutputBucketName`, specify the Amazon S3 bucket to store the transcription results.
- g. For the `Settings` object, specify the following.
  - i. `ShowSpeakerLabels` – true.
  - ii. `MaxSpeakerLabels` – An integer between 2 and 10 to indicate the number of speakers that you think are speaking in your audio. For best results, match the number of speakers that you ask to identify to the number of speakers in the input audio. If you specify a value less than the number of speakers in your input audio, the transcription text of the most similar sounding speakers are attributed to the same speaker label.

The following request uses the AWS SDK for Python (Boto3) to start a batch transcription job of a primary care clinician patient dialogue with speaker identification enabled.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET.s3-us-west-2.amazonaws.com/my-audio-file.flac"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName=job_name,
    Media = {'MediaUri': job_uri},
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    OutputBucketName = 's3://DOC-EXAMPLE-BUCKET',
    Settings = {'ShowSpeakerLabels': True,
                'MaxSpeakerLabels': 2
               }
)
while True:
    status = transcribe.get_medical_transcription_job(MedicalTranscriptionJobName=job_name)
    if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
    'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)
```

The following example code shows the transcription results of a transcription job with speaker identification enabled.

```
{
  "jobName": "job ID",
  "accountId": "account ID",
  "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

**To transcribe an audio file of a conversation between a clinician practicing primary care and a patient in an audio file and identify what each person said in the transcription output (AWS CLI)**

- Run the following code.

```
aws transcribe start-transcription-job \
--cli-input-json file://example-start-command.json
```

The following code shows the contents of `example-start-command.json`.

```
{
    "MedicalTranscriptionJobName": "my-first-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName": "s3://DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
    },
    "Settings": {
        "ShowSpeakerLabels": true,
        "MaxSpeakerLabels": 2
    }
}
```

The following is the response from running the preceding AWS CLI command.

```
{
    "MedicalTranscriptionJobName": "my-first-transcription-job",
    "LanguageCode": "en-US",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName": "s3://DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
    },
    "Settings": {
        "ShowSpeakerLabels": true,
        "MaxSpeakerLabels": 2
    }
}
```

## Identifying speakers and labeling their speech in real-time streams

To identify speakers and label their speech in a real-time stream, use the AWS Management Console or a streaming request. Speaker identification works best for identifying between two and five speakers in a stream. Although Amazon Transcribe Medical can identify more than five speakers in a stream, the accuracy of speaker identification decreases if you exceed that number.

To start an HTTP/2 request, use the [StartMedicalStreamTranscription](#) API. To start a WebSocket request, use a pre-signed URI. The URI contains the information required to set up bi-directional communication between your application and Amazon Transcribe Medical.

### Identifying speakers in audio that is spoken into your microphone (AWS Management Console)

You can use the AWS Management Console to start a real-time stream of a clinician-patient conversation, or a dictation that is spoken into your microphone in real-time.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, for Amazon Transcribe Medical choose **Real-time transcription**.
3. For **Audio input type**, choose the type of medical speech that you want to transcribe.
4. For **Additional settings**, choose **Speaker identification**.
5. Choose **Start streaming** to start transcribing your real-time audio.
6. Speak into the microphone.

### Identifying speakers in an HTTP/2 stream

To identify speakers in an HTTP/2 stream of a medical conversation, use the [StartMedicalStreamTranscription](#) API and specify the following:

- For **LanguageCode**, specify the language code that corresponds to the language in the stream. The valid value is `en-US`.
- For **MediaSampleHertz**, specify the sample rate of the audio.
- For **Specialty**, specify the medical specialty of the provider.
- **ShowSpeakerLabel** – `true`

For more information on setting up an HTTP/2 stream to transcribe a medical conversation, see [Streaming request \(p. 255\)](#).

### Identifying speakers in a WebSocket request

To identify speakers in WebSocket streams with the API, use the following format to create a pre-signed URI to start a WebSocket request and set `show-speaker-label` to `true`.

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/medical-stream-transcription-
websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=250
&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
```

The following code shows the truncated example response of a streaming request.

```
{
  "Transcript": {
    "Results": [
      {
        "Alternatives": [
          {
            "Items": [
              {
                "Content": "Hello, how are you today?",
                "Speaker": "Dr. Smith"
              }
            ],
            "Offset": 0.0,
            "TranscriptType": "Text"
          }
        ],
        "Offset": 0.0,
        "TranscriptType": "Text"
      }
    ],
    "Offset": 0.0,
    "TranscriptType": "Text"
  }
}
```

```
        "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": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX",
"StartTime": 18.74
}
]
}
```

Amazon Transcribe Medical breaks your incoming audio stream based on natural speech segments, such as a change in speaker or a pause in the audio. The transcription is returned progressively to your application, with each response containing more transcribed speech until the entire segment is transcribed. The preceding code is a truncated example of a fully-transcribed speech segment. Speaker labels only appear for entirely transcribed segments.

The following list shows the organization of the objects and parameters in a streaming transcription output.

#### **Transcript**

Each speech segment has its own `Transcript` object.

#### **Results**

Each `Transcript` object has its own `Results` object. This object contains the `isPartial` field. When its value is `false`, the results returned are for an entire speech segment.

#### **Alternatives**

Each `Results` object has an `Alternatives` object.

#### **Items**

Each `Alternatives` object has its own `Items` object that contains information about each word and punctuation mark in the transcription output. When you enable speaker identification, each word has a `Speaker` label for fully-transcribed speech segments. Amazon Transcribe Medical uses this label to assign a unique integer to each speaker it identifies in the stream. The `Type` parameter having a value of `speaker-change` indicates that one person has stopped speaking and that another person is about to begin.

#### **Transcript**

Each `Items` object contains a transcribed speech segment as the value of the `Transcript` field.

For more information about WebSocket requests, see [Creating a pre-signed URI \(p. 268\)](#).

## Transcribing multi-channel audio

If you have an audio file or stream that has multiple channels, you can use *channel identification* to transcribe the speech from each of those channels. Amazon Transcribe Medical transcribes the speech from each channel separately. It combines the separate transcriptions of each channel into a single transcription output.

Use channel identification to identify the separate channels in your audio and transcribe the speech from each of those channels. Enable this in situations such as a caller and agent scenario. Use this to

distinguish a caller from an agent in recordings or streams from contact centers that perform drug safety monitoring.

You can enable channel identification for both batch processing and real-time streaming. The following list describes how to enable it for each method.

- Batch transcription – AWS Management Console and [StartMedicalTranscriptionJob](#) API
- Streaming transcription – WebSocket streaming and [StartMedicalStreamTranscription](#) API

## Transcribing multi-channel audio files

When you transcribe an audio file, Amazon Transcribe Medical returns a list of *items* for each channel. An item is a transcribed word or punctuation mark. Each word has a start time and an end time. If a person on one channel speaks over a person on a separate channel, the start times and end times of the items for each channel overlap while the individuals are speaking over each other.

By default, you can transcribe audio files with two channels. You can request a quota increase if you need to transcribe files that have more than two channels. For information about requesting a quota increase, see [AWS service quotas](#).

To transcribe multi-channel audio in a batch transcription job, use the AWS Management Console or the [StartMedicalTranscriptionJob](#) API.

### AWS Management Console

To use the AWS Management Console to enable channel identification in your batch transcription job, you enable audio identification and then channel identification. Channel identification is a subset of audio identification in the AWS Management Console.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Transcription jobs**.
3. Choose **Create job**.
4. On the **Specify job details** page, provide information about your transcription job.
5. Choose **Next**.
6. Enable **Audio identification**.
7. For **Audio identification type**, choose **Channel identification**.
8. Choose **Create**.

### API

#### To transcribe a multi-channel audio file (API)

- For the [StartMedicalTranscriptionJob](#) API, specify the following.
  - a. For **TranscriptionJobName**, specify a name unique to your AWS account.
  - b. For **LanguageCode**, specify the language code that corresponds to the language spoken in the audio file. The valid value is **en-US**.
  - c. For the **MediaFileUri** parameter of the **Media** object, specify the name of the media file that you want to transcribe.
  - d. For the **Settings** object, set **ChannelIdentification** to **true**.

The following is an example request using the AWS SDK for Python (Boto3).

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName=job_name,
    Media = {'MediaFileUri': job_uri},
    MediaFormat = 'flac',
    LanguageCode = 'en-US',
    OutputBucketName = 's3://DOC-EXAMPLE-BUCKET',
    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

### To transcribe a multi-channel audio file using a batch transcription job (AWS CLI)

- Run the following code.

```
aws transcribe start-medical-transcription-job \
--cli-input-json file://example-start-command.json
```

The following is the code of `example-start-command.json`.

```
{
    "MedicalTranscriptionJobName": "my-first-med-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName": "s3://DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
    },
    "Settings": {
        "ChannelIdentification": true
    }
}
```

The following is the response.

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "my-first-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
        },
        "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"
    }
}
```

The following code shows the transcription output for an audio file that has a conversation on two channels.

```
{
    "jobName": "job id",
    "accountId": "account id",
    "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"
        },
        ...
    ]
}
```

## Transcribing multi-channel audio streams

You can transcribe audio from separate channels in either HTTP/2 or WebSocket streams using the [StartMedicalStreamTranscription](#) API.

By default, you can transcribe streams with two channels. You can request a quota increase if you need to transcribe streams that have more than two channels. For information about requesting a quota increase, see [AWS service quotas](#).

## Transcribing multi-channel audio in an HTTP/2 stream

To transcribe multi-channel audio in an HTTP/2 stream, use the [StartMedicalStreamTranscription API](#) and specify the following:

- `LanguageCode` – The language code of the audio. The valid value is `en-US`.
- `MediaEncoding` – The encoding of the audio. Valid values are `ogg-opus`, `flac`, and `pcm`.
- `EnableChannelIdentification` – `true`
- `NumberOfChannels` – the number of channels in your streaming audio.

For more information on setting up an HTTP/2 stream to transcribe a medical conversation, see [Streaming request \(p. 255\)](#).

## Transcribing multi-channel audio in a WebSocket stream

To identify speakers in WebSocket streams, use the following format to create a pre-signed URI and start a WebSocket request. Specify `enable-channel-identification` as `true` and the number of channels in your stream in `number-of-channels`. A pre-signed URI contains the information needed to set up bi-directional communication between your application and Amazon Transcribe Medical.

```
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=250
&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=number of channels in your audio stream
```

Parameter definitions can be found in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

For more information about WebSocket requests, see [Creating a pre-signed URI \(p. 268\)](#).

## Multi-channel streaming output

The output of a streaming transcription is the same for HTTP/2 and WebSocket requests. The following is an example output.

```
{
    "resultId": "xxxxxxxx-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"
}
```

For each speech segment, there is a `channelId` flag that indicates which channel the speech belongs to.

## Transcribing a medical dictation

You can use Amazon Transcribe Medical to transcribe clinician-dictated medical notes using either a batch transcription job or a real-time stream. Batch transcription jobs enable you to transcribe audio files. You specify the medical specialty of the clinician in your transcription job or stream to ensure that Amazon Transcribe Medical produces transcription results with the highest possible accuracy.

You can transcribe a medical dictation in the following specialties:

- Cardiology – available in streaming transcription only
- Neurology – available in streaming transcription only
- Oncology – available in streaming transcription only
- Primary Care – includes the following types of medical practice:
  - Family medicine
  - Internal medicine
  - Obstetrics and Gynecology (OB-GYN)
  - Pediatrics
- Radiology – available in streaming transcription only
- Urology – available in streaming transcription only

You can improve transcription accuracy by using custom vocabularies. For information on how medical custom vocabularies work, see [Improving transcription accuracy with medical custom vocabularies \(p. 299\)](#).

By default, Amazon Transcribe Medical returns the transcription with the highest confidence level. If you'd like to configure it to return alternative transcriptions, see [Generating alternative transcriptions \(p. 315\)](#).

For information about how numbers and medical measurements appear in the transcription output, see [Transcribing numbers \(p. 244\)](#) and [Transcribing medical terms and measurements \(p. 245\)](#).

### Topics

- [Transcribing an audio file of a medical dictation \(p. 294\)](#)

- [Transcribing a medical dictation in a real-time stream \(p. 297\)](#)

## Transcribing an audio file of a medical dictation

Use a batch transcription job to transcribe audio files of medical conversations. You can use this to transcribe a clinician-patient dialogue. You can start a batch transcription job in either the [StartMedicalTranscriptionJob](#) API or the AWS Management Console.

When you start a medical transcription job with the [StartMedicalTranscriptionJob](#) API, you specify **PRIMARYCARE** as the value of the **Specialty** parameter.

### AWS Management Console

#### To transcribe a clinician-patient dialogue (AWS Management Console)

To use the AWS Management Console to transcribe a clinician-patient dialogue, create a transcription job and choose **Conversation** for **Audio input type**.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Transcription jobs**.
3. Choose **Create job**.
4. On the **Specify job details** page, under **Job settings**, specify the following.
  - a. **Name** – the name of the transcription job.
  - b. **Audio input type – Dictation**
5. For the remaining fields, specify the Amazon S3 location of your audio file and where you want to store the output of your transcription job.
6. Choose **Next**.
7. Choose **Create**.

### API

#### To transcribe a medical conversation using a batch transcription job (API)

- For the [StartMedicalTranscriptionJob](#) API, specify the following.
  - a. For **MedicalTranscriptionJobName**, specify a name unique in your AWS account.
  - b. For **LanguageCode**, specify the language code that corresponds to the language spoken in your audio file and the language of your vocabulary filter.
  - c. In the **MediaFileUri** parameter of the **Media** object, specify the name of the audio file that you want to transcribe.
  - d. For **Specialty**, specify the medical specialty of the clinician speaking in the audio file.
  - e. For **Type**, specify **DICTATION**.
  - f. For **OutputBucketName**, specify the Amazon S3 bucket to store the transcription results.

The following is an example request that uses the AWS SDK for Python (Boto3) to transcribe a medical dictation of a clinician in the **PRIMARYCARE** specialty.

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

```

transcribe = boto3.client('transcribe')
job_name = "my-first-med-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'DICTATION',
    OutputBucketName = 's3://DOC-EXAMPLE-BUCKET'
)
while True:
    status =
    transcribe.get_medical_transcription_job(MedicalTranscriptionJobName=job_name)
    if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
    'FAILED']:
        break
    print("Not ready yet...")
    time.sleep(5)
print(status)

```

The following example code shows the transcription results of a medical dictation.

```
{
    "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

### To identify the speakers in an audio file using a batch transcription job (AWS CLI)

- Run the following code.

```
aws transcribe start-medical-transcription-job \
--cli-input-json file://filepath/example-start-command.json
```

The following code shows the contents of `example-start-command.json`.

```
{
    "MedicalTranscriptionJobName": "my-first-med-transcription-job",
    "LanguageCode": "en-US",
    "Specialty": "PRIMARYCARE",
```

```
"Type": "DICTATION",
"OutputBucketName": "s3://DOC-EXAMPLE-BUCKET",
"Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
}
```

The following is the response from running the preceding AWS CLI command.

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "my-first-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "en-US",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
        },
        "StartTime": "2020-09-20T00:35:22.256000+00:00",
        "CreationTime": "2020-09-20T00:35:22.218000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

## Transcribing a medical dictation in a real-time stream

Use a WebSocket stream to transcribe a medical dictation as an audio stream. You can also use the AWS Management Console to transcribe speech that you or others speak directly into a microphone.

For an HTTP/2 or a WebSocket stream, you can transcribe audio in the following medical specialties:

- Cardiology
- Oncology
- Neurology
- Primary Care
- Radiology
- Urology

Each medical specialty includes many types of procedures and appointments. Clinicians therefore dictate many different types of notes. Use the following examples as guidance to help you specify the value of the `specialty` URI parameter of the WebSocket request, or the `Specialty` parameter of the `StartMedicalStreamTranscription` API:

- For a dictation after electrophysiology or echocardiogram procedure, choose `CARDIOLOGY`.
- For a dictation after a surgical oncology or radiation oncology procedure, choose `ONCOLOGY`.
- For a physician dictating notes indicating a diagnosis of encephalitis, choose `NEUROLOGY`.
- For a dictation of procedure notes to break up a bladder stone, choose `UROLOGY`.
- For a dictation of clinician notes after an internal medicine consultation, choose `PRIMARYCARE`.

- For a dictation of a physician communicating the findings of a CT scan, PET scan, MRI, or radiograph, choose **RADIOLOGY**.
- For a dictation of physician notes after a gynecology consultation, choose **PRIMARYCARE**.

To improve transcription accuracy of specific terms in a real-time stream, use a custom vocabulary. To enable a custom vocabulary, set the value of **vocabulary-name** to the name of the custom vocabulary you want to use.

## Transcribing a dictation spoken into your microphone with the AWS Management Console

To use the AWS Management Console to transcribe streaming audio of a medical dictation, choose the option to transcribe a medical dictation, start the stream, and begin speaking into the microphone.

### To transcribe streaming audio of a medical dictation (AWS Management Console)

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Real-time transcription**.
3. Choose **Dictation**.
4. For **Medical specialty**, choose the medical specialty of the clinician speaking in the stream.
5. Choose **Start streaming**.
6. Speak into the microphone.

## Transcribing a dictation in an HTTP/2 stream

To transcribe an HTTP/2 stream of a medical dictation, use the [StartMedicalStreamTranscription](#) API and specify the following:

- **LanguageCode** – The language code. The valid value is `en-US`
- **MediaEncoding** – The encoding used for the input audio. Valid values are `pcm`, `ogg-opus`, and `flac`.
- **Specialty** – The specialty of the medical professional.
- **Type** – `DICTATION`

For more information on setting up an HTTP/2 stream to transcribe a medical dictation, see [Streaming request \(p. 255\)](#).

## Using a WebSocket streaming request to transcribe a medical dictation

To transcribe a medical dictation in a real-time stream using a WebSocket request, you create a pre-signed URI. This URI contains the information needed to set up the audio stream between your application and Amazon Transcribe Medical. For more information on creating WebSocket requests, see [Establish a bi-directional connection using the WebSocket protocol \(p. 267\)](#).

Use the following template to create your pre-signed URI.

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

```
&X-Amz-Expires=250
&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
```

For more information on creating pre-signed URIs, see [Creating a pre-signed URI \(p. 268\)](#).

## Improving transcription accuracy with medical custom vocabularies

To improve transcription accuracy in Amazon Transcribe Medical, create and use one or more medical custom vocabularies. A *custom vocabulary* is a collection of words or phrases that are domain-specific. This collection helps improve the performance of Amazon Transcribe Medical in transcribing those words or phrases.

You are responsible for the integrity of your own data when you use Amazon Transcribe Medical. Do not enter confidential information, personal information (PII), or protected health information (PHI), into a custom vocabulary.

For best results, create separate small custom vocabularies that each help transcribe a specific audio recording. You receive greater improvements in transcription accuracy than if you created one large custom vocabulary to use with all of your recordings.

By default, you can have up to 100 custom vocabularies in your AWS account. A custom vocabulary can't exceed 50 KB in size. For information on requesting an increase to the number of custom vocabularies that you can have in your account, see [AWS service quotas](#).

Custom vocabularies are available in US English (en-US).

### Topics

- [Creating a text file for your medical custom vocabulary \(p. 299\)](#)
- [Using a text file to create a medical custom vocabulary \(p. 302\)](#)
- [Transcribing an audio file using a medical custom vocabulary \(p. 304\)](#)
- [Transcribing a real-time stream using a medical custom vocabulary \(p. 305\)](#)
- [Character sets for Amazon Transcribe Medical \(p. 307\)](#)

## Creating a text file for your medical custom vocabulary

To create a custom vocabulary, you create a text file that is in UTF-8 format. In this file, you create a four column table, with each column specifying a field. Each field tells Amazon Transcribe Medical either how the domain-specific terms are pronounced or how to display these terms in your transcriptions. You store the text file containing these fields in an Amazon S3 bucket.

## Understanding how to format your text file

To create a medical custom vocabulary, you enter the column names as a header row. You enter the values for each column beneath the header row.

The following are the names of the four columns of the table:

- **Phrase** – column required, values required
- **IPA** – column required, values can be optional
- **SoundsLike** – column required, values can be optional
- **DisplayAs** – column required, values can be optional

When you create a custom vocabulary, make sure that you:

- Separate each column with a single Tab character. Amazon Transcribe throws an error message if you try to separate the columns with spaces or multiple Tab characters.
- Make sure that there's no trailing spaces or white space after each value within a column.

Make sure that the values that you enter for each column:

- Have fewer than 256 characters, including hyphens
- Use only characters from the allowed character set, see [Character sets for Amazon Transcribe Medical \(p. 307\)](#).

## Entering values for the columns of the table

The following information shows you how to specify values for the four columns of the table:

- **Phrase** – The word or phrase that should be recognized. You must enter values in this column.

If the entry is a phrase, separate the words with a hyphen (-). For example, enter **cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy as cerebral-autosomal-dominant-arteriopathy-with-subcortical-infarcts-and-leukoencephalopathy**.

Enter acronyms or other words whose letters should be pronounced individually as single letters followed by dots, such as **D.N.A.** or **S.T.E.M.I.**. To enter the plural form of an acronym, such as "STEMIs," separate the "s" from the acronym with a hyphen: "**S.T.E.M.I-s**" You can use either uppercase or lowercase letters for acronyms.

The **Phrase** column is required. You can use any of the allowed characters for the input language. For allowed characters, see [Character sets for Amazon Transcribe Medical \(p. 307\)](#). If you don't specify the **DisplayAs** column, Amazon Transcribe Medical uses the contents of the **Phrase** column in the output file.

- **IPA** (column required, values can be optional) – To specify the pronunciation of a word or phrase, you can include characters in the [International Phonetic Alphabet \(IPA\)](#) in this column. The IPA column can't contain leading or trailing spaces, and you must use a single space to separate each phoneme in the input. For example, in English you would enter the phrase **acute-respiratory-distress-syndrome as # k j u t # s p # t # i d # s t # s s # n d # o# m**. You would enter the phrase **A.L.L. as e# # l # 1**.

Even if you don't specify the contents of the **IPA** column, you must include a blank **IPA** column. If you include values in the **IPA** column, you can't provide values for the **SoundsLike** column.

For a list of allowed IPA characters for a specific language, see [Character sets for Amazon Transcribe Medical \(p. 307\)](#). US English is the only language available in Amazon Transcribe Medical.

- **SoundsLike** (column required, values can be optional) – You can break a word or phrase down into smaller segments and provide a pronunciation for each segment using the standard orthography of the language to mimic the way that the word sounds. For example, you can provide pronunciation hints for the phrase **cerebral-autosomal-dominant-arteriopathy-with-subcortical-infarcts-and-leukoencephalopathy** like this: **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**. The hint for the phrase **atrioventricular-nodal-reentrant-tachycardia** would look like this: **ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia**. You separate each part of the hint with a hyphen (-).

Even if you don't provide values for the SoundsLike column, you must include a blank SoundsLike column. If you include values in the SoundsLike column, you can't provide values for the IPA column.

You can use any of the allowed characters for the input language. For the list of allowed characters, see [Character sets for Amazon Transcribe Medical \(p. 307\)](#).

- **DisplayAs** (column required, values can be optional) – Defines how the word or phrase looks when it's output. For example, if the word or phrase is **cerebral-autosomal-dominant-arteriopathy-with-subcortical-infarcts-and-leukoencephalopathy**, you can specify the display form as cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy, so that the hyphen is not present. You can also specify DisplayAs as CADASIL if you'd like to show the acronym instead of the full term in the output.

If you don't specify the DisplayAs column, Amazon Transcribe Medical uses the Phrase column from the input file in the output.

You can use any UTF-8 character in the DisplayAs column.

You can include spaces only for the values in the IPA and DisplayAs columns.

To create the text file of your custom vocabulary, place each word or phrase in your text file on a separate line. Separate the columns with Tab characters. Include spaces only for values in the IPA and DisplayAs columns. Save the file with the extension .txt in an Amazon S3 bucket in the same AWS Region where you use Amazon Transcribe Medical to create your custom vocabulary.

If you edit your text file in Windows, make sure that your file is in LF format and not in CRLF format. Otherwise, you will be unable to create your custom vocabulary. Some text editors enable you to change the formatting with Find and Replace commands.

The following examples show text that you can use to create custom vocabularies. To create a custom vocabulary from these examples, copy an example into a text editor, replace [TAB] with a Tab character, and upload the saved text file to Amazon S3.

```
Phrase[TAB]IPA[TAB]SoundsLike[TAB]DisplayAs
acute-respiratory-distress-syndrome[TAB][TAB][TAB]acute respiratory distress syndrome
A.L.L.[TAB]e# # 1 # 1[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]
```

You can enter columns in any order. The following examples show other valid structures for the custom vocabulary input file.

```
Phrase[TAB]SoundsLike[TAB]IPA[TAB]DisplayAs
acute-respiratory-distress-syndrome[TAB][TAB][TAB]acute respiratory distress syndrome
A.L.L.[TAB][TAB]e# # 1 # 1[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# # 1 # 1[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
```

For reading ease, the following tables show the preceding examples more clearly in html format. They are meant only to illustrate the examples.

Phrase	IPA	SoundsLike	DisplayAs
acute-respiratory- distress-syndrome			acute respiratory distress syndrome
A.L.L.	eɪ ε l ε l		ALL
atrioventricular-nodal- reentrant-tachycardia		ay-tree-o-ven-trick-u- lar-node-al-re-entr-ant- tack-ih-card-ia	

Phrase	SoundsLike	IPA	DisplayAs
acute-respiratory- distress-syndrome			acute respiratory distress syndrome
atrioventricular-nodal- reentrant-tachycardia	ay-tree-o-ven-trick-u- lar-node-al-re-entr-ant- tack-ih-card-ia		
A.L.L.		eɪ ε l ε l	ALL

DisplayAs	SoundsLike	IPA	Phrase
acute respiratory distress syndrome			acute-respiratory- distress-syndrome
ALL		eɪ ε l ε l	A.L.L.
	ay-tree-o-ven-trick-u- lar-node-al-re-entr-ant- tack-ih-card-ia		atrioventricular-nodal- reentrant-tachycardia

## Using a text file to create a medical custom vocabulary

To create a custom vocabulary, you must have prepared a text file that contains a collection of words or phrases. Amazon Transcribe Medical uses this text file to create a custom vocabulary that you can use to

improve the transcription accuracy of those words or phrases. You can create a custom vocabulary using the [CreateMedicalVocabulary](#) API or the Amazon Transcribe Medical console.

## AWS Management Console

To use the AWS Management Console to create a custom vocabulary, you provide the Amazon S3 URI of the text file containing your words or phrases.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Custom vocabulary**.
3. For **Name**, under **Vocabulary settings**, choose a name for your custom vocabulary.
4. Specify the location of your audio file or video file in Amazon S3:
  - For **Vocabulary input file location on S3** under **Vocabulary settings**, specify the Amazon S3 URL that identifies the text file you will use to create your custom vocabulary.
  - For **Vocabulary input file location in S3**, choose **Browse S3** to browse for the text file and choose it.
5. Choose **Create vocabulary**.

You can see the processing status of your custom vocabulary in the AWS Management Console.

## API

### To create a medical custom vocabulary (API)

- For the [StartTranscriptionJob](#) API, specify the following.
  - a. For **LanguageCode**, specify **en-US**.
  - b. For **VocabularyFileUri**, specify the Amazon S3 location of the text file that you use to define your custom vocabulary.
  - c. For **VocabularyName**, specify a name for your custom vocabulary. The name you specify must be unique within your AWS account.

To see the processing status of your custom vocabulary, use the [GetMedicalVocabulary](#) API.

The following is an example request using the AWS SDK for Python (Boto3) to create a custom vocabulary.

```
from __future__ import print_function
import time
import boto3

transcribe = boto3.client('transcribe')
vocab_name = "example-med-custom-vocab"
text_file_uri = "https://DOC-EXAMPLE-BUCKET.s3-us-west-2.amazonaws.com/my-first-custom-
vocabulary.txt"
transcribe.create_medical_vocabulary(
    VocabularyName = vocab_name,
    VocabularyFileUri = text_file_uri,
    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

### To identify the speakers in an audio file using a batch transcription job (AWS CLI)

- Run the following code.

```
aws transcribe create-medical-vocabulary \
--vocabulary-name example-med-custom-vocab \
--language-code en-US \
--vocabulary-file-uri s3://DOC-EXAMPLE-BUCKET.us-west-2.amazonaws.com/my-first-
custom-vocabulary.txt
```

The following is the response from running the preceding AWS CLI command.

```
{
    "VocabularyName": "cli-medical-vocab-1",
    "LanguageCode": "en-US",
    "VocabularyState": "PENDING"
}
```

## Transcribing an audio file using a medical custom vocabulary

Use the [StartMedicalTranscriptionJob](#) or the AWS Management Console to start a transcription job that uses a custom vocabulary to improve transcription accuracy.

### AWS Management Console

- Sign in to the [AWS Management Console](#).
- In the navigation pane, under Amazon Transcribe Medical, choose **Transcription jobs**.
- Choose **Create job**.
- On the **Specify job details** page, provide information about your transcription job.
- Choose **Next**.
- Under **Customization**, enable **Custom vocabulary**.
- Under **Vocabulary selection**, choose a custom vocabulary.
- Choose **Create**.

### API

### To identify speakers in an audio file using a batch transcription job (API)

- For the [StartMedicalTranscriptionJob](#) API, specify the following.

- a. For `MedicalTranscriptionJobName`, specify a name that is unique in your AWS account.
- b. For `LanguageCode`, specify the language code that corresponds to the language spoken in your audio file and the language of your vocabulary filter.
- c. For the `MediaFileUri` parameter of the `Media` object, specify the name of the audio file that you want to transcribe.
- d. For `Specialty`, specify the medical specialty of the clinician speaking in the audio file.
- e. For `Type`, specify whether the audio file is a conversation or a dictation.
- f. For `OutputBucketName`, specify the Amazon S3 bucket to store the transcription results.
- g. For the `Settings` object, specify the following.
  - `VocabularyName` – the name of your custom vocabulary.

The following request uses the AWS SDK for Python (Boto3) to start a batch transcription job with a custom vocabulary.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-med-transcription-job"
job_uri = "https://DOC-EXAMPLE-BUCKET.s3-us-west-2.amazonaws.com/my-audio-file.flac"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName=job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    OutputBucketName = 's3://DOC-EXAMPLE-BUCKET',
    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)
```

## Transcribing a real-time stream using a medical custom vocabulary

To improve transcription accuracy in a real-time stream, you can use a custom vocabulary using either HTTP/2 or WebSocket streams. To start an HTTP/2 request, use the [StartMedicalStreamTranscription](#) API. You can use a custom vocabulary in real-time using either the AWS Management Console, the [StartMedicalStreamTranscription](#) API, or by using the WebSocket protocol.

## Transcribing a dictation that is spoken into your Microphone (AWS Management Console)

To use the AWS Management Console to transcribe streaming audio of a medical dictation, choose the option to transcribe a medical dictation, start the stream, and begin speaking into the microphone.

### To transcribe streaming audio of a medical dictation (AWS Management Console)

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Real-time transcription**.
3. For **Medical specialty**, choose the medical specialty of the clinician speaking in the stream.
4. For **Audio input type**, choose either **Conversation** or **Dictation**.
5. For **Additional settings**, choose **Custom vocabulary**.
  - For **Vocabulary selection**, choose the custom vocabulary.
6. Choose **Start streaming**.
7. Speak into the microphone.

## Identifying speakers in an HTTP/2 stream

The following is the syntax for the parameters of an HTTP/2 request.

```
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
```

Parameter descriptions:

- **host:** Update the AWS Region ('us-west-2' in the preceding example) with the AWS Region you are calling. For a list of valid AWS Regions, see [AWS Regions and Endpoints](#).
- **authorization:** This is a generated field. To learn more about creating a signature, see [Signing AWS requests with Signature Version 4](#).
- **x-amz-target:** Don't alter this field; use the content shown in the preceding example.
- **x-amz-content-sha256:** This is a generated field. To learn more about calculating a signature, see [Signing AWS requests with Signature Version 4](#).
- **x-amz-date:** The date and time the signature is created. The format is YYYYMMDDTHHMMSSZ, where YYYY=year, MM=month, DD=day, HH=hour, MM=minute, SS=seconds, and 'T' and 'Z' are fixed characters. For more information, refer to [Handling Dates in Signature Version 4](#).
- **x-amzn-transcribe-session-id:** The name for your streaming session.
- **x-amzn-transcribe-language-code:** The encoding used for your input audio. Refer to [StartStreamTranscription](#) or [Supported languages and language-specific features \(p. 3\)](#) for a list of valid values.

- **x-amzn-transcribe-media-encoding:** The encoding used for your input audio. Valid values are `pcm`, `ogg-opus`, and `flac`.
- **x-amzn-transcribe-sample-rate:** The sample rate of the input audio (in Hertz). Amazon Transcribe supports a range from 8,000 Hz to 48,000 Hz. Low-quality audio, such as telephone audio, is typically around 8,000 Hz. High-quality audio typically ranges from 16,000 Hz to 48,000 Hz. Note that the sample rate you specify **must** match that of your audio.
- **x-amzn-transcribe-vocabulary-name:** The name of the vocabulary you want to use with your transcription.
- **x-amzn-transcribe-specialty:** The medical specialty being transcribed.
- **x-amzn-transcribe-type:** Choose whether this is a dictation or a conversation.
- **x-amzn-transcribe-show-speaker-label:** To enable diarization, this value must be `true`.
- **content-type:** Don't alter this field; use the content shown in the preceding example.

## Identifying speakers in a WebSocket request

To identify speakers in WebSocket streams with the API, use the following format to create a pre-signed URI to start a WebSocket request and set `vocabulary-name` to the name of the custom vocabulary.

```
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=250
&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
```

## Character sets for Amazon Transcribe Medical

To use custom vocabularies in Amazon Transcribe Medical, use the following character sets.

### English character set

For English custom vocabularies, you can use the following characters in the `Phrase` and `SoundsLike` columns:

- a - z
- A - Z
- ' (apostrophe)
- - (hyphen)
- . (period)

You can use the following International Phonetic Alphabet (IPA) characters in the `IPA` column of the vocabulary input file.

<b>Character</b>	<b>Code</b>	<b>Character</b>	<b>Code</b>
au	0061 028A	w	0077
ar	0061 026A	z	007A
b	0062	æ	00E6
d	0064	ð	00F0
er	0065 026A	ŋ	014B
f	0066	ɑ	0251
g	0067	ɔ	0254
h	0068	ɔɪ	0254 026A
i	0069	ə	0259
j	006A	ɛ	025B
k	006B	ɜ	025D
l	006C	g	0261
l̄	006C 0329	ɪ	026A
m	006D	ɹ	0279
n	006E	ʃ	0283
ɳ	006E 0329	ʊ	028A
ou	006F 028A	ʌ	028C
p	0070	ʍ	028D
s	0073	ʒ	0292
t	0074	dʒ	02A4
u	0075	tʃ	02A7
v	0076	θ	03B8

## Identifying personal health information (PHI) in a transcription

Use *Personal Health Information Identification* to label personal health information (PHI) in your transcription results. By reviewing labels, you can find PHI that could be used to identify a patient.

You can identify PHI using either a real-time stream or batch transcription job.

You can use your own post-processing to redact the PHI identified in the transcription output.

Use Personal Health Information Identification to identify the following types of PHI:

- Personal PHI:

- Names – Full name or last name and initial
- Gender
- Age
- Phone numbers
- Dates (not including the year) that directly relate to the patient
- Email addresses
- Geographic PHI:
  - Physical address
  - Zip code
  - Name of medical center or practice
- Account PHI:
  - Fax numbers
  - Social security numbers (SSNs)
  - Health insurance beneficiary numbers
  - Account numbers
  - Certificate or license numbers
  - Biometric identifiers
  - Voice prints
- Vehicle PHI:
  - Vehicle identification number (VIN)
  - License plate number
- Other PHI:
  - Web Uniform Resource Location (URL)
  - Internet Protocol (IP) address numbers

Amazon Transcribe Medical is a Health Insurance Portability and Accountability Act of 1996 (HIPAA) compliant service. For more information, see [Amazon Transcribe Medical \(p. 241\)](#). For information about identifying PHI in an audio file, see [Identifying PHI in an audio file \(p. 309\)](#). For information about identifying PHI in a stream, see [Identifying PHI in a real-time stream \(p. 313\)](#).

#### Topics

- [Identifying PHI in an audio file \(p. 309\)](#)
- [Identifying PHI in a real-time stream \(p. 313\)](#)

## Identifying PHI in an audio file

Use a batch transcription job to transcribe audio files and identify the personal health information (PHI) within them. When you activate Personal Health Information (PHI) Identification, Amazon Transcribe Medical labels the PHI that it identified in the transcription results. For information about the PHI that Amazon Transcribe Medical can identify, see [Identifying personal health information \(PHI\) in a transcription \(p. 308\)](#).

You can start a batch transcription job using either the [StartMedicalTranscriptionJob](#) API or the AWS Management Console.

### AWS Management Console

To use the AWS Management Console to transcribe a clinician-patient dialogue, create a transcription job and choose **Conversation for Audio input type**.

## To transcribe an audio file and identify its PHI (AWS Management Console)

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Transcription jobs**.
3. Choose **Create job**.
4. On the **Specify job details** page, under **Job settings**, specify the following.
  - a. **Name** – The name of the transcription job that is unique to your AWS account.
  - b. **Audio input type – Conversation or Dictation**.
5. For the remaining fields, specify the Amazon S3 location of your audio file and where you want to store the output of your transcription job.
6. Choose **Next**.
7. Under **Audio settings**, choose **PHI Identification**.
8. Choose **Create**.

## API

### To transcribe an audio file and identify its PHI using a batch transcription job (API)

- For the [StartMedicalTranscriptionJob](#) API, specify the following.
  - a. For `MedicalTranscriptionJobName`, specify a name that is unique to your AWS account.
  - b. For `LanguageCode`, specify the language code that corresponds to the language spoken in your audio file.
  - c. For the `MediaFileUri` parameter of the `Media` object, specify the name of the audio file that you want to transcribe.
  - d. For `Specialty`, specify the medical specialty of the clinician speaking in the audio file as `PRIMARYCARE`.
  - e. For `Type`, specify either `CONVERSATION` or `DICTATION`.
  - f. For `OutputBucketName`, specify the Amazon S3 bucket where you want to store the transcription results.

The following is an example request that uses the AWS SDK for Python (Boto3) to transcribe an audio file and identify the PHI of a patient.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-transcription-job"
job_uri = "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    ContentIdentificationType = 'PHI',
    Specialty = 'PRIMARYCARE',
    Type = 'type', # Specify 'CONVERSATION' for a medical conversation. Specify
    'DICTATION' for a medical dictation.
    OutputBucketName = 'DOC-EXAMPLE-BUCKET'
)
while True:
    status =
    transcribe.get_medical_transcription_job(MedicalTranscriptionJobName=job_name)
```

```

        if status['MedicalTranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED',
'FAILED']:
    break
    print("Not ready yet...")
    time.sleep(5)
print(status)

```

The following example code shows the transcription results with patient PHI identified.

```
{
    "jobName": "my-medical-transcription-job-name",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "The patient's name is Bertrand."
            }
        ],
        "items": [
            {
                "start_time": "0.0",
                "end_time": "0.37",
                "alternatives": [
                    {
                        "confidence": "0.9993",
                        "content": "The"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "0.37",
                "end_time": "0.44",
                "alternatives": [
                    {
                        "confidence": "0.9981",
                        "content": "patient's"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "0.44",
                "end_time": "0.52",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "name"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "0.52",
                "end_time": "0.92",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "is"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "start_time": "0.92",
                "end_time": "0.9989",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Bertrand"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],

```

```
        "type": "punctuation"
    }],
    "entities": [
        {
            "content": "Bertrand",
            "category": "PHI*-Personal*",
            "startTime": 0.92,
            "endTime": 1.2,
            "confidence": 0.9989
        }
    ],
    "status": "COMPLETED"
}
```

## AWS CLI

### To transcribe an audio file and identify PHI using a batch transcription job (AWS CLI)

- Run the following code.

```
aws transcribe start-medical-transcription-job \
--medical-transcription-job-name my-medical-transcription-job-name \
--language-code en-US \
--media MediaFileUri="s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac" \
--output-bucket-name DOC-EXAMPLE-BUCKET \
--specialty PRIMARYCARE \
--type type \ # Choose CONVERSATION to transcribe a medical conversation.
Choose DICTATION to transcribe a medical dictation.
--content-identification-type PHI
```

The following is the response from running the preceding AWS CLI command.

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "my-first-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "en-US",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
        },
        "StartTime": "2021-04-27T22:21:52.505000+00:00",
        "CreationTime": "2021-04-27T22:21:52.459000+00:00",
        "ContentIdentificationType": "PHI",
        "Specialty": "PRIMARYCARE",
        "Type": "type"
    }
}
```

## Identifying PHI in a real-time stream

You can identify Personally Identifiable Health Information (PHI) in either HTTP/2 or WebSocket streams. When you activate PHI Identification, Amazon Transcribe Medical labels the PHI that it identifies in the transcription results. For information about the PHI that Amazon Transcribe Medical can identify, see [Identifying personal health information \(PHI\) in a transcription \(p. 308\)](#).

### Identifying PHI in a dictation that is spoken into your microphone

To use the AWS Management Console to transcribe the speech picked up by your microphone and identify any PHI, choose **Dictation** as the audio input type, start the stream, and begin speaking into the microphone on your computer.

#### To identify PHI in a dictation using the AWS Management Console

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, choose **Real-time transcription**.
3. For **Audio input type**, choose **Dictation**.
4. For **Additional settings**, choose **PHI identification**.
5. Choose **Start streaming** and speak into the microphone.
6. Choose **Stop streaming** to end the dictation.

### Identifying PHI in an HTTP/2 stream

To start an HTTP/2 stream with PHI Identification activated, use the [StartMedicalStreamTranscription](#) API and specify the following:

- For **LanguageCode**, specify the language code for the language spoken in the stream. For US English, specify **en-US**.
- For **MediaSampleHertz**, specify the sample rate of the audio.
- For **content-identification-type**, specify **PHI**.

### Identifying PHI in a WebSocket stream

To start a WebSocket stream with PHI Identification activated, use the following format to create a pre-signed URL.

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/medical-stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=250
&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
```

Parameter definitions can be found in the [API Reference](#); parameters common to all AWS API operations are listed in the [Common Parameters](#) section.

The following example is an example response to a streaming request with PHI Identification activated. For brevity, metadata has been removed.

```
{
    "TranscriptResultStream": {
        "TranscriptEvent": {
            "Transcript": {
                "Results": [
                    {
                        "Alternatives": [
                            {
                                "Transcript": "my name is john doe",
                                "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": "john",
                                        "EndTime": 0.9199375,
                                        "StartTime": 0.7999375,
                                        "Type": "pronunciation"
                                    },
                                    {
                                        "Content": "doe",
                                        "EndTime": 1.0199375,
                                        "StartTime": 0.9299375,
                                        "Type": "pronunciation"
                                    }
                                ],
                                "Entities": [
                                    {
                                        "Content": "john doe",
                                        "Category": "PHI-Personal",
                                        "Type": "Personal",
                                        "StartTime": 0.7999375,
                                        "EndTime": 1.0199375,
                                        "Confidence": 0.9989
                                    }
                                ]
                            ],
                            "EndTime": 1.02,
                            "IsPartial": false,
                            "ResultId": "2db76dc8-d728-11e8-9f8b-f2801f1b9fd1",
                            "StartTime": 0.0199375
                        }
                    ]
                }
            }
        }
    }
}
```

}

## Generating alternative transcriptions

When you use Amazon Transcribe Medical, you get the transcription that has the highest confidence level. However, you can configure Amazon Transcribe Medical to return additional transcriptions with lower confidence levels.

Use alternative transcriptions to see different interpretations of the transcribed audio. For example, in an application that enables a person to review the transcription, you can present the alternative transcriptions for the person to choose from.

You can generate alternative transcriptions with the AWS Management Console or the [StartMedicalTranscriptionJob](#) API.

### AWS Management Console

To use the AWS Management Console to generate alternative transcriptions, you enable alternative results when you configure your job.

1. Sign in to the [AWS Management Console](#).
2. In the navigation pane, under Amazon Transcribe Medical, choose **Transcription jobs**.
3. Choose **Create job**.
4. On the **Specify job details** page, provide information about your transcription job.
5. Choose **Next**.
6. Enable **Alternative results**.
7. For **Maximum alternatives**, enter an integer value between 2 and 10, for the maximum number of alternative transcriptions you want in the output.
8. Choose **Create**.

### API

#### To identify speakers in an audio file using a batch transcription job (API)

- For the [StartMedicalTranscriptionJob](#) API, specify the following.
  - a. For `MedicalTranscriptionJobName`, specify a name that is unique in your AWS account.
  - b. For `LanguageCode`, specify the language code that corresponds to the language spoken in your audio file and the language of your vocabulary filter.
  - c. In the `MediaFileUri` parameter of the `Media` object, specify the location of the audio file you want to transcribe.
  - d. For `Specialty`, specify the medical specialty of the clinician speaking in the audio file.
  - e. For `Type`, specify whether you're transcribing a medical conversation or a dictation.
  - f. For `OutputBucketName`, specify the Amazon S3 bucket to store the transcription results.
  - g. For the `Settings` object, specify the following.
    - i. `ShowAlternatives` – `true`.
    - ii. `MaxAlternatives` - An integer between 2 and 10 to indicate the number of alternative transcriptions you want in the transcription output.

The following request uses the AWS SDK for Python (Boto3) to start a transcription job that generates up to two alternative transcriptions.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "my-first-transcription-job"
job_uri = s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName=job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'type', # Specify 'CONVERSATION' for a medical conversation. Specify 'DICTATION' for a medical dictation.
    OutputBucketName = 's3://DOC-EXAMPLE-BUCKET'
),
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

**To transcribe an audio file of a conversation between a primary care clinician and a patient in an audio file and identify what each person said in the transcription output (AWS CLI)**

- Run the following code.

```
aws transcribe start-transcription-job \
--cli-input-json file://filepath/example-start-command.json
```

The following code shows the contents of example-start-command.json.

```
{
    "MedicalTranscriptionJobName": "my-first-transcription-job",
    "LanguageCode": "en-US",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName": "DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"
    },
    "Settings": {
        "ShowAlternatives": true,
        "MaxAlternatives": 2
    }
}
```

}

The following is the response from running the preceding AWS CLI command.

```
{  
    "MedicalTranscriptionJob": {  
        "MedicalTranscriptionJobName": "my-first-transcription-job",  
        "TranscriptionJobStatus": "IN_PROGRESS",  
        "LanguageCode": "en-US",  
        "Media": {  
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/my-audio-file.flac"  
        },  
        "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"  
    }  
}
```

## Guidelines and quotas

### Supported AWS Regions

For a list of AWS Regions where Amazon Transcribe Medical is available, see [Amazon Transcribe Endpoints and Quotas](#) in the *AWS General Reference*.

### Guidelines

For best results:

- Use a lossless format, such as FLAC or WAV, with PCM 16-bit encoding.
- Use a sample rate of 16,000 Hz or higher.

Amazon Transcribe Medical may store your content to continuously improve the quality of its analysis models. See the [Amazon Transcribe Medical FAQ](#) to learn more. To request that we delete content that may have been stored by Amazon Transcribe Medical, open a case with AWS Support.

### Quotas

You can request a quota increase for the following resources.

Resource	Default
Number of concurrent batch transcription jobs	250

Resource	Default
Number of concurrent HTTP/2 or WebSocket requests	25
Total number of medical vocabularies per account	100
Number of pending medical vocabularies	10

The following API limits can also be increased upon request:

API	Maximum transactions per second
<a href="#">StartMedicalTranscriptionJob</a>	25
<a href="#">StartMedicalStreamTranscription</a>	25
<a href="#">GetMedicalTranscriptionJob</a>	30
<a href="#">DeleteMedicalTranscriptionJob</a>	5
<a href="#">ListMedicalTranscriptionJobs</a>	5
<a href="#">CreateMedicalVocabulary</a>	10
<a href="#">UpdateMedicalVocabulary</a>	10
<a href="#">DeleteMedicalVocabulary</a>	5
<a href="#">GetMedicalVocabulary</a>	20
<a href="#">ListMedicalVocabularies</a>	5

**Note**

For information about requesting a quota increase, see [AWS service quotas](#) in the *AWS General Reference*.

The following quotas **cannot** be increased:

Description	Quotas
Maximum audio file length	14,400 seconds
Maximum audio file size	2 GB
Number of days that job records are retained	90
Number of channels for channel identification	2
Minimum audio file duration, in milliseconds (ms)	500

# Amazon Transcribe Medical and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and Amazon Transcribe Medical by creating an *interface VPC endpoint*. Interface endpoints are powered by [AWS PrivateLink](#), a technology that enables you to privately access Amazon Transcribe Medical APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with Amazon Transcribe Medical APIs. Traffic between your VPC and Amazon Transcribe Medical does not leave the Amazon network.

Each interface endpoint is represented by one or more [Elastic Network Interfaces](#) in your subnets.

For more information, see [Interface VPC endpoints \(AWS PrivateLink\)](#) in the *Amazon VPC User Guide*.

## Considerations for Amazon Transcribe Medical VPC endpoints

Before you set up an interface VPC endpoint for Amazon Transcribe Medical, ensure that you review [Interface endpoint properties and limitations](#) in the *Amazon VPC User Guide*.

Amazon Transcribe Medical supports making calls to all of its API actions from your VPC.

## Creating an interface VPC endpoint for Amazon Transcribe Medical

You can create a VPC endpoint for the Amazon Transcribe Medical service using either the AWS Management Console or the AWS CLI. For more information, see [Creating an interface endpoint](#) in the *Amazon VPC User Guide*.

For batch transcription in Amazon Transcribe Medical, create a VPC endpoint using the following service name:

- com.amazonaws.**us-west-2**.transcribe

For streaming transcription in Amazon Transcribe Medical, create a VPC endpoint using the following service name:

- com.amazonaws.**us-west-2**.transcribestreaming

If you enable private DNS for the endpoint, you can make API requests to Amazon Transcribe Medical using its default DNS name for the AWS Region, for example, `transcribestreaming.us-east-2.amazonaws.com`.

For more information, see [Accessing a service through an interface endpoint](#) in the *Amazon VPC User Guide*.

## Creating a VPC endpoint policy for Amazon Transcribe Medical streaming

You can attach an endpoint policy to your VPC endpoint that controls access to Amazon Transcribe Medical. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see [Controlling access to services with VPC endpoints](#) in the *Amazon VPC User Guide*.

#### **Example: VPC endpoint policy for Amazon Transcribe Medical streaming transcription actions**

The following is an example of an endpoint policy for streaming transcription in Amazon Transcribe Medical. When attached to an endpoint, this policy grants access to the listed Amazon Transcribe Medical actions for all principals on all resources.

```
{  
    "Statement": [  
        {  
            "Principal": "*",
            "Effect": "Allow",
            "Action": [  
                "transcribe:StartMedicalStreamTranscription",
            ],
            "Resource": "*"
        }
    ]
}
```

#### **Example: VPC endpoint policy for Amazon Transcribe Medical batch transcription actions**

The following is an example of an endpoint policy for batch transcription in Amazon Transcribe Medical. When attached to an endpoint, this policy grants access to the listed Amazon Transcribe Medical actions for all principals on all resources.

```
{  
    "Statement": [  
        {  
            "Principal": "*",
            "Effect": "Allow",
            "Action": [  
                "transcribe:StartMedicalTranscriptionJob"
            ],
            "Resource": "*"
        }
    ]
}
```

# Document history for Amazon Transcribe and Amazon Transcribe Medical

- **Latest documentation update:** 1 April 2022

The following table describes important changes in each release of Amazon Transcribe. For notification about updates to this documentation, you can subscribe to an RSS feed.

update-history-change	update-history-description	update-history-date
<a href="#">Guide Update</a>	The Amazon Transcribe API Reference is now a standalone guide.	April 1, 2022
<a href="#">New chapter</a>	A new comparison table for Amazon Transcribe, Amazon Transcribe Medical, and Amazon Transcribe Call Analytics is included.	March 21, 2022
<a href="#">New chapter</a>	A new SDK code examples chapter is included.	March 21, 2022
<a href="#">Feature update</a>	Call Analytics now provides call summarization.	March 21, 2022
<a href="#">Chapter update</a>	The introductory chapter now showcases Amazon Transcribe use cases.	March 21, 2022
<a href="#">Chapter update</a>	The Getting started chapter has been updated to be method-specific.	March 21, 2022
<a href="#">Chapter update</a>	The Streaming chapter has been updated and restructured.	March 21, 2022
<a href="#">Feature update</a>	Language identification now supports custom vocabularies and vocabulary filters with streaming transcriptions.	March 11, 2022
<a href="#">New event</a>	There is a new event type: Vocabulary events.	February 7, 2022
<a href="#">Section update</a>	Updates have been made to the custom vocabularies section.	January 20, 2022
<a href="#">New feature</a>	Language identification can now be used with streaming transcriptions.	November 23, 2021

New feature	Language identification can now be used with custom language models, custom vocabularies, vocabulary filtering, and content redaction.	October 29, 2021
New feature	Amazon Transcribe now supports custom language models with streaming transcriptions.	October 20, 2021
New feature	Amazon Transcribe can now generate subtitles for your video files.	September 16, 2021
New feature	Amazon Transcribe now supports PII redaction and identification for streaming.	September 14, 2021
New feature	Amazon Transcribe now supports AWS KMS encryption context for an added level of security for your account resources.	September 10, 2021
New languages	Amazon Transcribe now supports Afrikaans, Danish, Mandarin Chinese (Traditional), Thai, New Zealand English, and South African English.	August 26, 2021
New feature	Amazon Transcribe now supports resource tagging.	August 24, 2021
New feature	Amazon Transcribe now supports Call Analytics for batch transcription jobs.	August 4, 2021
New feature	Amazon Transcribe Medical now supports personal health information (PHI) identification in batch transcription jobs.	May 14, 2021
New feature	Amazon Transcribe now supports using custom vocabularies with batch custom language models.	May 12, 2021
New feature	Amazon Transcribe now supports partial result stabilization for streaming transcription.	May 11, 2021
New feature	Amazon Transcribe now supports Australian English, British English, Hindi, and US Spanish for custom language models.	March 19, 2021

New feature	Amazon Transcribe now supports Australian English, British English, Hindi, and US Spanish for custom language models.	March 19, 2021
New feature	Amazon Transcribe Medical now supports personal health information identification.	January 29, 2021
New feature	Amazon Transcribe Medical now supports channel identification.	December 21, 2020
New feature	Amazon Transcribe Medical now supports the HTTP/2 streaming protocol.	November 24, 2020
New feature	Amazon Transcribe Medical now supports the neurology medical specialty for streaming audio transcription.	November 24, 2020
New feature	Amazon Transcribe Medical now supports the urology medical specialty for streaming audio transcription.	November 24, 2020
New feature	Amazon Transcribe Medical now supports the radiology medical specialty for streaming audio transcription.	November 24, 2020
New feature	Amazon Transcribe Medical now supports the cardiology medical specialty for streaming audio transcription.	November 24, 2020
New feature	Amazon Transcribe Medical now supports the oncology medical specialty for streaming audio transcription.	November 24, 2020
New feature	Amazon Transcribe now supports OGG/OPUS and FLAC codecs for streaming audio transcription.	November 24, 2020
New languages	Amazon Transcribe adds support for Italian and German for streaming audio transcription.	November 4, 2020
AWS Region expansion	Amazon Transcribe is now available in the Frankfurt (eu-central-1) and London (eu-west-2).	November 4, 2020

New feature	Amazon Transcribe adds support for interface VPC endpoints in batch transcription. For more information, see <a href="#">Amazon Transcribe and interface VPC endpoints (AWS PrivateLink)</a> .	October 9, 2020
New feature	Amazon Transcribe adds support for channel identification in streaming. For more information, see <a href="#">Transcribing Multi-Channel Audio</a> .	September 17, 2020
New feature	Amazon Transcribe adds support for automatic language identification in batch transcription. For more information, see <a href="#">Identifying the Language</a> .	September 15, 2020
New feature	Amazon Transcribe adds support for speaker identification in streaming. For more information, see <a href="#">Filtering Streaming Transcriptions</a> .	August 19, 2020
New feature	Amazon Transcribe adds support for custom language models. For more information, see <a href="#">Improving Transcription Accuracy with Custom Models</a> .	August 5, 2020
New feature	Amazon Transcribe adds support for interface VPC endpoints in streaming. For more information, see <a href="#">Amazon Transcribe and interface VPC endpoints (AWS PrivateLink)</a> .	June 26, 2020
New feature	Amazon Transcribe adds support for vocabulary filtering in streaming. For more information, see <a href="#">Filtering Streaming Transcriptions</a> .	May 20, 2020
New feature	Amazon Transcribe Medical adds support for custom vocabularies in both batch processing and streaming. For more information, see <a href="#">Medical Custom Vocabularies</a> .	April 29, 2020
New feature	Amazon Transcribe Medical adds support for batch processing of audio files. For more information, see <a href="#">Batch Transcription Overview</a> .	April 1, 2020

New feature	Amazon Transcribe adds support for automatically redacting personally identifiable information. For more information, see <a href="#">Automatic Content Redaction</a> .	February 26, 2020
New feature	Amazon Transcribe adds support for creating a vocabulary of words to filter from a transcription. For more information, see <a href="#">Vocabulary Filtering</a> .	December 20, 2019
New feature	Amazon Transcribe adds support for queuing transcription jobs. For more information, see <a href="#">Job Queuing</a> .	December 19, 2019
New languages	Amazon Transcribe adds support for Gulf Arabic, Hebrew, Japanese, Malay, Swiss German, Telugu, and Turkish.	November 21, 2019
AWS Region expansion	Amazon Transcribe is now available in the Asia Pacific (Tokyo) (ap-northeast-1).	November 21, 2019
New feature	Amazon Transcribe adds support for alternative transcriptions. For more information, see <a href="#">Alternative Transcriptions</a> .	November 20, 2019
New languages	Amazon Transcribe adds support for Dutch, Farsi, Indonesian, Irish English, Portuguese, Scottish English, Tamil, and Welsh English.	November 12, 2019
New language	Amazon Transcribe now supports streaming transcription for Australian English (en-AU).	October 25, 2019
AWS Region expansion	Amazon Transcribe is now available in the China (Beijing) (cn-north-1) and China (Ningxia) (cn-northwest-1).	October 9, 2019
New feature	Amazon Transcribe enables you to provide your own KMS key to encrypt your transcription output files. For more information, see the <a href="#">OutputEncryptionKMSKeyId</a> parameter of the <a href="#">StartStreamTranscription</a> API.	September 24, 2019

New languages	Amazon Transcribe adds support for Chinese (Mandarin), Simplified, Mainland China and Russian.	August 23, 2019
New feature	Amazon Transcribe adds support for streaming audio transcription using the WebSocket protocol. For more information, see <a href="#">Streaming Transcription</a> .	July 19, 2019
New feature	AWS CloudTrail now records events for the <a href="#">StartStreamTranscription</a> API.	July 19, 2019
AWS Region expansion	Amazon Transcribe is now available in the US West (N. California) (us-west-1).	June 27, 2019
New language	Amazon Transcribe adds support for Modern Standard Arabic.	May 28, 2019
New feature	Amazon Transcribe now transcribes numeric words into numbers for US English. For example, "forty-two" is transcribed as "42". For more information, see <a href="#">Transcribing Numbers</a> .	May 23, 2019
New language	Amazon Transcribe adds support for Hindi and Indian English.	May 15, 2019
New SDK	The AWS SDK for C++ now supports Amazon Transcribe.	May 8, 2019
New language	Amazon Transcribe adds support for Spanish.	April 19, 2019
AWS Region expansion	Amazon Transcribe is now available in the EU (Frankfurt) (eu-central-1) and Asia Pacific (Seoul) (ap-northeast-2).	April 18, 2019
New language	Amazon Transcribe adds support for streaming transcription in British English, French, and Canadian French.	April 5, 2019
New feature	The AWS SDK for Ruby V3 now supports Amazon Transcribe	March 25, 2019

New feature	Amazon Transcribe now enables you to create custom vocabularies, lists of specific words that you want Amazon Transcribe to recognize in your audio input. For more information, see <a href="#">Custom Vocabularies</a> .	March 25, 2019
New languages	Amazon Transcribe adds support for German and Korean.	March 22, 2019
New language	Amazon Transcribe now supports streaming transcription for US Spanish (es-US).	February 7, 2019
AWS Region expansion	Amazon Transcribe is now available in the South America (Sao Paulo) (sa-east-1).	February 7, 2019
AWS Region expansion	Amazon Transcribe is now available in the Asia Pacific (Mumbai) (ap-south-1), Asia Pacific (Singapore) (ap-southeast-1), EU (London) (eu-west-2), and EU (Paris) (eu-west3).	January 24, 2019
New languages	Amazon Transcribe adds support for French, Italian, and Brazilian Portuguese.	December 20, 2018
New feature	Amazon Transcribe now supports transcription of audio streams. For more information, see <a href="#">Streaming Transcription</a> .	November 19, 2018
New languages	Amazon Transcribe adds support for Australian English, British English, and Canadian French.	November 15, 2018
AWS Region expansion	Amazon Transcribe is now available in Canada (Central) (ca-central-1) and Asia Pacific (Sydney) (ap-southeast-2).	July 17, 2018
New feature	You can now specify your own location to store the output from a transcription job. For more information, see the <a href="#">TranscriptionJobSummary</a> data type.	July 11, 2018
New feature	Added AWS CloudTrail and Amazon CloudWatch Events integration. For more information, see <a href="#">Monitoring Amazon Transcribe</a> .	June 28, 2018

New feature

Amazon Transcribe adds support for custom vocabularies April 4, 2018

New guide (p. 321)

This is the first release of the *Amazon Transcribe Developer Guide*.

November 29, 2017

# AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.