# Table of Contents

- **What Is Amazon Transcribe?** .......................................................................................................................... 1  
  - Recognizing Voices ................................................................................................................................................. 2  
  - Transcribing Separate Audio Channels ................................................................................................................... 2  
  - Transcribing Streaming Audio ................................................................................................................................. 3  
  - Custom Vocabulary .................................................................................................................................................... 3  
- **Are You a First-time User of Amazon Transcribe ?** .............................................................................................. 3  
- **How It Works** ....................................................................................................................................................... 4  
  - Speech Input ........................................................................................................................................................... 5  
    - Audio Containers and Formats For Batch Transcription ......................................................................................... 6  
    - Audio Containers and Formats For Streaming Transcription .................................................................................... 6  
  - Transcribing Numbers ................................................................................................................................................. 6  
    - Rules for Transcribing Numbers in English ........................................................................................................... 7  
    - Rules for Transcribing Numbers in German ........................................................................................................ 8  
- **Alternative Transcriptions** ...................................................................................................................................... 9  
- **Transcribing Streaming Audio** ............................................................................................................................... 11  
  - Custom Vocabularies .................................................................................................................................................. 12  
    - Create a Custom Vocabulary Using a List ................................................................................................................ 13  
    - Create a Custom Vocabulary Using a Table ........................................................................................................... 13  
    - Character Sets for Custom Vocabularies ................................................................................................................ 15  
- **Automatic Content Redaction** .............................................................................................................................. 44  
- **Job Queuing** ......................................................................................................................................................... 47  
  - IAM Policies for Job Queuing ................................................................................................................................. 47  
- **Getting Started** ..................................................................................................................................................... 50  
  - Step 1: Set Up an Account ........................................................................................................................................... 50  
    - Sign up for AWS ....................................................................................................................................................... 50  
    - Create an IAM User ............................................................................................................................................... 50  
    - Next Step ............................................................................................................................................................... 50  
  - Step 2: Set Up the AWS CLI ........................................................................................................................................ 51  
    - Next Step ............................................................................................................................................................... 51  
  - Step 3: Getting Started Using the Console ............................................................................................................... 52  
    - Create a Transcription Job ..................................................................................................................................... 52  
    - View a Transcription Job ....................................................................................................................................... 53  
  - Step 4: Getting Started Using the API ..................................................................................................................... 56  
    - Getting Started (AWS CLI) .................................................................................................................................... 56  
    - Getting Started (SDK for Python) ........................................................................................................................ 58  
  - Step 5: Getting Started With Streaming Audio ...................................................................................................... 60  
- **Improving Transcription Accuracy with Custom Models** .................................................................................... 64  
  - Step 1: Preparing the Data ....................................................................................................................................... 65  
  - Step 2: Providing Amazon Transcribe with Data Permissions ................................................................................ 66  
  - Step 3: Creating a Custom Language Model ......................................................................................................... 68  
    - Using S3 Prefixes to Retrieve Your Data .............................................................................................................. 68  
  - Step 4: Transcribing with a Custom Language Model ............................................................................................ 70  
  - Step 5: Viewing and Updating Your Custom Language Models ............................................................................. 72  
- **Streaming Transcription** ........................................................................................................................................... 73  
  - Event Stream Encoding ............................................................................................................................................. 76  
  - Using WebSocket Streaming ..................................................................................................................................... 77  
    - Adding a Policy for WebSocket Requests to Your IAM Role .................................................................................. 77  
    - Creating a Pre-Signed URL ................................................................................................................................... 78  
    - Handling the WebSocket Upgrade Response ........................................................................................................ 81  
    - Making a WebSocket Streaming Request ............................................................................................................ 82  
    - Handling a WebSocket Streaming Response ....................................................................................................... 82  
    - Handling WebSocket Streaming Errors ............................................................................................................... 83  
  - Using HTTP/2 Streaming ......................................................................................................................................... 83  
  - Streaming Request ................................................................................................................................................... 84
Document content not provided in the given image.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering Streaming Transcriptions</td>
<td>178</td>
</tr>
<tr>
<td>Identifying Speakers</td>
<td>181</td>
</tr>
<tr>
<td>Identifying Speakers in Audio Files</td>
<td>181</td>
</tr>
<tr>
<td>Identifying Speakers in Real-time Streams</td>
<td>184</td>
</tr>
<tr>
<td>HTTP/2 Streaming</td>
<td>185</td>
</tr>
<tr>
<td>Websocket Streaming</td>
<td>185</td>
</tr>
<tr>
<td>Streaming Transcription Output</td>
<td>186</td>
</tr>
<tr>
<td>Transcribing Multi-Channel Audio</td>
<td>189</td>
</tr>
<tr>
<td>Transcribing Multi-Channel Audio Files</td>
<td>189</td>
</tr>
<tr>
<td>Transcribing Multi-Channel Audio Streams</td>
<td>193</td>
</tr>
<tr>
<td>Transcribing Multi-Channel Audio in an HTTP/2 Stream</td>
<td>193</td>
</tr>
<tr>
<td>Transcribing Multi-Channel Audio in a WebSocket Stream</td>
<td>194</td>
</tr>
<tr>
<td>Multi-Channel Streaming Output</td>
<td>194</td>
</tr>
<tr>
<td>Security</td>
<td>196</td>
</tr>
<tr>
<td>Data Protection</td>
<td>196</td>
</tr>
<tr>
<td>Encryption at Rest</td>
<td>197</td>
</tr>
<tr>
<td>Encryption in Transit</td>
<td>197</td>
</tr>
<tr>
<td>Key Management</td>
<td>197</td>
</tr>
<tr>
<td>Opting Out of Using Your Data for Service Improvement</td>
<td>198</td>
</tr>
<tr>
<td>VPC endpoints (AWS PrivateLink)</td>
<td>199</td>
</tr>
<tr>
<td>Identity and Access Management</td>
<td>201</td>
</tr>
<tr>
<td>Audience</td>
<td>201</td>
</tr>
<tr>
<td>Authenticating with Identities</td>
<td>201</td>
</tr>
<tr>
<td>Managing Access Using Policies</td>
<td>203</td>
</tr>
<tr>
<td>How Amazon Transcribe Works with IAM</td>
<td>205</td>
</tr>
<tr>
<td>Identity-Based Policy Examples</td>
<td>208</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>212</td>
</tr>
<tr>
<td>Monitoring Amazon Transcribe</td>
<td>214</td>
</tr>
<tr>
<td>Monitoring Amazon Transcribe with CloudTrail</td>
<td>214</td>
</tr>
<tr>
<td>CloudWatch Events</td>
<td>216</td>
</tr>
<tr>
<td>Compliance Validation</td>
<td>219</td>
</tr>
<tr>
<td>Resilience</td>
<td>219</td>
</tr>
<tr>
<td>Infrastructure Security</td>
<td>220</td>
</tr>
<tr>
<td>Guidelines and Quotas</td>
<td>221</td>
</tr>
<tr>
<td>Supported Regions</td>
<td>163</td>
</tr>
<tr>
<td>Throttling</td>
<td>221</td>
</tr>
<tr>
<td>Guidelines</td>
<td>222</td>
</tr>
<tr>
<td>Quotas</td>
<td>223</td>
</tr>
<tr>
<td>Document History</td>
<td>224</td>
</tr>
<tr>
<td>API Reference</td>
<td>230</td>
</tr>
<tr>
<td>Actions</td>
<td>230</td>
</tr>
<tr>
<td>Amazon Transcribe Service</td>
<td>231</td>
</tr>
<tr>
<td>Amazon Transcribe Streaming Service</td>
<td>312</td>
</tr>
<tr>
<td>Data Types</td>
<td>324</td>
</tr>
<tr>
<td>Amazon Transcribe Service</td>
<td>324</td>
</tr>
<tr>
<td>Amazon Transcribe Streaming Service</td>
<td>353</td>
</tr>
<tr>
<td>Common Errors</td>
<td>373</td>
</tr>
<tr>
<td>Common Parameters</td>
<td>374</td>
</tr>
<tr>
<td>AWS glossary</td>
<td>377</td>
</tr>
</tbody>
</table>
What Is Amazon Transcribe?

Amazon Transcribe uses advanced machine learning technologies to recognize speech in audio files and transcribe them into text. You can use Amazon Transcribe to convert audio to text and to create applications that incorporate the content of audio files. For example, you can transcribe the audio track from a video recording to create closed captioning for the video.

You can use Amazon Transcribe to transcribe the following languages to text:

- Gulf Arabic (ar-AE)
- Modern Standard Arabic (ar-SA)
- Mandarin Chinese - Mainland (zh-CN)
- Dutch (nl-NL)
- Australian English (en-AU)
- British English (en-GB)
- Indian English (en-IN)
- Irish English (en-IE)
- Scottish English (en-AB)
- US English (en-US)
- Welsh English (en-WL)
- Spanish (es-ES)
- US Spanish (es-US)
- French (fr-FR)
- Canadian French (fr-CA)
- Farsi (fa-IR)
- German (de-DE)
- Swiss German (de-CH)
- Hebrew (he-IL)
- Indian Hindi (hi-IN)
- Indonesian (id-ID)
- Italian (it-IT)
- Japanese (ja-JP)
- Korean (ko-KR)
- Malay (ms-MY)
- Portuguese (pt-PT)
- Brazilian Portuguese (pt-BR)
- Russian (ru-RU)
- Tamil (ta-IN)
- Telugu (te-IN)
- Turkish (tr-TR)

You can transcribe streaming audio in the following languages.

<table>
<thead>
<tr>
<th>Language</th>
<th>Supported Sample Rates</th>
<th>Supported In</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australian English (en-AU)</td>
<td>8 kHz</td>
<td>API</td>
</tr>
</tbody>
</table>
Recognizing Voices

You can use Amazon Transcribe with other AWS services to create applications. For example, you can:

- Use Amazon Transcribe to convert voice to text, send the text to Amazon Translate to translate it into another language, and send the translated text to Amazon Polly to speak the translated text.
- Use Amazon Transcribe to transcribe recordings of customer service calls for analysis. After transcribing a recording, send the transcription to Amazon Comprehend to identify keywords, topics, or sentiments.
- Use Amazon Transcribe to transcribe live broadcasts such as television to provide real-time subtitles.

To use Amazon Transcribe you store your audio file in an Amazon S3 bucket. The output from the transcription job is also stored in an S3 bucket. Content delivered to Amazon S3 buckets might contain customer content. For more information about removing sensitive data, see How Do I Empty an S3 Bucket? or How Do I Delete an S3 Bucket?.

Recognizing Voices

Amazon Transcribe can identify the individual speakers in an audio clip, a technique known as **diarization** or **speaker identification**. When you activate speaker identification, Amazon Transcribe includes an attribute that identifies each speaker in the audio clip. You can use speaker identification to:

- identify the customer and the support representative in a recorded customer support call
- identify characters for closed captions
- identify the speaker and questioners in a recorded press conference or lecture

You can specify the number of voices that you want Amazon Transcribe to recognize in an audio clip.

Transcribing Separate Audio Channels

To create a transcript for each channel, or single stream of recorded sound, in an audio file, use **channel identification**. With channel identification, Amazon Transcribe returns two or more transcriptions: a combined transcription of all of the audio channels and a transcription of each audio channel.
Use channel identification when your audio is on multiple channels. For example, use channel identification:

- When your recording has a customer service representative on one channel and a customer on another
- When you transcribe a podcast where the host is recorded on one channel and the guest on another

For more information about channel identification, see Transcribing Multi-Channel Audio (p. 189).

Transcribing Streaming Audio

You can use Amazon Transcribe to transcribe streaming audio in real-time. You send Amazon Transcribe a stream of audio and Amazon Transcribe returns a stream of JSON objects containing the transcription of the audio.

For more information about processing audio streams, see Streaming Transcription (p. 73).

Custom Vocabulary

Create a custom vocabulary to help Amazon Transcribe recognize words that are specific to your use case and improve its accuracy in converting speech to text. For example, you might create a custom vocabulary that includes industry-specific words and phrases.

Use a custom vocabulary to help Amazon Transcribe recognize:

- words that are not being recognized
- unfamiliar words that are specific to your domain

For more information about creating a custom vocabulary, see Custom Vocabularies (p. 12).

Are You a First-time User of Amazon Transcribe?

If you are a first-time user, we recommend that you read the following sections in order:

1. How Amazon Transcribe Works (p. 4)—Introduces Amazon Transcribe.
2. Getting Started with Amazon Transcribe (p. 50)—Explains how to set up your AWS account and use Amazon Transcribe.
3. API Reference (p. 230)—Contains reference documentation for Amazon Transcribe operations.
How Amazon Transcribe Works

Amazon Transcribe analyzes audio files that contain 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.

To transcribe an audio file, Amazon Transcribe uses three operations:

- **StartTranscriptionJob** (p. 297) – Starts a batch job to transcribe the speech in an audio file to text.
- **ListTranscriptionJobs** (p. 283) – Returns a list of transcription jobs that have been started. You can specify the status of the jobs that you want the operation to return. For example, you can get a list of all pending jobs, or a list of completed jobs.
- **GetTranscriptionJob** (p. 265) – Returns the result of a transcription job. The response contains a link to a JSON file containing the results.

To transcribe streaming audio to text, Amazon Transcribe provides one operation:

- **StartStreamTranscription** (p. 319) – Starts a bi-directional HTTP/2 stream where audio is streamed to Amazon Transcribe and the transcription results are streamed to your application.

You can also start a WebSocket protocol stream to send audio to the Amazon Transcribe. For more information, see Using Amazon Transcribe Streaming with WebSockets (p. 77).

You can use Amazon Transcribe to create and manage custom vocabularies for your solution. A custom vocabulary gives Amazon Transcribe more information about how to process speech in an audio clip.

- **CreateVocabulary** (p. 238) – Creates a custom vocabulary that you can use in your transcription jobs.
- **DeleteVocabulary** (p. 253) – Deletes a custom vocabulary from your account.
- **GetVocabulary** (p. 268) – Gets information about a custom vocabulary and a URL that you can use to download the contents of a vocabulary.
- **ListVocabularies** (p. 286) – Gets a list of custom vocabularies in your account.
- **UpdateVocabulary** (p. 306) – Updates an existing vocabulary.

You can transcribe speech in any of the following languages:

- Gulf Arabic (ar-AE)
- Modern Standard Arabic (ar-SA)
- Mandarin Chinese - Mainland (zh-CN)
- Dutch (nl-NL)
- Australian English (en-AU)
- British English (en-GB)
- Indian English (en-IN)
- Irish English (en-IE)
- Scottish English (en-AB)
- US English (en-US)
- Welsh English (en-WL)
- French (fr-FR)
- Canadian French (fr-CA)
- Farsi (fa-IR)
Speech Input

Amazon Transcribe 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.
Audio Containers and Formats For Batch Transcription

When you transcribe an audio file using the StartTranscriptionJob (p. 297) operation or the Amazon Transcribe console, make sure that the file is:

- In FLAC, MP3, MP4, Ogg, WebM, AMR, or WAV file format
- Less than 4 hours in length or less than 2 GB of audio data

**Note**

For AMR, Amazon Transcribe 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 supports the Opus codec.

For best results:

- Use a lossless format. You can choose either FLAC, or WAV with PCM 16-bit encoding.
- Use a sample rate of 8000 Hz for telephone audio.

Audio Containers and Formats For Streaming Transcription

When you transcribe a real-time stream using the StartStreamTranscription (p. 319) operation or a WebSocket request, make sure that your stream is encoded in:

- PCM 16-bit signed little endian
- FLAC
- OPUS encoded audio in the Ogg container

For best results:

- Use a lossless format, such as FLAC or PCM encoding.
- Use a sample rate of 8000 Hz for telephone audio.

For more information on using a WebSocket request to transcribe your streaming audio, see Using Amazon Transcribe Streaming with WebSockets (p. 77).

Transcribing Numbers

When using the StartTranscriptionJob (p. 297) operation, numbers are transcribed as digits instead of words for the following languages:

- Australian English (en-AU)
- British English (en-GB)
• Indian English (en-IN)
• Irish English (en-IE)
• Scottish English (en-AB)
• US English (en-US)
• Welsh English (en-WL)
• German (de-DE)
• Swiss German (de-CH)

For streaming transcription, numbers are transcribed as digits for the following languages:

• Australian English (en-AU)
• British English (en-GB)
• US English (en-US)

For the preceding languages, the spoken number "one thousand two hundred forty-two" is transcribed as 1242. For all other languages, numbers are transcribed into their word forms.

**Rules for Transcribing Numbers in English**

For all English languages, such British English (en-GB) or US English (en-US), numbers are transcribed according to the following rules.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert cardinal numbers greater than 10 to numbers.</td>
<td>• &quot;Fifty five&quot; &gt; 55&lt;br&gt;• &quot;a hundred&quot; &gt; 100&lt;br&gt;• &quot;One thousand and thirty one&quot; &gt; 1031&lt;br&gt;• &quot;One hundred twenty-three million four hundred fifty six thousand seven hundred eight nine&quot; &gt; 123,456,789</td>
</tr>
<tr>
<td>Convert cardinal numbers followed by &quot;million&quot; or &quot;billion&quot; to numerals followed by a word when &quot;million&quot; or &quot;billion&quot; is not followed by a number.</td>
<td>• &quot;one hundred million&quot; &gt; 100 million&lt;br&gt;• &quot;one billion&quot; &gt; 1 billion&lt;br&gt;• &quot;two point three million&quot; &gt; 2.3 million</td>
</tr>
<tr>
<td>Convert ordinal numbers greater than 10 to numbers.</td>
<td>• &quot;Forty third&quot; &gt; 43rd&lt;br&gt;• &quot;twenty sixth avenue&quot; &gt; 26th avenue</td>
</tr>
<tr>
<td>Convert fractions to their numeric format.</td>
<td>• &quot;a quarter&quot; &gt; 1/4&lt;br&gt;• &quot;three sixteenths&quot; &gt; 3/16&lt;br&gt;• &quot;a half&quot; &gt; 1/2&lt;br&gt;• &quot;a hundredth&quot; &gt; 1/100</td>
</tr>
<tr>
<td>Convert numbers less than 10 to digits if there are more than one in a row.</td>
<td>• &quot;three four five&quot; &gt; 345&lt;br&gt;• &quot;My phone number is four two five five five one two one two&quot; &gt; 4255551212</td>
</tr>
<tr>
<td>Decimals are indicated by &quot;dot&quot; or &quot;point.&quot;</td>
<td>• &quot;three hundred and three dot five&quot; &gt; 303.5&lt;br&gt;• &quot;three point twenty three&quot; &gt; 3.23&lt;br&gt;• &quot;zero point four&quot; &gt; 0.4&lt;br&gt;• &quot;point three&quot; &gt; 0.3</td>
</tr>
</tbody>
</table>

7
Rules for Transcribing Numbers in German

For German (de-DE) and Swiss German (de-CH), numbers are transcribed according to the following rules.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Example</th>
</tr>
</thead>
</table>
| Convert the word "percent" after a number to a percent symbol (%). | • "twenty three percent" > 23%  
• "twenty three point four five percent" > 23.45% |
| Convert the words "dollar," "US dollar," "Australian dollar," "AUD," or "USD" after a number to a dollar sign ($) before the number. | • "one dollar and fifteen cents" > $1.15  
• "twenty three USD" > $23  
• "twenty three Australian dollars" > $23 |
| Convert the words "pounds," "British pounds," or "GDB" after a number to pound sign (£) before the number. | • "twenty three pounds" > £23  
• "I have two thousand pounds" > I have £2,000  
• "five pounds thirty three pence" > £5.33 |
| Convert the words "rupees," "Indian rupees," or "INR" after a number to rupee sign (₹) before the number. | • "twenty three rupees" > ₹23  
• "fifty rupees thirty paise" > ₹50.30 |
| Convert times to numbers. | • "seven a.m. eastern standard time" > 7 a.m. eastern standard time  
• "twelve thirty p.m." > 12:30 p.m. |
| Convert cardinal numbers greater than 10 to numbers. | • "fünfundfünfzig" > 55  
• "vier tausend sechs hundert einundachtzig" > 4681  
• "eine Sache" > "eine Sache" |
| Convert cardinal numbers followed by "million" or "billion" to numerals followed by a word when "million" or "billion" is not followed by a number. | • "zehn Millionen Menschen" > 10 Millionen Menschen  
• "zehn Millionen fünf hundert tausend" > 10,500,000 |
| Convert ordinal numbers greater than 10 to numbers. | • "dreiundzwanzigste" > 23.  
• "vierzigster" > 40.  
• "ich war Erster" > "ich war Erster" |
Alternative Transcriptions

When Amazon Transcribe transcribes an audio file, it returns the transcription with the highest confidence level. You can specify that Amazon Transcribe 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. Alternative transcriptions are only available for the StartTranscriptionJob (p. 297) operation.

You can configure Amazon Transcribe to return alternative transcription using the console or by using the Amazon Transcribe API. To get alternative transcriptions using the API, set the ShowAlternatives field to true and set the MaxAlternatives field to the number of alternatives to return when you call the StartTranscriptionJob (p. 297) operation. You can specify that Amazon Transcribe return up to 10 alternative transcriptions.

You can combine alternative transcriptions with speaker identification and channel identification. Alternative transcriptions are available in all supported languages.

Alternatives are presented at the segment level of the transcription. Segments are defined by natural pauses in speech, such as a change in speaker or pause in the audio. For example, the spoken phrase "It is raining today in Seattle, but not in Portland" is separated into two segments: "It is raining today in Seattle" and "but not in Portland."

Amazon Transcribe returns an overall transcription of your audio file in the response. When you have configured Amazon Transcribe to return alternatives, the overall transcription is built from the segment alternatives with the highest confidence level. Alternative transcriptions are returned in the segments structure in the output JSON. If Amazon Transcribe doesn’t find alternatives, it returns fewer than the number of alternatives specified in the MaxAlternatives field.

The following is the JSON output from Amazon Transcribe. It is the transcription output for this input:

```
Uh, you can just call this number if I don't pick up, just leave a voicemail
```

### Table of Rules and Examples

<table>
<thead>
<tr>
<th>Rule</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fractions are not converted into a numeric format.</td>
<td>• &quot;ein Drittel&quot; &gt; &quot;ein Drittel&quot;</td>
</tr>
<tr>
<td>Convert numbers less than 10 to digits if there are more than one in a row.</td>
<td>• &quot;eins zwei drei&quot; &gt; 123</td>
</tr>
<tr>
<td></td>
<td>• &quot;plus vier neun zwei vier eins&quot; &gt; +49241</td>
</tr>
<tr>
<td>Decimals are indicated by &quot;,,&quot;.</td>
<td>• &quot;zweiundzwanzig komma drei&quot; &gt; 22,3</td>
</tr>
<tr>
<td>Convert the word &quot;percent&quot; after a number to a percent symbol (%).</td>
<td>• &quot;fünf Prozent Hürde&quot; &gt; 5 % Hürde</td>
</tr>
<tr>
<td></td>
<td>• &quot;dreiundzwanzig komma vier Prozent&quot; &gt; 23,4%</td>
</tr>
<tr>
<td>Convert the words &quot;Euro&quot; to a euro sign.</td>
<td>• &quot;ein euro&quot; &gt; 1 €</td>
</tr>
<tr>
<td></td>
<td>• &quot;ein Euro vierzig&quot; &gt; 1,40 €</td>
</tr>
<tr>
<td></td>
<td>• &quot;ein Euro vierzig Cent&quot; &gt; 1,40 €</td>
</tr>
<tr>
<td>Convert times to numbers.</td>
<td>• &quot;vierzehn Uhr fünfzehn&quot; &gt; 14:15 Uhr</td>
</tr>
<tr>
<td>Convert dates to numbers.</td>
<td>• &quot;dritter Dezember neunzehn hundert sechundfünfzig&quot; &gt; 3. Dezember 1956</td>
</tr>
<tr>
<td>Display slashes and dashes.</td>
<td>• &quot;55 Schrägstrich 13&quot; &gt; 55/13</td>
</tr>
<tr>
<td></td>
<td>• &quot;55 Strich 13&quot; &gt; 55-13</td>
</tr>
<tr>
<td>Display numbered paragraphs</td>
<td>• &quot;Paragraf 17&quot; &gt; § 17</td>
</tr>
</tbody>
</table>
and I'll get back to you. Okay. And that's the number. The 1166 number, you mean?

The following is the JSON output with `ShowAlternatives` set to `false`.

```json
{
    "results": {
        "transcripts": [
            "Uh, you can just call this number if I don't pick up and leave a voicemail and I'll get back to you. Okay. And that's the number. The 1166 number, you mean",
        ],
        "items": [
            {
                "start_time": 12.35,
                "end_time": 12.57,
                "alternatives": [
                    {
                        "confidence": 0.9989,
                        "content": "Uh"
                    }
                ],
                "type": "pronunciation"
            },
            Items removed for brevity.
        ]
    }
}
```

The following is the JSON output for the same input with `ShowAlternatives` set to `true` and `MaxAlternatives` set to 2.

```json
{
    "results": {
        "transcripts": [
            "Uh, you can just call this number if I don't pick up and leave a voicemail and I'll get back to you. Okay. And that's the number. The 1166 number, you mean",
        ],
        "items": [
            {
                "start_time": 12.35,
                "end_time": 12.57,
                "alternatives": [
                    {
                        "confidence": 0.9989,
                        "content": "Uh"
                    }
                ],
                "type": "pronunciation"
            },
            Items removed for brevity.
        ],
        "segments": [
            {
                "start_time": 11.84,
                "end_time": 19.665,
                "alternatives": [
                    {
                        "transcript": "Uh, you can just call this number if I don't pick up and leave a voicemail and I'll get back to you."
                    }
                ],
                "type": "pronunciation"
            },
            Items removed for brevity.
        ]
    }
}
```
Transcribing Streaming Audio

Streaming transcription takes a stream of your audio data and transcribes it in real time. Streaming uses HTTP/2 or WebSocket streams so that the results of the transcription are returned to your application while you send more audio to Amazon Transcribe. Use streaming transcription when you want to make the results of live audio transcription available immediately or when you have an audio file that you want to process as it is transcribed.

You can use streaming transcription with the following languages:

- Australian English (en-AU)
Custom Vocabularies

Topics

- Create a Custom Vocabulary Using a List (p. 13)
- Create a Custom Vocabulary Using a Table (p. 13)
- Character Sets for Custom Vocabularies (p. 15)

You can give Amazon Transcribe more information about how to process speech in your input file by creating a custom vocabulary. A custom vocabulary is a list of specific words that you want Amazon Transcribe to recognize in your audio input. These are generally domain-specific words and phrases, words that Amazon Transcribe isn't recognizing, or proper nouns.

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.

Custom vocabularies work best when used to target specific words or phrases. We recommend that you create separate small vocabularies tailored to specific audio recordings instead of creating a single vocabulary with many terms to use for all of your recordings. You can have up to 100 vocabularies in your account. The size limit for a custom vocabulary is 50 Kb.

You specify the custom vocabulary in a text file. You can specify either a list of words in the vocabulary, or a four-column table that gives you more control over the input and output of the words in the custom vocabulary.

For more information about creating a custom vocabulary, see Create a Custom Vocabulary Using a List (p. 13) and Create a Custom Vocabulary Using a Table (p. 13).

To create a custom vocabulary, use the CreateVocabulary (p. 238) operation or the Amazon Transcribe console. After you submit the CreateVocabulary request, Amazon Transcribe processes the vocabulary. To see the processing status of the vocabulary, use the console or the GetVocabulary (p. 268) operation.

Note

If you are uploading the custom vocabulary using the Amazon Transcribe console, you must use vocabulary list instead of a vocabulary table. To use the console to create a custom vocabulary using a vocabulary table, the source file must be in an Amazon S3 bucket.

To use the custom vocabulary, set the VocabularyName field of the Settings field when you call the StartTranscriptionJob (p. 297) operation or choose the vocabulary in the console when you create the transcription job.
Create a Custom Vocabulary Using a List

You can create a custom vocabulary using a list of words or phrases in a text file. You can place each word on its own line, or you can put multiple words on a single line, separating the words or phrases from each other with a comma.

Each entry must contain:

- Fewer than 256 characters, including hyphens
- Only characters from the allowed character set

For valid character sets, see Character Sets for Custom Vocabularies (p. 15).

If an entry in the list is a phrase, separate the words of the phrase with a hyphen. For example, if the phrase is Los Angeles, you would enter it in the file as Los-Angeles.

Enter acronyms or other words whose letters should be pronounced individually as single letters separated by dots, such as A.B.C. or F.B.I.. 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 either upper or lower case letters to enter an acronym. Acronyms are supported in the following languages:

- Dutch
- All English variants
- All French variants
- All German variants
- Hindi
- Indonesian
- Italian
- Malay
- All Portuguese variants
- All Spanish variants
- Turkish

The following example shows an input file with the vocabulary words and phrases on separate lines:

<table>
<thead>
<tr>
<th>Los-Angeles</th>
</tr>
</thead>
<tbody>
<tr>
<td>F.B.I.</td>
</tr>
<tr>
<td>Etienne</td>
</tr>
</tbody>
</table>

The following example shows an input file with the vocabulary words and phrases on a single line, separated by commas:

| Los-Angeles,F.B.I.,Etienne |

Create a Custom Vocabulary Using a Table

You can create a custom vocabulary by creating a table in a text file. Each row in the table is a word or phrase followed by the optional IPA, SoundsLike, and DisplayAs fields. Each field must contain:

- Fewer than 256 characters, including hyphens
Create a Custom Vocabulary Using a Table

- Only characters from the allowed character set

For valid character sets, see Character Sets for Custom Vocabularies (p. 15)

Place each word or phrase in your text file on a separate line. Separate the fields with TAB characters. Save the file with the extension .txt in an Amazon S3 bucket in the same region that you are calling the API.

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 are input files in text format. The examples use spaces to align the columns. Your input files should use TAB characters to separate the columns. Include spaces only in the IPA and DisplayAs columns. If you copy these examples, remove the extra spaces between columns and replace "[TAB]" with a TAB character.

<table>
<thead>
<tr>
<th>Phrase</th>
<th>TABIPA</th>
<th>TABSoundsLike</th>
<th>TABDisplayAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los-Angeles</td>
<td>[TAB]</td>
<td>[TAB]</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>F.B.I.</td>
<td>[TAB]# f b i a#[TAB]</td>
<td>[TAB]FBI</td>
<td></td>
</tr>
<tr>
<td>Etienne</td>
<td>[TAB]eh-tee-en [TAB]</td>
<td>[TAB]</td>
<td></td>
</tr>
</tbody>
</table>

Columns can be entered in any order. The following are also valid structures for the custom vocabulary input file.

<table>
<thead>
<tr>
<th>Phrase</th>
<th>TABSoundsLike</th>
<th>TABIPA</th>
<th>TABDisplayAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los-Angeles</td>
<td>[TAB]</td>
<td>[TAB]</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>F.B.I.</td>
<td>[TAB]# f b i a#[TAB]</td>
<td>[TAB]FBI</td>
<td></td>
</tr>
<tr>
<td>Etienne</td>
<td>[TAB]eh-tee-en [TAB]</td>
<td>[TAB]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DisplayAs</th>
<th>TABSoundsLike</th>
<th>TABIPA</th>
<th>TABPhrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Angeles</td>
<td>[TAB]</td>
<td>[TAB]</td>
<td>Los-Angeles</td>
</tr>
<tr>
<td>FBI</td>
<td>[TAB]# f b i a#[TAB]</td>
<td>[TAB]F.B.I.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[TAB]eh-tee-en [TAB]</td>
<td>[TAB]Etienne</td>
<td></td>
</tr>
</tbody>
</table>

- **Phrase** – The word or phrase that should be recognized.

If the entry is a phrase, separate the words with a hyphen (-). For example, you type **Los Angeles** as **Los-Angeles**.

Enter acronyms or other words whose letters should be pronounced individually as single letters followed by dots, such as **A.B.C.** or **F.B.I.**. 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 either upper- or lower-case letters to enter an acronym. For a list of languages that support acronyms, see Create a Custom Vocabulary Using a List (p. 13).

The **Phrase** field is required. You can use any of the allowed characters for the input language. For the list of allowed characters, see the individual languages. If you do not specify the **DisplayAs** field, Amazon Transcribe uses the contents of the **Phrase** field in the output file.

- **IPA** – To specify the pronunciation of your word or phrase, you can include characters in the International Phonetic Alphabet (IPA) in this field. The **IPA** field 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 **Los-Angeles** as **l # s a n # # l # s**. You would enter the phrase **F.B.I.** as **f b i a#**.
If you don't specify the contents of the IPA field, you must include a blank IPA field. If you specify the IPA field, you can't specify the SoundsLike field.

For a list of allowed IPA characters for a specific language, see the table for individual languages.

- **SoundsLike** – You can break a word or phrase down into smaller pieces and provide a pronunciation for each piece using the standard orthography of the language to mimic the way that the word sounds. For example, in English you can provide pronunciation hints for the phrase *Los-Angeles* like this: *loss-ann-gel-es*. The hint for the word *Etienne* would look like this: *eh-tee-en*. You separate each part of the hint with a hyphen (-).

If you don't specify the SoundsLike field you must include a blank SoundsLike field. If you specify the SoundsLike field you can't specify the IPA field.

You can use any of the allowed characters for the input language. For the list of allowed characters, see the individual languages.

- **DisplayAs** – Defines the how the word or phrase looks when it's output. For example, if the word or phrase is *Los-Angeles*, you can specify the display form as "Los Angeles" so that the hyphen is not present in the output.

If you don't specify the DisplayAs field, Amazon Transcribe uses the Phrase field from the input file in the output.

You can use any UTF-8 character in the DisplayAs field.

## 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.

**Topics**
- Arabic Character Set (p. 16)
- Chinese Character Set (p. 17)
- Dutch Character Set (p. 18)
- English Character Set (p. 19)
- Farsi Character Set (p. 20)
- French Character Set (p. 22)
- German Character Set (p. 23)
- Hebrew Character Set (p. 25)
- Hindi Character Set (p. 26)
- Indonesian Character Set (p. 28)
- Italian Character Set (p. 29)
- Japanese Character Set (p. 31)
- Korean Character Set (p. 32)
- Malay Character Set (p. 33)
- Portuguese Character Set (p. 34)
- Russian Character Set (p. 35)
- Spanish Character Set (p. 37)
- Tamil Character Set (p. 38)
- Telugu Character Set (p. 40)
- Turkish Character Set (p. 42)
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.

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ء</td>
<td>0621</td>
<td>س</td>
<td>0633</td>
</tr>
<tr>
<td>آ</td>
<td>0622</td>
<td>ش</td>
<td>0634</td>
</tr>
<tr>
<td>أ</td>
<td>0623</td>
<td>ص</td>
<td>0635</td>
</tr>
<tr>
<td>و</td>
<td>0624</td>
<td>ض</td>
<td>0636</td>
</tr>
<tr>
<td>ا</td>
<td>0625</td>
<td>ط</td>
<td>0637</td>
</tr>
<tr>
<td>ئ</td>
<td>0626</td>
<td>ط</td>
<td>0638</td>
</tr>
<tr>
<td>ا</td>
<td>0627</td>
<td>ع</td>
<td>0639</td>
</tr>
<tr>
<td>ب</td>
<td>0628</td>
<td>غ</td>
<td>063A</td>
</tr>
<tr>
<td>ة</td>
<td>0629</td>
<td>ف</td>
<td>0641</td>
</tr>
<tr>
<td>ت</td>
<td>062A</td>
<td>ق</td>
<td>0642</td>
</tr>
<tr>
<td>ن</td>
<td>062B</td>
<td>ل</td>
<td>0643</td>
</tr>
<tr>
<td>ج</td>
<td>062C</td>
<td>ل</td>
<td>0644</td>
</tr>
<tr>
<td>ح</td>
<td>062D</td>
<td>م</td>
<td>0645</td>
</tr>
<tr>
<td>خ</td>
<td>062E</td>
<td>ن</td>
<td>0646</td>
</tr>
<tr>
<td>د</td>
<td>062F</td>
<td>ه</td>
<td>0647</td>
</tr>
<tr>
<td>ذ</td>
<td>0630</td>
<td>و</td>
<td>0648</td>
</tr>
<tr>
<td>ر</td>
<td>0631</td>
<td>ي</td>
<td>0649</td>
</tr>
<tr>
<td>ز</td>
<td>0632</td>
<td>ي</td>
<td>064A</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0061</td>
<td>tˤ</td>
<td>0074 02E4</td>
</tr>
<tr>
<td>aː</td>
<td>0061 02D0</td>
<td>u</td>
<td>0075</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>uː</td>
<td>0075 02D0</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>dˤ</td>
<td>0064 02E4</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>x</td>
<td>0078</td>
</tr>
<tr>
<td>Character</td>
<td>Code</td>
<td>Character</td>
<td>Code</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>zˤ</td>
<td>007A 02E4</td>
</tr>
<tr>
<td>iː</td>
<td>0069 02D0</td>
<td>ʔ</td>
<td>00F0</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>δˤ</td>
<td>00F0 02E4</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>硏</td>
<td>0127</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>γ</td>
<td>0263</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>ItemSelectedListener</td>
<td>026A</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ᵗ</td>
<td>026B</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>ʃ</td>
<td>0283</td>
</tr>
<tr>
<td>q</td>
<td>0071</td>
<td>ʒ</td>
<td>0292</td>
</tr>
<tr>
<td>r</td>
<td>0072</td>
<td>ʔ</td>
<td>0294</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>ʃ</td>
<td>0295</td>
</tr>
<tr>
<td>sˤ</td>
<td>0073 02E4</td>
<td>θ</td>
<td>03B8</td>
</tr>
<tr>
<td>t</td>
<td>0074</td>
<td>χ</td>
<td>03C7</td>
</tr>
</tbody>
</table>

**Chinese Character Set**

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

- [chinese-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 Mandarin Chinese using numbers. The following table shows how tone marks are mapped for the word "ma."

<table>
<thead>
<tr>
<th>Tone</th>
<th>Tone Mark</th>
<th>Tone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tone 1</td>
<td>mā</td>
<td>ma1</td>
</tr>
<tr>
<td>Tone 2</td>
<td>má</td>
<td>ma2</td>
</tr>
<tr>
<td>Tone 3</td>
<td>mā</td>
<td>ma3</td>
</tr>
<tr>
<td>Tone 4</td>
<td>mā</td>
<td>ma4</td>
</tr>
</tbody>
</table>
Chinese 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.

<table>
<thead>
<tr>
<th>Phrase</th>
<th>SoundsLike</th>
<th>IPA</th>
<th>DisplayAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ɛɛ</td>
<td>kang1-jian4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ɛɛ</td>
<td>qian3-ze2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ɛɛɛɛ</td>
<td>guo2-fang2-da4-chenh2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ɛɛɛɛɛ</td>
<td>shi4-jie4-bo2-lan3-hui4</td>
<td>ɛɛɛ</td>
<td></td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>à</td>
<td>00E0</td>
<td>î</td>
<td>00EE</td>
</tr>
<tr>
<td>á</td>
<td>00E1</td>
<td>ũ</td>
<td>00EF</td>
</tr>
<tr>
<td>â</td>
<td>00E2</td>
<td>ŋ</td>
<td>00F1</td>
</tr>
<tr>
<td>ä</td>
<td>00E4</td>
<td>ö</td>
<td>00F2</td>
</tr>
<tr>
<td>ç</td>
<td>00E7</td>
<td>ó</td>
<td>00F3</td>
</tr>
<tr>
<td>è</td>
<td>00E8</td>
<td>ô</td>
<td>00F4</td>
</tr>
<tr>
<td>é</td>
<td>00E9</td>
<td>ô</td>
<td>00F6</td>
</tr>
<tr>
<td>ë</td>
<td>00EA</td>
<td>ü</td>
<td>00F9</td>
</tr>
<tr>
<td>ë</td>
<td>00EB</td>
<td>ü</td>
<td>00FA</td>
</tr>
<tr>
<td>ì</td>
<td>00EC</td>
<td>ù</td>
<td>00FB</td>
</tr>
<tr>
<td>í</td>
<td>00ED</td>
<td>ù</td>
<td>00FC</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a:</td>
<td>0061 003A</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>Character</td>
<td>Code</td>
<td>Character</td>
<td>Code</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>bː</td>
<td>0062</td>
<td>øː</td>
<td>00F8</td>
</tr>
<tr>
<td>ð</td>
<td>014B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>øːy</td>
<td>0153</td>
</tr>
<tr>
<td>eː</td>
<td>006S</td>
<td>øːː</td>
<td>015D</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>øː</td>
<td>0251</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>øː</td>
<td>0254</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>øːu</td>
<td>0254</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>øː</td>
<td>0254</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>øː</td>
<td>0259</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>ε</td>
<td>025B</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>εː</td>
<td>025B</td>
</tr>
<tr>
<td>ñ</td>
<td>006E</td>
<td>εi</td>
<td>0269</td>
</tr>
<tr>
<td>oː</td>
<td>006F</td>
<td>h</td>
<td>0266</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>i</td>
<td>026A</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>n</td>
<td>0272</td>
</tr>
<tr>
<td>t</td>
<td>0074</td>
<td>r</td>
<td>027E</td>
</tr>
<tr>
<td>u</td>
<td>0075</td>
<td>f</td>
<td>0283</td>
</tr>
<tr>
<td>v</td>
<td>0076</td>
<td>y</td>
<td>028F</td>
</tr>
<tr>
<td>w</td>
<td>0077</td>
<td>z</td>
<td>0292</td>
</tr>
<tr>
<td>y</td>
<td>0079</td>
<td>Χ</td>
<td>03C7</td>
</tr>
</tbody>
</table>

**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:
Character | Code | Character | Code
---|---|---|---
aʊ | 0061 028A | w | 0077
æ | 0062 | æ | 00E6
d̩ | 0064 0329 | ɗ | 00F0
æ | 0065 026A | η | 014B
f | 0066 | ơ | 0251
g | 0067 | ơ | 0254
h | 0068 | ơ | 0254 026A
i | 0069 | ə | 0259
j | 006A | ɛ | 025B
k̩ | 006C 0329 | İ | 026A
m | 006D | ʃ | 0279
n̩ | 006E 0329 | ʊ | 028A
ou | 006F 028A | η | 028C
p | 0070 | ʍ | 028D
s | 0073 | ʒ | 0292
t | 0074 | ʤ | 02A4
u | 0075 | Џ | 02A7
v | 0076 | θ | 03B8

Farsi Character Set

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
</table>
ء | 0621 | ئ | 0638 |
ی | 0622 | ی | 0639 |
ی | 0623 | غ | 063A |
فو | 0624 | ە | 0641 |
### Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ئ</td>
<td>0626</td>
<td>ق</td>
<td>0642</td>
</tr>
<tr>
<td>ا</td>
<td>0627</td>
<td>ل</td>
<td>0644</td>
</tr>
<tr>
<td>ب</td>
<td>0628</td>
<td>م</td>
<td>0645</td>
</tr>
<tr>
<td>ن</td>
<td>062A</td>
<td>ن</td>
<td>0646</td>
</tr>
<tr>
<td>ن</td>
<td>062B</td>
<td>ه</td>
<td>0647</td>
</tr>
<tr>
<td>خ</td>
<td>062C</td>
<td>و</td>
<td>0648</td>
</tr>
<tr>
<td>ح</td>
<td>062D</td>
<td>ـ</td>
<td>064E</td>
</tr>
<tr>
<td>خ</td>
<td>062E</td>
<td>م</td>
<td>064F</td>
</tr>
<tr>
<td>د</td>
<td>062F</td>
<td>ـ</td>
<td>0650</td>
</tr>
<tr>
<td>ذ</td>
<td>0630</td>
<td>ـ</td>
<td>0651</td>
</tr>
<tr>
<td>ر</td>
<td>0631</td>
<td>پ</td>
<td>067E</td>
</tr>
<tr>
<td>ز</td>
<td>0632</td>
<td>ج</td>
<td>0686</td>
</tr>
<tr>
<td>س</td>
<td>0633</td>
<td>ز</td>
<td>0698</td>
</tr>
<tr>
<td>ش</td>
<td>0634</td>
<td>ی</td>
<td>06A9</td>
</tr>
<tr>
<td>ص</td>
<td>0635</td>
<td>گ</td>
<td>06AF</td>
</tr>
<tr>
<td>ص</td>
<td>0636</td>
<td>ی</td>
<td>06CC</td>
</tr>
<tr>
<td>ط</td>
<td>0637</td>
<td>ۭ</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0062</td>
<td>u</td>
<td>0075</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>æ</td>
<td>00E6</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>ː</td>
<td>0252</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>ɛ</td>
<td>025B</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>r</td>
<td>027E</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>ʁ</td>
<td>0281</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>f</td>
<td>0283</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>ʒ</td>
<td>0292</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>؟</td>
<td>0294</td>
</tr>
<tr>
<td>Character</td>
<td>Code</td>
<td>Character</td>
<td>Code</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>o</td>
<td>006F</td>
<td>?</td>
<td>0294</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>ð</td>
<td>02A4</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>ñ</td>
<td>02A7</td>
</tr>
<tr>
<td>t</td>
<td>0074</td>
<td>χ</td>
<td>03C7</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>À</td>
<td>00C0</td>
<td>à</td>
<td>00E0</td>
</tr>
<tr>
<td>Â</td>
<td>00C2</td>
<td>â</td>
<td>00E2</td>
</tr>
<tr>
<td>Ç</td>
<td>00C7</td>
<td>ç</td>
<td>00E7</td>
</tr>
<tr>
<td>È</td>
<td>00C8</td>
<td>è</td>
<td>00E8</td>
</tr>
<tr>
<td>É</td>
<td>00C9</td>
<td>ë</td>
<td>00E9</td>
</tr>
<tr>
<td>Ê</td>
<td>00CA</td>
<td>ë</td>
<td>00EA</td>
</tr>
<tr>
<td>Ë</td>
<td>00CB</td>
<td>ë</td>
<td>00EB</td>
</tr>
<tr>
<td>Î</td>
<td>00CE</td>
<td>î</td>
<td>00EE</td>
</tr>
<tr>
<td>Ï</td>
<td>00CF</td>
<td>ï</td>
<td>00EF</td>
</tr>
<tr>
<td>Ô</td>
<td>00D4</td>
<td>ô</td>
<td>00F4</td>
</tr>
<tr>
<td>Ö</td>
<td>00D6</td>
<td>ö</td>
<td>00F6</td>
</tr>
<tr>
<td>Û</td>
<td>00D9</td>
<td>û</td>
<td>00F9</td>
</tr>
<tr>
<td>Ù</td>
<td>00DB</td>
<td>ù</td>
<td>00FB</td>
</tr>
<tr>
<td>Ü</td>
<td>00DC</td>
<td>ü</td>
<td>00FC</td>
</tr>
</tbody>
</table>

You can use the following International Phonetic Alphabet in the *IPA* field of your vocabulary file:
### Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0061</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>ā</td>
<td>00E3</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>ŏ</td>
<td>00F5</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>ø</td>
<td>00F8</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>η</td>
<td>014B</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>õe</td>
<td>0153</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>õẽ</td>
<td>0153 0303</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>ε</td>
<td>0250</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>ɔ</td>
<td>0254</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>ə</td>
<td>0259</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ç</td>
<td>0258</td>
</tr>
<tr>
<td>o</td>
<td>006F</td>
<td>ɡ</td>
<td>0261</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>η</td>
<td>0265</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>ɲ</td>
<td>0272</td>
</tr>
<tr>
<td>t</td>
<td>0074</td>
<td>ɕ</td>
<td>0281</td>
</tr>
<tr>
<td>u</td>
<td>0075</td>
<td>ʃ</td>
<td>0283</td>
</tr>
<tr>
<td>v</td>
<td>0076</td>
<td>ʒ</td>
<td>0292</td>
</tr>
<tr>
<td>w</td>
<td>0077</td>
<td>ẽ</td>
<td>1EBD</td>
</tr>
<tr>
<td>y</td>
<td>0079</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ä</td>
<td>00E4</td>
<td>Å</td>
<td>00C4</td>
</tr>
</tbody>
</table>
### Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ö</td>
<td>00F6</td>
<td>Ö</td>
<td>00D6</td>
</tr>
<tr>
<td>ü</td>
<td>00FC</td>
<td>Ü</td>
<td>00DC</td>
</tr>
<tr>
<td>ß</td>
<td>00DF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0061</td>
<td>ts</td>
<td>0074 0073</td>
</tr>
<tr>
<td>at</td>
<td>0061 026A</td>
<td>uː</td>
<td>0075 02D0</td>
</tr>
<tr>
<td>au</td>
<td>0061 028A</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>aː</td>
<td>0061 02D0</td>
<td>x</td>
<td>0078</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>yː</td>
<td>0079 02D0</td>
</tr>
<tr>
<td>eː</td>
<td>0065 02D0</td>
<td>ä</td>
<td>00E3</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>ç</td>
<td>00E7</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>øː</td>
<td>00F8 02D0</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>η</td>
<td>014B</td>
</tr>
<tr>
<td>iː</td>
<td>0069 02D0</td>
<td>æ</td>
<td>0153</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>ŋ</td>
<td>0250 032F</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>ø</td>
<td>0254</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>œv</td>
<td>0254 028F</td>
</tr>
<tr>
<td>l̩</td>
<td>006C 0329</td>
<td>æː</td>
<td>0259</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>ε</td>
<td>025B</td>
</tr>
<tr>
<td>m̩</td>
<td>006D 0329</td>
<td>εː</td>
<td>025B 02D0</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ι</td>
<td>026A</td>
</tr>
<tr>
<td>n̩</td>
<td>006E 0329</td>
<td>ɐ</td>
<td>0281</td>
</tr>
<tr>
<td>oː</td>
<td>006F 02D0</td>
<td>ʃ</td>
<td>0283</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>u</td>
<td>028A</td>
</tr>
<tr>
<td>pf</td>
<td>0070 0066</td>
<td>γ</td>
<td>028F</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>ʧ</td>
<td>02A7</td>
</tr>
<tr>
<td>t</td>
<td>0074</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

24
# Hebrew Character Set

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-</code></td>
<td>002D</td>
<td><code>ט</code></td>
<td>05DD</td>
</tr>
<tr>
<td><code>א</code></td>
<td>05D0</td>
<td><code>מ</code></td>
<td>05DE</td>
</tr>
<tr>
<td><code>ב</code></td>
<td>05D1</td>
<td><code>ר</code></td>
<td>05DF</td>
</tr>
<tr>
<td><code>ג</code></td>
<td>05D2</td>
<td><code>נ</code></td>
<td>05E0</td>
</tr>
<tr>
<td><code>ד</code></td>
<td>05D3</td>
<td><code>ס</code></td>
<td>05E1</td>
</tr>
<tr>
<td><code>ה</code></td>
<td>05D4</td>
<td><code>ע</code></td>
<td>05E2</td>
</tr>
<tr>
<td><code>י</code></td>
<td>05D5</td>
<td><code>ך</code></td>
<td>05E3</td>
</tr>
<tr>
<td><code>ן</code></td>
<td>05D6</td>
<td><code>ס</code></td>
<td>05E4</td>
</tr>
<tr>
<td><code>ס</code></td>
<td>05D7</td>
<td><code>ץ</code></td>
<td>05E5</td>
</tr>
<tr>
<td><code>ך</code></td>
<td>05D8</td>
<td><code>ף</code></td>
<td>05E6</td>
</tr>
<tr>
<td><code>ר</code></td>
<td>05D9</td>
<td><code>ך</code></td>
<td>05E7</td>
</tr>
<tr>
<td><code>ך</code></td>
<td>05DA</td>
<td><code>ר</code></td>
<td>05E8</td>
</tr>
<tr>
<td><code>ש</code></td>
<td>05DB</td>
<td><code>ץ</code></td>
<td>05E9</td>
</tr>
<tr>
<td><code>ת</code></td>
<td>05DC</td>
<td><code>ת</code></td>
<td>05EA</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a</code></td>
<td>0061</td>
<td><code>p</code></td>
<td>0070</td>
</tr>
<tr>
<td><code>b</code></td>
<td>0062</td>
<td><code>s</code></td>
<td>0073</td>
</tr>
<tr>
<td><code>d</code></td>
<td>0064</td>
<td><code>t</code></td>
<td>0074</td>
</tr>
<tr>
<td><code>e</code></td>
<td>0065</td>
<td><code>u</code></td>
<td>0075</td>
</tr>
<tr>
<td><code>f</code></td>
<td>0066</td>
<td><code>v</code></td>
<td>0076</td>
</tr>
<tr>
<td><code>g</code></td>
<td>0067</td>
<td><code>w</code></td>
<td>0077</td>
</tr>
<tr>
<td><code>h</code></td>
<td>0068</td>
<td><code>z</code></td>
<td>007A</td>
</tr>
<tr>
<td><code>i</code></td>
<td>0069</td>
<td><code>η</code></td>
<td>014B</td>
</tr>
<tr>
<td><code>j</code></td>
<td>006A</td>
<td><code>γ</code></td>
<td>0263</td>
</tr>
<tr>
<td><code>k</code></td>
<td>0068</td>
<td><code>f</code></td>
<td>0283</td>
</tr>
</tbody>
</table>
### Hindu Character Set

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-</code></td>
<td>002D</td>
<td><code>ब</code></td>
<td>0925</td>
</tr>
<tr>
<td><code>.</code></td>
<td>002E</td>
<td><code>द</code></td>
<td>0926</td>
</tr>
<tr>
<td>`-'</td>
<td>0901</td>
<td><code>ध</code></td>
<td>0927</td>
</tr>
<tr>
<td>`.'</td>
<td>0902</td>
<td><code>न</code></td>
<td>0928</td>
</tr>
<tr>
<td><code>:</code></td>
<td>0903</td>
<td><code>प</code></td>
<td>0929</td>
</tr>
<tr>
<td><code>अ</code></td>
<td>0905</td>
<td><code>फ</code></td>
<td>092B</td>
</tr>
<tr>
<td><code>आ</code></td>
<td>0906</td>
<td><code>ब</code></td>
<td>092C</td>
</tr>
<tr>
<td><code>इ</code></td>
<td>0907</td>
<td><code>भ</code></td>
<td>092D</td>
</tr>
<tr>
<td><code>ई</code></td>
<td>0908</td>
<td><code>म</code></td>
<td>092E</td>
</tr>
<tr>
<td><code>उ</code></td>
<td>0909</td>
<td><code>व</code></td>
<td>092F</td>
</tr>
<tr>
<td><code>ऋ</code></td>
<td>090A</td>
<td><code>र</code></td>
<td>0930</td>
</tr>
<tr>
<td><code>ॠ</code></td>
<td>090B</td>
<td><code>ल</code></td>
<td>0932</td>
</tr>
<tr>
<td><code>ए</code></td>
<td>090F</td>
<td><code>व</code></td>
<td>0935</td>
</tr>
<tr>
<td><code>ऐ</code></td>
<td>0910</td>
<td><code>श</code></td>
<td>0936</td>
</tr>
<tr>
<td><code>ओ</code></td>
<td>0911</td>
<td><code>ष</code></td>
<td>0937</td>
</tr>
<tr>
<td><code>औ</code></td>
<td>0913</td>
<td><code>स</code></td>
<td>0938</td>
</tr>
<tr>
<td><code>ऋ</code></td>
<td>0914</td>
<td><code>ह</code></td>
<td>0939</td>
</tr>
<tr>
<td><code>क</code></td>
<td>0915</td>
<td><code>ا</code></td>
<td>093E</td>
</tr>
<tr>
<td><code>ख</code></td>
<td>0916</td>
<td><code>ि</code></td>
<td>093F</td>
</tr>
<tr>
<td><code>ग</code></td>
<td>0917</td>
<td><code>ी</code></td>
<td>0940</td>
</tr>
<tr>
<td><code>घ</code></td>
<td>0918</td>
<td><code>़</code></td>
<td>0941</td>
</tr>
<tr>
<td><code>ङ</code></td>
<td>0919</td>
<td><code>ऽ</code></td>
<td>0942</td>
</tr>
</tbody>
</table>
### Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>च</td>
<td>091A</td>
<td>ृ</td>
<td>0943</td>
</tr>
<tr>
<td>छ</td>
<td>091B</td>
<td>ॅ</td>
<td>0945</td>
</tr>
<tr>
<td>ज</td>
<td>091C</td>
<td>े</td>
<td>0947</td>
</tr>
<tr>
<td>झ</td>
<td>091D</td>
<td>ै</td>
<td>0948</td>
</tr>
<tr>
<td>ञ</td>
<td>091E</td>
<td>ँ</td>
<td>0949</td>
</tr>
<tr>
<td>ट</td>
<td>091F</td>
<td>ो</td>
<td>094B</td>
</tr>
<tr>
<td>ठ</td>
<td>0920</td>
<td>ो</td>
<td>094C</td>
</tr>
<tr>
<td>ह</td>
<td>0921</td>
<td>ौ</td>
<td>094D</td>
</tr>
<tr>
<td>ड</td>
<td>0922</td>
<td>ज़</td>
<td>095B</td>
</tr>
<tr>
<td>ढ</td>
<td>0923</td>
<td>ढ</td>
<td>095C</td>
</tr>
<tr>
<td>त</td>
<td>0924</td>
<td>ढ</td>
<td>095D</td>
</tr>
</tbody>
</table>

Amazon Transcribe maps the following characters:

<table>
<thead>
<tr>
<th>Character</th>
<th>Mapped to</th>
</tr>
</thead>
<tbody>
<tr>
<td>ऩ (0929)</td>
<td>न (0928)</td>
</tr>
<tr>
<td>ऱ (0931)</td>
<td>र (0930)</td>
</tr>
<tr>
<td>क़ (0958)</td>
<td>क (0915)</td>
</tr>
<tr>
<td>ख (0959)</td>
<td>ख (0916)</td>
</tr>
<tr>
<td>फ़ (095A)</td>
<td>फ (0917)</td>
</tr>
<tr>
<td>फ (095E)</td>
<td>फ (092B)</td>
</tr>
<tr>
<td>य़ (095F)</td>
<td>य (092F)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>aː</td>
<td>0097 0720</td>
<td>अ</td>
<td>0331</td>
</tr>
<tr>
<td>b</td>
<td>0098</td>
<td>ब</td>
<td>0598</td>
</tr>
<tr>
<td>bʰ</td>
<td>0098 0689</td>
<td>ब़</td>
<td>0596 0720</td>
</tr>
<tr>
<td>d</td>
<td>0100</td>
<td>द</td>
<td>0598 0689</td>
</tr>
<tr>
<td>dʰ</td>
<td>0100 0689</td>
<td>द़</td>
<td>0601</td>
</tr>
<tr>
<td>eː</td>
<td>0101 0720</td>
<td>ए</td>
<td>0603 0720</td>
</tr>
<tr>
<td>f</td>
<td>0102</td>
<td>ग</td>
<td>0609</td>
</tr>
</tbody>
</table>
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:
### Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>0064</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>x</td>
<td>0078</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>y</td>
<td>0079</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>η</td>
<td>014B</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>ɔ</td>
<td>0254</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>ə</td>
<td>0259</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>ɛ</td>
<td>0258</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>g</td>
<td>0261</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>γ</td>
<td>0263</td>
</tr>
<tr>
<td>o</td>
<td>006F</td>
<td>і</td>
<td>026A</td>
</tr>
<tr>
<td>oj</td>
<td>006F 0069 032F</td>
<td>η</td>
<td>0272</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>ʃ</td>
<td>0283</td>
</tr>
<tr>
<td>q</td>
<td>0071</td>
<td>u</td>
<td>028A</td>
</tr>
</tbody>
</table>

#### 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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>À</td>
<td>00C0</td>
<td>à</td>
<td>00E0</td>
</tr>
<tr>
<td>Ä</td>
<td>00C4</td>
<td>ä</td>
<td>00E4</td>
</tr>
<tr>
<td>Ç</td>
<td>00C7</td>
<td>ç</td>
<td>00E7</td>
</tr>
<tr>
<td>È</td>
<td>00C8</td>
<td>è</td>
<td>00E8</td>
</tr>
<tr>
<td>É</td>
<td>00C9</td>
<td>é</td>
<td>00E9</td>
</tr>
<tr>
<td>È</td>
<td>00CA</td>
<td>è</td>
<td>00EA</td>
</tr>
<tr>
<td>Ê</td>
<td>00CB</td>
<td>ë</td>
<td>00EB</td>
</tr>
<tr>
<td>Character</td>
<td>Code</td>
<td>Character</td>
<td>Code</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>ì</td>
<td>00CC</td>
<td>ì</td>
<td>00EC</td>
</tr>
<tr>
<td>Ò</td>
<td>00D2</td>
<td>ò</td>
<td>00F2</td>
</tr>
<tr>
<td>Ù</td>
<td>00D9</td>
<td>ù</td>
<td>00F9</td>
</tr>
<tr>
<td>Ü</td>
<td>00DC</td>
<td>ü</td>
<td>00FC</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0061</td>
<td>ss</td>
<td>0073 0073</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>t</td>
<td>0074</td>
</tr>
<tr>
<td>bb</td>
<td>0062 0062</td>
<td>tt</td>
<td>0074 0074</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>u</td>
<td>0075</td>
</tr>
<tr>
<td>dd</td>
<td>0064 0064</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>wv</td>
<td>0076 0076</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>ff</td>
<td>0066 0066</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>gg</td>
<td>0067 0067</td>
<td>ç</td>
<td>0254</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>e</td>
<td>025B</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>ɡ</td>
<td>0261</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>n</td>
<td>0272</td>
</tr>
<tr>
<td>kk</td>
<td>006B 006B</td>
<td>ɲɲ</td>
<td>0272 0272</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>ʃ</td>
<td>0283</td>
</tr>
<tr>
<td>ll</td>
<td>006C 006C</td>
<td>ʃʃ</td>
<td>0283 0283</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>ʎ</td>
<td>028E</td>
</tr>
<tr>
<td>mm</td>
<td>006D 006D</td>
<td>ʎʎ</td>
<td>028E 028E</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ʣ</td>
<td>02A3</td>
</tr>
<tr>
<td>nn</td>
<td>006E 006E</td>
<td>ʣʣ</td>
<td>02A3 02A3</td>
</tr>
<tr>
<td>o</td>
<td>006F</td>
<td>ʤ</td>
<td>02A4</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>ʤʤ</td>
<td>02A4 02A4</td>
</tr>
<tr>
<td>pp</td>
<td>0070 0070</td>
<td>ʦʦ</td>
<td>02A6</td>
</tr>
<tr>
<td>r</td>
<td>0072</td>
<td>ʦʦ</td>
<td>02A6 02A6</td>
</tr>
<tr>
<td>rr</td>
<td>0072 0072</td>
<td>ʧʧ</td>
<td>02A7</td>
</tr>
<tr>
<td>Character</td>
<td>Code</td>
<td>Character</td>
<td>Code</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>ʧʧ</td>
<td>02A7 02A7</td>
</tr>
</tbody>
</table>

**Japanese Character Set**

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

- *japanese-character-set.txt*

Amazon Transcribe supports Romaji characters in the **SoundsLike** field. You can use the following lower-case characters:

- a - k
- m - p
- r - w
- y - z

Represent long vowels by doubling the vowel:

<table>
<thead>
<tr>
<th>Vowel</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ā</td>
<td>aa</td>
</tr>
<tr>
<td>ē</td>
<td>ee</td>
</tr>
<tr>
<td>Ĭ</td>
<td>ii</td>
</tr>
<tr>
<td>ō</td>
<td>oo</td>
</tr>
<tr>
<td>ŭ</td>
<td>uu</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0061</td>
<td>p</td>
<td>0070</td>
</tr>
<tr>
<td>aː</td>
<td>0061 02D0</td>
<td>s</td>
<td>0073</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>t</td>
<td>0074</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>ts</td>
<td>0074 0073</td>
</tr>
<tr>
<td>dz</td>
<td>0064 007A</td>
<td>tɕ</td>
<td>0074 0255</td>
</tr>
<tr>
<td>dz</td>
<td>0064 0291</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>eː</td>
<td>0065 02D0</td>
<td>ç</td>
<td>00E7</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>η</td>
<td>0148</td>
</tr>
</tbody>
</table>
### Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>0068</td>
<td>ɕ</td>
<td>0255</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>ɯ</td>
<td>026F</td>
</tr>
<tr>
<td>iː</td>
<td>0069 02D0</td>
<td>ɯː</td>
<td>026F 02D0</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>ɲ</td>
<td>0274</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>ɸ</td>
<td>0278</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>r</td>
<td>027E</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ʐ</td>
<td>0291</td>
</tr>
<tr>
<td>o</td>
<td>006F</td>
<td>ʔ</td>
<td>0294</td>
</tr>
<tr>
<td>oː</td>
<td>006F 02D0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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](https://en.wikipedia.org/wiki/Hangul) on Wikipedia.

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>00061</td>
<td>s#</td>
<td>0073 0348</td>
</tr>
<tr>
<td>e</td>
<td>00065</td>
<td>t</td>
<td>0074</td>
</tr>
<tr>
<td>h</td>
<td>00068</td>
<td>tɕ</td>
<td>0074 0255</td>
</tr>
<tr>
<td>i</td>
<td>00069</td>
<td>tɕʰ</td>
<td>0074 0255 02B0</td>
</tr>
<tr>
<td>je</td>
<td>006A 0065</td>
<td>tʰ</td>
<td>0074 02B0</td>
</tr>
<tr>
<td>jo</td>
<td>006A 006F</td>
<td>t#</td>
<td>0074 0348</td>
</tr>
<tr>
<td>ju</td>
<td>006A 0075</td>
<td>t#ɕ</td>
<td>0074 0348 0255</td>
</tr>
<tr>
<td>jc</td>
<td>006A 025B</td>
<td>u</td>
<td>0075</td>
</tr>
<tr>
<td>ja</td>
<td>006A 0061</td>
<td>we</td>
<td>0077 0065</td>
</tr>
<tr>
<td>ja</td>
<td>006A 0061</td>
<td>wi</td>
<td>0077 0069</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>we</td>
<td>0077 025B</td>
</tr>
<tr>
<td>kʰ</td>
<td>006B 02B0</td>
<td>wa</td>
<td>0077 028C</td>
</tr>
<tr>
<td>k#</td>
<td>006B 0348</td>
<td>wa</td>
<td>0077 0061</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>ɶ</td>
<td>00F8</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>η</td>
<td>0014B</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ε</td>
<td>0025B</td>
</tr>
</tbody>
</table>
### 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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0046</td>
<td>r</td>
<td>0072</td>
</tr>
<tr>
<td>a</td>
<td>0061</td>
<td>s</td>
<td>0073</td>
</tr>
<tr>
<td>ai</td>
<td>0061 0069</td>
<td>t</td>
<td>0074</td>
</tr>
<tr>
<td>au</td>
<td>0061 0075</td>
<td>tʃ</td>
<td>0074 0283</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>dʒ</td>
<td>0064 0292</td>
<td>x</td>
<td>0078</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>y</td>
<td>0079</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>η</td>
<td>014B</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>ɔ</td>
<td>0254</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>ə</td>
<td>0259</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>ε</td>
<td>025B</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>g</td>
<td>0261</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>γ</td>
<td>0263</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ì</td>
<td>026A</td>
</tr>
<tr>
<td>o</td>
<td>006F</td>
<td>ɲ</td>
<td>0272</td>
</tr>
</tbody>
</table>
### 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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>À</td>
<td>00C0</td>
<td>à</td>
<td>00E0</td>
</tr>
<tr>
<td>Á</td>
<td>00C1</td>
<td>á</td>
<td>00E1</td>
</tr>
<tr>
<td>Â</td>
<td>00C2</td>
<td>â</td>
<td>00E2</td>
</tr>
<tr>
<td>Ä</td>
<td>00C3</td>
<td>ä</td>
<td>00E3</td>
</tr>
<tr>
<td>Å</td>
<td>00C4</td>
<td>ä</td>
<td>00E4</td>
</tr>
<tr>
<td>Ç</td>
<td>00C7</td>
<td>ç</td>
<td>00E7</td>
</tr>
<tr>
<td>È</td>
<td>00C8</td>
<td>è</td>
<td>00E8</td>
</tr>
<tr>
<td>É</td>
<td>00C9</td>
<td>é</td>
<td>00E9</td>
</tr>
<tr>
<td>Ê</td>
<td>00CA</td>
<td>ê</td>
<td>00EA</td>
</tr>
<tr>
<td>Ë</td>
<td>00CB</td>
<td>ë</td>
<td>00EB</td>
</tr>
<tr>
<td>Í</td>
<td>00CD</td>
<td>í</td>
<td>00ED</td>
</tr>
<tr>
<td>Ñ</td>
<td>00D1</td>
<td>ñ</td>
<td>00F1</td>
</tr>
<tr>
<td>Ô</td>
<td>00D3</td>
<td>ó</td>
<td>00F3</td>
</tr>
<tr>
<td>Ô</td>
<td>00D4</td>
<td>ô</td>
<td>00F4</td>
</tr>
<tr>
<td>Ö</td>
<td>00D5</td>
<td>ö</td>
<td>00F5</td>
</tr>
<tr>
<td>Ò</td>
<td>00D6</td>
<td>ò</td>
<td>00F6</td>
</tr>
<tr>
<td>Ú</td>
<td>00DA</td>
<td>ú</td>
<td>00FA</td>
</tr>
<tr>
<td>Ë</td>
<td>00DC</td>
<td>ù</td>
<td>00FC</td>
</tr>
</tbody>
</table>
You can use the following International Phonetic Alphabet characters in the IPA field of your input file:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0061</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>w~</td>
<td>0077 0303</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>ö</td>
<td>00F5</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>ï</td>
<td>00129</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>ū</td>
<td>00169</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>ĕ</td>
<td>0250 0303</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>ɔ</td>
<td>0254</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>ɛ</td>
<td>025B</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>ŋ</td>
<td>0272</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>r</td>
<td>027E</td>
</tr>
<tr>
<td>o</td>
<td>006F</td>
<td>ŋ</td>
<td>0281</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>ʃ</td>
<td>0283</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>ʒ</td>
<td>028E</td>
</tr>
<tr>
<td>t</td>
<td>0074</td>
<td>ʒ</td>
<td>0292</td>
</tr>
<tr>
<td>tf</td>
<td>0074 0283</td>
<td>ðζ</td>
<td>02A4</td>
</tr>
<tr>
<td>u</td>
<td>0075</td>
<td>ɛ</td>
<td>1EBD</td>
</tr>
</tbody>
</table>

**Russian Character Set**

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>'</td>
<td>0027</td>
<td>ы</td>
<td>043F</td>
</tr>
<tr>
<td>-</td>
<td>002D</td>
<td>р</td>
<td>0440</td>
</tr>
<tr>
<td>.</td>
<td>002E</td>
<td>с</td>
<td>0441</td>
</tr>
<tr>
<td>a</td>
<td>0430</td>
<td>т</td>
<td>0442</td>
</tr>
<tr>
<td>6</td>
<td>0431</td>
<td>ь</td>
<td>0443</td>
</tr>
<tr>
<td>в</td>
<td>0432</td>
<td>ф</td>
<td>0444</td>
</tr>
<tr>
<td>г</td>
<td>0433</td>
<td>х</td>
<td>0445</td>
</tr>
<tr>
<td>Character</td>
<td>Code</td>
<td>Character</td>
<td>Code</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>д</td>
<td>0434</td>
<td>ц</td>
<td>0446</td>
</tr>
<tr>
<td>е</td>
<td>0435</td>
<td>ч</td>
<td>0447</td>
</tr>
<tr>
<td>ж</td>
<td>0436</td>
<td>ш</td>
<td>0448</td>
</tr>
<tr>
<td>з</td>
<td>0437</td>
<td>щ</td>
<td>0449</td>
</tr>
<tr>
<td>и</td>
<td>0438</td>
<td>ъ</td>
<td>044A</td>
</tr>
<tr>
<td>й</td>
<td>0439</td>
<td>ё</td>
<td>044B</td>
</tr>
<tr>
<td>к</td>
<td>043A</td>
<td>ъ</td>
<td>044C</td>
</tr>
<tr>
<td>л</td>
<td>043B</td>
<td>э</td>
<td>044D</td>
</tr>
<tr>
<td>м</td>
<td>043C</td>
<td>ю</td>
<td>044E</td>
</tr>
<tr>
<td>н</td>
<td>043D</td>
<td>я</td>
<td>044F</td>
</tr>
<tr>
<td>о</td>
<td>043E</td>
<td>ё</td>
<td>0451</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0062</td>
<td>t</td>
<td>0074</td>
</tr>
<tr>
<td>bʲ</td>
<td>0062 02B2</td>
<td>tʃ</td>
<td>0074 0283</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>tʲ</td>
<td>0074 02B2</td>
</tr>
<tr>
<td>dʲ</td>
<td>0064 02B2</td>
<td>u</td>
<td>0075</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>fʲ</td>
<td>0066 02B2</td>
<td>vʲ</td>
<td>0076 02B2</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>x</td>
<td>0078</td>
</tr>
<tr>
<td>gʲ</td>
<td>067 02B2</td>
<td>xʲ</td>
<td>0078 02B2</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>zʲ</td>
<td>007A 02B2</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>æ</td>
<td>00E6</td>
</tr>
<tr>
<td>kʲ</td>
<td>006B 02B2</td>
<td>ə</td>
<td>0259</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>ɛ</td>
<td>025B</td>
</tr>
<tr>
<td>lʲ</td>
<td>006C 02B2</td>
<td>ɨ</td>
<td>0268</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>ʃ</td>
<td>0283</td>
</tr>
<tr>
<td>mʲ</td>
<td>006D 02B2</td>
<td>ʃʲ</td>
<td>0283 02B2</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ʊ</td>
<td>028A</td>
</tr>
<tr>
<td>Character</td>
<td>Code</td>
<td>Character</td>
<td>Code</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>nʲ</td>
<td>006E 02B2</td>
<td>Σ</td>
<td>028C</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>ζ</td>
<td>0292</td>
</tr>
<tr>
<td>pʲ</td>
<td>0070 02B2</td>
<td>'i</td>
<td>02C8 0069</td>
</tr>
<tr>
<td>r</td>
<td>0072</td>
<td>'o</td>
<td>02C8 006F</td>
</tr>
<tr>
<td>rʲ</td>
<td>0072 02B2</td>
<td>'v</td>
<td>02C8 0075</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>'ɛ</td>
<td>02C8 025B</td>
</tr>
<tr>
<td>sʲ</td>
<td>0073 02B2</td>
<td>'i</td>
<td>02C8 0268</td>
</tr>
<tr>
<td>ts</td>
<td>0074 0073</td>
<td>'a</td>
<td>02C8 0061</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Á</td>
<td>00C1</td>
<td>á</td>
<td>00E1</td>
</tr>
<tr>
<td>É</td>
<td>00C9</td>
<td>é</td>
<td>00E9</td>
</tr>
<tr>
<td>í</td>
<td>00CD</td>
<td>ì</td>
<td>00ED</td>
</tr>
<tr>
<td>Ó</td>
<td>00D3</td>
<td>ó</td>
<td>0XF3</td>
</tr>
<tr>
<td>Ú</td>
<td>00DA</td>
<td>Ú</td>
<td>00FA</td>
</tr>
<tr>
<td>Ñ</td>
<td>00D1</td>
<td>ñ</td>
<td>0XF1</td>
</tr>
<tr>
<td>ü</td>
<td>00FC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0061</td>
<td>r</td>
<td>0072</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>s</td>
<td>0073</td>
</tr>
<tr>
<td>Character</td>
<td>Code</td>
<td>Character</td>
<td>Code</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>t</td>
<td>0074</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>u</td>
<td>0075</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>x</td>
<td>0078</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>η</td>
<td>014B</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>η</td>
<td>0272</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>r</td>
<td>027E</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>j</td>
<td>0283</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>i</td>
<td>029D</td>
</tr>
<tr>
<td>o</td>
<td>006F</td>
<td>ⱪ</td>
<td>02A7</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>θ</td>
<td>03B8</td>
</tr>
</tbody>
</table>

**Tamil Character Set**

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>அ</td>
<td>0B85</td>
<td>ட</td>
<td>0BB0</td>
</tr>
<tr>
<td>ஆ</td>
<td>0B86</td>
<td>த</td>
<td>0BB2</td>
</tr>
<tr>
<td>இ</td>
<td>0B87</td>
<td>஥</td>
<td>0BB5</td>
</tr>
<tr>
<td>ஈ</td>
<td>0B88</td>
<td>ஞ</td>
<td>0BB4</td>
</tr>
<tr>
<td>உ</td>
<td>0B89</td>
<td>டூ</td>
<td>0BB3</td>
</tr>
<tr>
<td>ஊ</td>
<td>0B8A</td>
<td>டௌ</td>
<td>0BB1</td>
</tr>
<tr>
<td>஋</td>
<td>0B8E</td>
<td>டொ</td>
<td>0BA9</td>
</tr>
<tr>
<td>஌</td>
<td>0B8F</td>
<td>டோ</td>
<td>0B9C</td>
</tr>
<tr>
<td>஍</td>
<td>0B90</td>
<td>டௌ</td>
<td>0BB6</td>
</tr>
<tr>
<td>எ</td>
<td>0B92</td>
<td>ட்</td>
<td>0BB7</td>
</tr>
<tr>
<td>ஏ</td>
<td>0B93</td>
<td>ட்</td>
<td>0BB8</td>
</tr>
<tr>
<td>ஐ</td>
<td>0B94</td>
<td>ட௎</td>
<td>0BB9</td>
</tr>
<tr>
<td>஑</td>
<td>0B83</td>
<td>டை</td>
<td>0BCD</td>
</tr>
</tbody>
</table>
### Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>க</td>
<td>0B95</td>
<td>ந</td>
<td>0BBF</td>
</tr>
<tr>
<td>ம</td>
<td>0B9A</td>
<td>ப</td>
<td>0BCA</td>
</tr>
<tr>
<td>ல</td>
<td>0B99</td>
<td>ர</td>
<td>0BCB</td>
</tr>
<tr>
<td>ல</td>
<td>0B9E</td>
<td>ல</td>
<td>0BCC</td>
</tr>
<tr>
<td>ல</td>
<td>0B9F</td>
<td>த</td>
<td>0BCC</td>
</tr>
<tr>
<td>ல</td>
<td>0BA3</td>
<td>த</td>
<td>0BCA</td>
</tr>
<tr>
<td>ல</td>
<td>0BA4</td>
<td>த</td>
<td>0BCA</td>
</tr>
<tr>
<td>ல</td>
<td>0BA8</td>
<td>த</td>
<td>0BCC</td>
</tr>
<tr>
<td>ல</td>
<td>0BA9</td>
<td>த</td>
<td>0BCC</td>
</tr>
<tr>
<td>ல</td>
<td>0BAE</td>
<td>த</td>
<td>0BCC</td>
</tr>
<tr>
<td>ல</td>
<td>0BAF</td>
<td>த</td>
<td>0BCC</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0061</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>aː</td>
<td>0061 02D0</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>æ</td>
<td>00E6</td>
</tr>
<tr>
<td>dʒ</td>
<td>0064 0292</td>
<td>ˈd</td>
<td>00F0</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>ɛ</td>
<td>014B</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>α</td>
<td>0251</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>ɔ</td>
<td>0254</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>ə</td>
<td>0259</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>ɛ</td>
<td>025B</td>
</tr>
<tr>
<td>iː</td>
<td>0069 02D0</td>
<td>ɡ</td>
<td>0261</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>ɪ</td>
<td>026A</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>ɫ</td>
<td>026D</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>ɲ</td>
<td>0272</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>ɳ</td>
<td>0273</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ɭ</td>
<td>0279</td>
</tr>
<tr>
<td>η</td>
<td>006E 032A</td>
<td>ɭ</td>
<td>0279</td>
</tr>
</tbody>
</table>
Character | Code | Character | Code
---|---|---|---
o | 006F | ɹ̩ | 0279 0329
oː | 006F 02D0 | r | 027E
p | 0070 | ş | 0282
r | 0072 | ɾ | 0283
s | 0073 | ɻ | 0288
t | 0074 | u | 028A
tː | 0074 032A | ū | 028B
tʃ | 0074 0283 | ʌ | 028C
u | 0075 | ʒ | 0292
uː | 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 | ఠ | 0C24
c | 0C01 | డ | 0C25
o | 0C02 | ౢ | 0C26
s | 0C03 | ౣ | 0C27
 Cosby | 0C05 | ౥ | 0C28
 Cosby | 0C06 | ౦ | 0C2A
 Cosby | 0C07 | ౧ | 0C2B
 Cosby | 0C08 | ౡ | 0C2C
 Cosby | 0C09 | ౢ | 0C2D
 Cosby | 0C0A | ౣ | 0C2E
 Cosby | 0C0B | ౤ | 0C2F
 Cosby | 0C0C | ౥ | 0C30
 Cosby | 0C0E | ౦ | 0C31
 Cosby | 0C0F | ౧ | 0C32
 Cosby | 0C10 | ౡ | 0C33
 Cosby | 0C12 | ౣ | 0C35
### Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ఓ</td>
<td>0C13</td>
<td>ప</td>
<td>0C36</td>
</tr>
<tr>
<td>చే</td>
<td>0C14</td>
<td>ఱ</td>
<td>0C37</td>
</tr>
<tr>
<td>టే</td>
<td>0C15</td>
<td>న</td>
<td>0C38</td>
</tr>
<tr>
<td>తా</td>
<td>0C16</td>
<td>పే</td>
<td>0C39</td>
</tr>
<tr>
<td>హే</td>
<td>0C17</td>
<td>మ</td>
<td>0C3E</td>
</tr>
<tr>
<td>యే</td>
<td>0C18</td>
<td>యే</td>
<td>0C3F</td>
</tr>
<tr>
<td>యె</td>
<td>0C19</td>
<td>యొ</td>
<td>0C40</td>
</tr>
<tr>
<td>లే</td>
<td>0C1A</td>
<td>లే</td>
<td>0C41</td>
</tr>
<tr>
<td>లే</td>
<td>0C1B</td>
<td>లే</td>
<td>0C42</td>
</tr>
<tr>
<td>లే</td>
<td>0C1C</td>
<td>లే</td>
<td>0C43</td>
</tr>
<tr>
<td>లే</td>
<td>0C1D</td>
<td>లే</td>
<td>0C44</td>
</tr>
<tr>
<td>లే</td>
<td>0C1E</td>
<td>లే</td>
<td>0C47</td>
</tr>
<tr>
<td>లే</td>
<td>0C1F</td>
<td>లే</td>
<td>0C48</td>
</tr>
<tr>
<td>లే</td>
<td>0C20</td>
<td>లే</td>
<td>0C4A</td>
</tr>
<tr>
<td>లే</td>
<td>0C21</td>
<td>లే</td>
<td>0C4B</td>
</tr>
<tr>
<td>లే</td>
<td>0C22</td>
<td>లే</td>
<td>0C4C</td>
</tr>
<tr>
<td>లే</td>
<td>0C23</td>
<td>లే</td>
<td>0C4D</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>d̪</td>
<td>0064 032A</td>
<td>ð</td>
<td>00F0</td>
</tr>
<tr>
<td>d̪̤</td>
<td>0064 032A 0324</td>
<td>ŋ</td>
<td>014B</td>
</tr>
<tr>
<td>dʒ</td>
<td>0064 0292</td>
<td>ɑ</td>
<td>0251</td>
</tr>
<tr>
<td>dʒ̤</td>
<td>0064 0292 0324</td>
<td>ɔ</td>
<td>0254</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>Ʉ</td>
<td>0256</td>
</tr>
<tr>
<td>eː</td>
<td>0065 02D0</td>
<td>Ʉ</td>
<td>0256 0324</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>ə</td>
<td>0259</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>ɛ</td>
<td>025B</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>ɡ</td>
<td>0261</td>
</tr>
<tr>
<td>iz</td>
<td>0069 0290</td>
<td>ɡ</td>
<td>0261 0324</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>Ʉ</td>
<td>026A</td>
</tr>
</tbody>
</table>
### Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>006B</td>
<td>l</td>
<td>006C</td>
</tr>
<tr>
<td>kʰ</td>
<td>006B 02B0</td>
<td>n</td>
<td>0272</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>n</td>
<td>0279</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>o</td>
<td>027D</td>
</tr>
<tr>
<td>o:</td>
<td>006F 02D0</td>
<td>ş</td>
<td>0282</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>ŋ</td>
<td>0283</td>
</tr>
<tr>
<td>pʰ</td>
<td>0070 02B0</td>
<td>t</td>
<td>0288</td>
</tr>
<tr>
<td>r</td>
<td>0072</td>
<td>ɹ̩</td>
<td>028A</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>ʊ</td>
<td>028B</td>
</tr>
<tr>
<td>t</td>
<td>0074</td>
<td>ʋ</td>
<td>028C</td>
</tr>
<tr>
<td>t̪</td>
<td>0074 02B0</td>
<td>ʒ</td>
<td>0292</td>
</tr>
<tr>
<td>tʰ</td>
<td>0074 02B0</td>
<td>ʒ</td>
<td>0292</td>
</tr>
<tr>
<td>t̪ʰ</td>
<td>0074 02B0</td>
<td>ʒ</td>
<td>0292</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ç</td>
<td>00C7</td>
<td>ö</td>
<td>00F6</td>
</tr>
<tr>
<td>Ö</td>
<td>00D6</td>
<td>ū</td>
<td>00FB</td>
</tr>
<tr>
<td>Ü</td>
<td>00DC</td>
<td>ü</td>
<td>00FC</td>
</tr>
<tr>
<td>â</td>
<td>00E2</td>
<td>ğ</td>
<td>011E</td>
</tr>
<tr>
<td>ä</td>
<td>00E4</td>
<td>ğ</td>
<td>011F</td>
</tr>
</tbody>
</table>
## Character Sets for Custom Vocabularies

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ç</td>
<td>00E7</td>
<td>ï</td>
<td>0130</td>
</tr>
<tr>
<td>è</td>
<td>00E8</td>
<td>i</td>
<td>0131</td>
</tr>
<tr>
<td>é</td>
<td>00E9</td>
<td>Ş</td>
<td>015E</td>
</tr>
<tr>
<td>ê</td>
<td>00EA</td>
<td>ş</td>
<td>015F</td>
</tr>
<tr>
<td>i</td>
<td>00ED</td>
<td>ū</td>
<td>0161</td>
</tr>
<tr>
<td>î</td>
<td>00EE</td>
<td>ź</td>
<td>017E</td>
</tr>
<tr>
<td>ó</td>
<td>00F3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0061</td>
<td>u</td>
<td>0075</td>
</tr>
<tr>
<td>aː</td>
<td>0061 02D0</td>
<td>uː</td>
<td>0075 02D0</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>v</td>
<td>0076</td>
</tr>
<tr>
<td>c</td>
<td>0063</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>y</td>
<td>0079</td>
</tr>
<tr>
<td>e</td>
<td>0065</td>
<td>yː</td>
<td>0079 02D0</td>
</tr>
<tr>
<td>eː</td>
<td>0065 02D0</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>ø</td>
<td>00F8</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>øː</td>
<td>00F8 02D0</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>η</td>
<td>014B</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>j</td>
<td>025F</td>
</tr>
<tr>
<td>iː</td>
<td>0069 02D0</td>
<td>γ</td>
<td>0263</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>ı</td>
<td>026B</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>w</td>
<td>026F</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>wː</td>
<td>026F 02D0</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>r</td>
<td>027E</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ḟ</td>
<td>0283</td>
</tr>
<tr>
<td>o</td>
<td>006F</td>
<td>ʒ</td>
<td>0292</td>
</tr>
<tr>
<td>oː</td>
<td>006F 02D0</td>
<td>ئ</td>
<td>0294</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>dʒ</td>
<td>02A4</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>ʧ</td>
<td>02A7</td>
</tr>
</tbody>
</table>
Amazon Transcribe Developer Guide
Automatic Content Redaction

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>0074</td>
</tr>
</tbody>
</table>

## Automatic Content Redaction

Amazon Transcribe's automatic content redaction feature automatically redacts sensitive personally identifiable information (PII) from your transcription results. It replaces each identified instance of PII with a `[PII]` tag in the transcript. You can use this feature to protect privacy and comply with local laws and regulations. Automatic content redaction enables you to easily review and share transcripts to improve the customer service experience, coach agents, and discover new business opportunities while protecting sensitive personal information. You can use this feature for source audio in US English (en-US) with batch API calls.

Personally Identifiable Information includes:

<table>
<thead>
<tr>
<th>PII Entity</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank Account Number</td>
<td>A number that uniquely identifies a bank account.</td>
</tr>
<tr>
<td>Bank Routing Number</td>
<td>A number that identifies the location of a bank account.</td>
</tr>
<tr>
<td>Credit Card Number or Debit Card Number</td>
<td>A value that uniquely defines a payment card issued by a bank.</td>
</tr>
<tr>
<td>Credit Card or Debit Card CVV Code</td>
<td>A 3 digit or 4 digit security code on each credit card.</td>
</tr>
<tr>
<td>Credit Card or Debit Card Expiration Date</td>
<td>The month and year a card expires.</td>
</tr>
<tr>
<td>Debit Card PIN or Credit Card PIN</td>
<td>A security code issued by a bank or credit union. This number is used for bank accounts and payment cards.</td>
</tr>
<tr>
<td>Email address</td>
<td>The unique identifier of an email box where messages are delivered.</td>
</tr>
<tr>
<td>US Mailing address</td>
<td>The U.S. mailing address of an individual.</td>
</tr>
<tr>
<td>Name</td>
<td>The first name and last name of a person.</td>
</tr>
<tr>
<td>US phone number</td>
<td>A 10-digit phone number within the United States.</td>
</tr>
<tr>
<td>Social Security Number</td>
<td>A 9 digit number or the last 4 digits of that number. Issued to U.S. citizens, permanent residents, and temporary residents with employment.</td>
</tr>
</tbody>
</table>

For each transcription job with automatic content redaction enabled, you generate either:

- Only the redacted transcript.
- Both the redacted transcript and the unredacted transcript.
Both redacted and unredacted transcripts are stored in the same output S3 bucket. Amazon Transcribe stores them in either a bucket you specify or in the default S3 bucket managed by the service.

To enable content redaction, use the console or the API. In the console, you enable **Automatic content redaction** in the **Content removal** section.

To enable content redaction using the API, complete the request parameters of the `ContentRedaction` object in the `StartTranscriptionJob` operation. See the request syntax for the `StartTranscriptionJob (p. 297)` action for more information. To see if content redaction has been enabled for a particular transcription job, use `GetTranscriptionJob (p. 265)`. To see which jobs have content redaction enabled, use `ListTranscriptionJobs (p. 283)`.

A redacted transcript has sensitive PII replaced with the `[PII]` tag, shown in the first truncated JSON output on this page. Because Amazon Transcribe has redacted this transcript, the `isRedacted` field of this JSON output is `true`. Each JSON output of a transcription job has a `results` section that contains the transcription results. Every word, number, punctuation mark, or redaction has a confidence value.

Transcription jobs using automatic 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 transcript output below, 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 transcript output below, the confidence of `0.9999` indicates that Amazon Transcribe is 99.99 percent confident that the entity it redacted in the transcript is PII.

The following is the redacted JSON output:

```json
{
  "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 Security number [PII]. My credit card number is [PII] and my C V V code is [PII]. I hope that Amazon Transcribe is doing a good job at redacting that personal information away. Let's check."
      },
      Items removed for brevity
      {
        "start_time": "5.56",
        "end_time": "6.25",
        "alternatives": [
          {
            "content": "[PII]",
            "redactions": [ ]
          }
        ]
      }
    ],
    "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": [ ]
          }
        ]
      }
    ]
  }
}`
```
If you generate an additional unredacted transcript, its JSON output looks similar to the output of a transcription job with content redaction disabled. The only difference is the additional `isRedacted` field being set to `false`.

The following is the unredacted JSON output:

```json
{
    "jobName": "job id",
    "accountId": "account id",
    "isRedacted": false,
    "results": {
        "transcripts": [
            {
                "transcript": "Good morning, everybody. My name is Mike, and today I feel like sharing a whole lot of personal information with you. Let's start with my Social Security number 000000000. My credit card number 5555555555555555 is and my C V V code is 000. I hope that Amazon Transcribe is doing a good job at redacting that personal information away. Let's check."
            },
            Items removed for brevity
        ],
        "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": "1.0",
                        "content": "Mike"
                    }
                ],
                "type": "pronunciation"
            },
            Items removed for brevity
        ],
        "status": "COMPLETED"
    }
}
```
Amazon Transcribe throws an error message if you use automatic content redaction in an unsupported region. Similarly, you receive an error message if you use content redaction with an unsupported language.

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 100 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 console, or you can set the AllowDeferredExecution field in the JobExecutionSettings parameter to true when you call the StartTranscriptionJob operation.

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 operation. When a job is queued, the Status field of the TranscriptionJob object returned by the StartTranscriptionJob operation 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 operation 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 need to provide Amazon Transcribe with a data access role that permits access to your audio file to be transcribed. You can choose the data access role using the console, or you use the DataAccessRoleArn field of the JobExecutionSettings parameter of the StartTranscriptionJob operation to specify the role to use.

The role policies that you use depends on where you are storing your input files, where you are storing your output files, and whether or not you are encrypting the output with an AWS KMS customer master key (CMK). 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 AWS KMS CMK 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.

```json
{

```
Input Bucket Policy

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

```json
{
   "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.

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

AWS KMS Key Policy for Input Buckets

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

```json
{
   "Version": "2012-10-17",
   "Statement": {
```

48
"Effect": "Allow",
"Action": [
  "kms:Decrypt"
],
"Resource": [
  "arn:aws:kms:::input-bucket-cmk-name"
]
}

AWS KMS Key Policy for Output Buckets

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

{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey"
    ],
    "Resource": [
      "arn:aws:kms:::output-bucket-cmk-name"
    ]
  }
}
Getting Started with Amazon Transcribe

To get started using Amazon Transcribe, set up an AWS account and create an AWS Identity and Access Management (IAM) user. To use the AWS Command Line Interface (AWS CLI), download and configure it.

Topics
- Step 1: Set up an AWS Account and Create an Administrator User (p. 50)
- Step 2: Set up the AWS Command Line Interface (AWS CLI) (p. 51)
- Step 3: Getting Started Using the Console (p. 52)
- Step 4: Getting Started Using the API (p. 56)
- Step 5: Getting Started With Streaming Audio (p. 60)

Step 1: Set up an AWS Account and Create an Administrator User

Before you use Amazon Transcribe for the first time, complete the following tasks:

1. Sign up for AWS (p. 50)
2. Create an IAM User (p. 50)

Sign up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all AWS services, including Amazon Transcribe. You are charged only for the services that you use.

With Amazon Transcribe, you pay only for the resources that you use. If you are a new AWS customer, you can get started with Amazon Transcribe 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

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, 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 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

Step 2: Set up the AWS Command Line Interface (AWS CLI) (p. 51)

Step 2: Set up the AWS Command Line Interface (AWS CLI)

You don’t need the AWS CLI to perform the steps in the Getting Started exercises. However, some of the other exercises in this guide do require it. If you prefer, you can skip this step and set up the AWS CLI later.

To set up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the AWS Command Line Interface User Guide:
   - Getting Set Up with the AWS Command Line Interface
   - Configuring the AWS Command Line Interface
2. In the AWS CLI config file, add a named profile for the administrator user:

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

   You use this profile when executing the AWS CLI commands. For more information about named profiles, see Named Profiles in the AWS Command Line Interface User Guide. For a list of AWS Regions, see Regions and Endpoints in the Amazon Web Services General Reference.
3. Verify the setup by typing the following help command at the command prompt:

   aws help
Next Step

Step 3: Getting Started Using the Console (p. 52)

Step 3: Getting Started Using the Console

The easiest way to get started with Amazon Transcribe is to submit a job using the console to transcribe an audio file. If you haven't reviewed the concepts and terminology in How Amazon Transcribe Works (p. 4), we recommend that you do that before proceeding.

Topics

- Create a Transcription Job (p. 52)
- View a Transcription Job (p. 53)

Create a Transcription Job

Use the Amazon Transcribe console to create a transcription job for your audio files.

1. Provide the following information:
   
   - **Transcription job name**—A name for the job. The name must be unique within your AWS account.
   - **Amazon S3 input URL**—The Amazon S3 location of your input audio file. The location must be in the same region as the endpoint that you are calling.
   - **Language**—Choose the language of your input file.
   - **Format**—The format of the audio file. For best results you should use a lossless format such as FLAC or WAV with PCM 16-bit encoding.
   - **Media sampling rate (Hz)**—Optional. The bit sampling rate of the audio file. Amazon Transcribe accepts sample rates between 8000 Hz and 48000 Hz. For best results, you should use 8000 Hz for low-fidelity audio and 16000 for high-fidelity audio.

   The following shows the Create Transcription Job filled out for a sample job.

   [Create Transcription Job](#)
2. Choose **Create** to submit the job for processing.

**View a Transcription Job**

Completed transcription jobs are displayed in a list that contains a brief description of the job. The **Availability** column shows the remaining time that the job results will be kept on the server. Jobs are kept for 90 days and then deleted from the system.
Choose a job in the list to see information about the job.

The information page about the transcription job has three sections. The **Detail** section provides details about the transcription job, including the name, information about when the job will be deleted from the server, and the input and output URLs. Use the output URL to download the output from your transcription job.

The **Output** section contains the transcription of the audio submitted to Amazon Transcribe. You can download the transcription by choosing the **Download transcription** button.

The **Code samples** section contains the JSON input for the `StartTranscriptionJob` operation and the output from the `GetTranscriptionJob` operation.
**Code Samples**

**Audio conversion**

**JSON Request**

```
{
    "TranscriptionJobName": "040cba",
    "LanguageCode": "en-US",
    "MediaSampleRateHertz": 8000,
    "MediaFormat": "wav",
    "Media": {
        "MediaFileUri": "https://answer2.wav"
    }
}
```

**JSON Response**

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "040cba",
        "TranscriptionJobStatus": "COMPLETED",
        "LanguageCode": "en-US",
        "MediaSampleRateHertz": 8000,
        "MediaFormat": "wav",
        "Media": {
        }
    }
}
```

**Next Step**

*Step 4: Getting Started Using the API (p. 56)*
Step 4: Getting Started Using the API

This section contains examples that demonstrate using the Amazon Transcribe API. You can use these samples to learn about the API or as building blocks in your own applications.

Topics
- Getting Started (AWS Command Line Interface) (p. 56)
- Getting Started (AWS SDK for Python (Boto)) (p. 58)

Getting Started (AWS Command Line Interface)

In the following exercise, you use the AWS Command Line Interface (AWS CLI) to transcribe speech into text. To complete this exercise, you need to:

1. Have a text editor.
2. Be familiar with the AWS CLI. For more information, see Step 2: Set up the AWS Command Line Interface (AWS CLI) (p. 51).
3. Have a speech file in .WAV or .MP4 format that is stored in an S3 bucket that has the proper permissions. For more information about the permissions needed for Amazon Transcribe, see Permissions Required for IAM User Roles (p. 210).

To transcribe text, you have to provide the input parameters in a JSON file.

To transcribe text

1. Copy your input speech to an S3 bucket. The location must be in the same region as the endpoint that you are calling. This example assumes that the file is in an S3 bucket named test-transcribe and that the file name is answer2.wav.
2. Create a JSON file named test-start-command.json that contains the input parameters for the StartTranscriptionJob (p. 297) operation. Enter a unique name for your transcription job under "TranscriptionJobName".

```json
{
  "TranscriptionJobName": "unique job name",
  "LanguageCode": "en-US",
  "MediaFormat": "wav",
  "Media": {
    "MediaFileUri": "https://S3 endpoint/test-transcribe/answer2.wav"
  }
}
```
3. In the AWS CLI, run the following command. The example is formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

```bash
aws transcribe start-transcription-job \
  --region region \
  --cli-input-json file://test-start-command.json
```

Amazon Transcribe responds with the following:

```json
{
  "TranscriptionJob": {
    "TranscriptionJobName": "unique job name",
```
To list transcription jobs

- Run the following command:

```
aws transcribe list-transcription-jobs \
  --region region \
  --status IN_PROGRESS
```

Amazon Transcribe responds with the following:

```
{
  "Status": "IN_PROGRESS",
  "TranscriptionJobSummaries": [
    {
      "TranscriptionJobName": "unique job name",
      "LanguageCode": "en-US",
      "CreationTime": timestamp,
      "TranscriptionJobStatus": "IN_PROGRESS"
    }
  ]
}
```

To get the results of a transcription job

1. When the job has the status COMPLETED, get the results of the job. Type the following command:

```
aws transcribe get-transcription-job \
  --region region \
  --transcription-job-name "unique job name"
```

Amazon Transcribe responds with the following:

```
{
  "TranscriptionJob": {
    "TranscriptionJobName": "unique job name",
    "LanguageCode": "en-US",
    "TranscriptionJobStatus": "COMPLETED",
    "Media": {
      "MediaFileUri": "input URI"
    },
    "CreationTime": timestamp,
    "CompletionTime": timestamp,
    "Transcript": {
      "TranscriptFileUri": "output URI"
    }
  }
}
```
2. Use the output URI to get the transcribed text from the audio file. The following is the output from transcribing a short audio clip:

```json
{
  "jobName": "job ID",
  "accountId": "account ID",
  "results": {
    "transcripts": [
      {
        "transcript": "that's no answer"
      }
    ],
    "items": [
      {
        "start_time": "0.180",
        "end_time": "0.470",
        "alternatives": [
          {
            "confidence": 0.84,
            "content": "that's"
          }
        ],
        "type": "pronunciation"
      },
      {
        "start_time": "0.470",
        "end_time": "0.710",
        "alternatives": [
          {
            "confidence": 0.99,
            "content": "no"
          }
        ],
        "type": "pronunciation"
      },
      {
        "start_time": "0.710",
        "end_time": "1.080",
        "alternatives": [
          {
            "confidence": 0.874,
            "content": "answer"
          }
        ],
        "type": "pronunciation"
      }
    ],
    "status": "COMPLETED"
  }
}
```

**Getting Started (AWS SDK for Python (Boto))**

In this exercise you create script that uses the SDK for Python to transcribe speech into text. To complete this exercise, you need to:

- Install the AWS CLI. For more information, see Step 2: Set up the AWS Command Line Interface (AWS CLI) (p. 51). This installs the AWS SDK for Python (Boto).
- Have a speech file in .WAV or .MP4 format that is stored in an S3 bucket that has the proper permissions. For more information about the permissions needed for Amazon Transcribe, see Permissions Required for IAM User Roles (p. 210). The location must be in the same region as the
endpoint that you are calling. This example assumes that the file is in an Amazon S3 bucket named test-transcribe and that the file name is answer2.wav.

```python
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "job name"
job_uri = "https://S3 endpoint/test-transcribe/answer2.wav"
transcribe.start_transcription_job(
    TranscriptionJobName=job_name,
    Media={'MediaFileUri': job_uri},
    MediaFormat='wav',
    LanguageCode='en-US')
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)
```

When the transcription job is complete, the result links to an Amazon S3 presigned URL that contains the transcription in JSON format:

```json
{
    "jobName": "job ID",
    "accountId": "account ID",
    "results": {
        "transcripts": [
            {
                "transcript": "that's no answer"
            }
        ],
        "items": [
            {
                "start_time": "0.180",
                "end_time": "0.470",
                "alternatives": [
                    {
                        "confidence": 0.84,
                        "word": "that's"
                    }
                ]
            },
            {
                "start_time": "0.470",
                "end_time": "0.710",
                "alternatives": [
                    {
                        "confidence": 0.99,
                        "word": "no"
                    }
                ]
            },
            {
                "start_time": "0.710",
                "end_time": "1.080",
                "alternatives": [
                    {
                        "confidence": 0.87,
                        "word": "answer"
                    }
                ]
            }
        ]
    }
}
```
Step 5: Getting Started With Streaming Audio

The following example is a Java program that transcribes streaming audio. The input comes from your computer's microphone or a file upload and the output is presented on your computer's standard output.

To run this example, you need 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.

```java
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args) throws URISyntaxException, ExecutionException, InterruptedException, LineUnavailableException {
        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

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

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

    private static InputStream getStreamFromMic() throws LineUnavailableException {
        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

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

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

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

    private static AwsCredentialsProvider getCredentials() {
        // Credential provider implementation
    }
```
return DefaultCredentialsProvider.create();

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

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

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

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

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

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {
        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
        }
    }

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

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

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

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {
            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
            }
        }
    }
}
this.currentSubscription = new SubscriptionImpl(s, inputStream);
)
s.onSubscribe(currentSubscription);
}

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

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

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be positive"));
        }
        demand.getAndAdd(n);
        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent = audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                } while (demand.decrementAndGet() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }

    @Override
    public void cancel() {
        executor.shutdown();
    }

    private ByteBuffer getNextEvent() {
        ByteBuffer audioBuffer = null;
        byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];
        int len = 0;
        try {
            len = inputStream.read(audioBytes);
            if (len <= 0) {
                audioBuffer = ByteBuffer.allocate(0);
            } else {
                audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
            }
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }
        return audioBuffer;
    }
}
```java
return audioBuffer;
}

private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
            .audioChunk(SdkBytes.fromByteBuffer(bb))
            .build();
}
```
Improving Domain-Specific Transcription Accuracy with Custom Language Models

Use custom language models to train and develop language models that are domain-specific. For example, you can use custom language models to improve transcription performance for domains such as legal, hospitality, finance, and insurance. Although the general model provided by Amazon Transcribe works well in most instances, custom language models might produce even more accurate results.

To train a custom language model, you must upload text data from your specific use case to Amazon Simple Storage Service (Amazon S3), provide Amazon Transcribe with permission to access that data, and choose a base model. A base model is a general speech recognition model, which you customize with your text data.

Custom language models use your text data to improve transcription accuracy for your use case. Your text data can include domain-specific text or audio transcripts. Domain-specific text data includes website content, instruction manuals, and technical documentation. Audio transcript data consists of ground-truth audio transcripts. Ground-truth audio transcripts have been processed with very high accuracy and are the ideal representation of the source audio.

You must provide text data that represents the audio that you want to transcribe. The domain-specific data that you provide must be related to your use case, and the audio transcript data that you provide should be similar to the audio that you want to transcribe. Any potential improvement in transcription accuracy depends on how closely your text data represents your audio and how much text data you provide. The quality of your text data is much more important than its quantity in creating custom language models that generate accurate transcriptions.

You upload your text data to Amazon Simple Storage Service (Amazon S3) and then give Amazon Transcribe access to the S3 buckets that contain that data. After uploading your data to Amazon S3, you choose a base model. A base model is a general speech recognition model that you customize with your text data.

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

1. Upload your text as training data. You use training data to train your custom language model for your specific use case.
2. Upload your domain-specific text as training data and your audio transcripts as tuning data. You use tuning data to optimize your custom language model and increase its transcription accuracy.

Use the following table to determine how to upload your data.

<table>
<thead>
<tr>
<th>If you have</th>
<th>Upload this</th>
</tr>
</thead>
<tbody>
<tr>
<td>A large amount of domain-specific text and a much smaller amount of audio transcript text data</td>
<td>Domain-specific text as training data. Upload your transcription text as tuning data.</td>
</tr>
<tr>
<td>A minimum of 10,000 words of audio transcript text</td>
<td>Audio transcript text as training data.</td>
</tr>
</tbody>
</table>
## Step 1: Preparing the 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.

### If you have

<table>
<thead>
<tr>
<th>At least 100,000 words of audio transcript text and a large amount of additional domain-specific text</th>
<th>Audio transcript text as training data. Typically, this method leads to the greatest possible increase in transcription accuracy. Follow the first method described in this table if this method doesn't produce the desired increase in transcription accuracy.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain-specific text only</td>
<td>Domain-specific text as training data. We recommend any of the preceding methods of uploading your data over this one.</td>
</tr>
</tbody>
</table>

You can provide up to 2 GB of training data and 200 MB of tuning data.

If you have enough text that represents the audio you want to transcribe, training custom language models can produce significant improvements in accuracy over using Custom Vocabularies (p. 12). Custom vocabularies improve the ability of Amazon Transcribe to recognize terms without using the context in which they're spoken. Custom language models not only recognize individual terms, but additionally use each term's context to transcribe your audio.

Custom language models can also add words to their recognition vocabularies automatically, eliminating the need for you to manually input new words. You can't use custom language models with custom vocabularies.

Custom language models are available only in US English (en-US).

You can't use AWS Key Management Service (AWS KMS) to encrypt your training data, but you can use AWS KMS condition keys to encrypt your transcription output. For information about condition keys, see Key Management (p. 197).

To use custom language models in your transcription jobs, you do the following:

- Prepare and upload plain text data
- Provide Amazon Transcribe with permissions to access your data
- Create a custom language model
- Use a custom language model in a transcription job
- View and update your custom language models to take advantage of speech recognition improvements in Amazon Transcribe

### Topics

- Step 1: Preparing the Data (p. 65)
- Step 2: Providing Amazon Transcribe with Data Permissions (p. 66)
- Step 3: Creating a Custom Language Model (p. 68)
- Step 4: Transcribing with a Custom Language Model (p. 70)
- Step 5: Viewing and Updating Your Custom Language Models (p. 72)
To prepare your text data:

1. Properly format it and save it in one or more text files. Make sure that each text file:
   • Is 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).
   • Has a single sentence per line.
   • 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. You can provide 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.

2. Upload those files to Amazon Simple Storage Service (Amazon S3). If you intend to tune the model, store your tuning data in a separate 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 is most convenient for you.

Next Step

Step 2: Providing Amazon Transcribe with Data Permissions (p. 66)
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**.

- Choose **Create role**.

- 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.

```json
{
   "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 that you want to attach the policy to.

- Choose **Create policy**.

   a. Choose **JSON**.

   b. Enter the following code, which adds **GetObject** and **ListBucket** permissions:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": [
            "s3:GetObject"
         ],
         "Resource": [
            "arn:aws:s3:::your-input-bucket/**
         ],
         "Effect": "Allow"
      },
      {
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [  
```
Step 3: Creating a Custom Language Model

To create a custom language model, you must provide your text data using an Amazon S3 prefix and specify a base model. There are two base models:

- **NarrowBand** - For audio with a sample rate of less than 16 kHz. You typically use this type of model for telephone conversations recorded at 8 kHz.
- **WideBand** - For audio with a sample rate of 16 kHz or greater. This includes audio from media sources.

You can create a custom language model with the base model and the training data using console, the `CreateLanguageModel` operation, the AWS Command Line Interface (AWS CLI).

**Using Prefixes in Amazon Simple Storage Service to Retrieve Your Data**

To create a custom language model, you use prefixes in Amazon Simple Storage Service to specify the training and the optional tuning data. To understand how to use prefixes, you need to be familiar with the following Amazon S3 concepts:

- **Bucket** - A container to store your objects.
- **Object** - The entity stored in the S3 bucket. In this case, it's your training or tuning text files.
- **Key** - The unique identifier for an object within a bucket.

You store an object, your text file, in a bucket with a key that uniquely identifies the file.

For example, the key `myfiles/2020/may/file.txt` uniquely identifies a text file in an S3 bucket. The prefix can be any part of the key that comes before its final delimiter.

You specify prefixes in the `CreateLanguageModel` operation using the following fields:

- **S3Uri** for the training data
- Optional: TuningDataS3Uri for the tuning data

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

You create a custom language model by providing prefixes to train a base model in either the console or the API.

**Creating a Custom Language Model (Console)**

**To creating a custom language model (Console)**

To use the console to create a custom language model with the console, you must have your training data stored in an Amazon S3 bucket.

1. Sign in to AWS Management Console and open the Amazon Transcribe console at [Amazon Transcribe console](https://transcribe.aws.amazon.com/).
2. In the navigation pane, choose Custom language models.
3. Choose Train model.
4. For Name, enter a name for your custom language model that is unique within your AWS account.
5. For Base model, choose Narrow band or Wide band, as appropriate for the sample rate of the audio that you want to transcribe.
6. Under Training data, for Training data location on S3, specify the S3 prefix that accesses only your training data.
7. Optional: Under Tuning data - optional, for Tuning data location on S3, specify the S3 prefix for the bucket where your tuning data is stored.
8. For Access permissions, use or create an IAM data access role that provides Amazon Transcribe with the ListBucket and GetObject permissions.
9. Choose Train model.

**Creating a Custom Language Model (API)**

To create a custom language model use the [CreateLanguageModel](https://docs.aws.amazon.com/transcribe/latest/dg/API_CreateLanguageModel.html) operation.

**To create a custom language model (API)**

- In an CreateLanguageModel request, specify the following:
  - BaseModelName - The type of base model that you want to use for your custom language model
  - InputDataConfig - Specify the Amazon S3 object location and the IAM data access role for your training data:
    - DataAccessRoleARN - The Amazon Resource Name (ARN) that identifies the permissions to your Amazon S3 bucket.
    - S3Uri - The prefix of the keys to your training data.
    - (Optional) TuningDataS3Uri - The prefix of the keys to your tuning data.
  - LanguageCode - The language code for the language that your training data is in.

    US English (en-US) is the only valid language code for custom language models.
• ModelName - The name of your custom language model.

The following is an example request using the AWS SDK for Python (Boto3).

```python
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
model_name = "example-language-model"
transcribe.create_language_model(
    LanguageCode = 'en-US',
    BaseModelName = 'WideBand',
    ModelName = model_name,
    InputDataConfig = {'S3Uri': 's3://DOC-EXAMPLE-BUCKET/clm-training/',
                      'TuningDataS3Uri': 's3://DOC-EXAMPLE-BUCKET/clm-tuning',
                      'DataAccessRoleArn': 'arn:aws:iam::AWS-account-number:role/IAM role'}
)
```

Creating a Custom Language Model (AWS CLI)

To create a custom language model (AWS CLI)

• Run the following code.

```bash
aws transcribe create-language-model \
  --language-code en-US \ 
  --base-model-name NarrowBand \ 
  --model-name example-model-name \ 
  --input-data-config S3Uri="s3://example-bucket",DataAccessRoleArn="arn:aws:iam::aws-account-number:role/IAM role"
```

Next Step

Step 4: Transcribing with a Custom Language Model (p. 70)

Step 4: Transcribing with a Custom Language Model

You can use a custom language model in transcription jobs with the Amazon Transcribe console, the StartTranscriptionJob (p. 297) operation, or the AWS CLI.

Transcribing with a Custom Language Model (Console)

To start a transcription job (console)

1. Sign in to AWS Management Console and open the Amazon Transcribe console at Amazon Transcribe console.
2. In the navigation pane, choose **Transcription jobs**.
3. For **Name**, enter a transcription job name that is unique within your AWS account.
4. For **Model selection**, choose your custom language model.
5. For **Input file location on S3**, enter the URI of the media file. If you can’t remember the URI, choose **Browse S3** and choose it.
6. Choose **Next**.
7. Enable any of the available features for your transcription job that you want to use.
8. Choose **Create**.

**Transcribing with a Custom Language Model (API)**

**To start a transcription job (API)**

1. Specify values for the required parameters:
   - **TranscriptionJobName** - The name of the transcription job.
   - **LanguageCode** - The language code of the transcription job. US English (en-US) is the only valid language code.
   - **MediaFileUri** parameter of the **Media** object - The Amazon S3 location of the media file that you want to transcribe.
   - **LanguageModelName** parameter of the **ModelSettings** object - The name of the custom language model.
2. Specify values for optional parameters. The following code shows both required and optional parameters:

```json
{
   "JobExecutionSettings": {
      "AllowDeferredExecution": boolean,
      "DataAccessRoleArn": "string"
   },
   "LanguageCode": "string",
   "Media": {
      "MediaFileUri": "string"
   },
   "MediaFormat": "string",
   "MediaSampleRateHertz": number,
   "ModelSettings": {
      "LanguageModelName": "string"
   },
   "OutputBucketName": "string",
   "OutputEncryptionKMSKeyId": "string",
   "Settings": {
      "ChannelIdentification": boolean,
      "MaxAlternatives": number,
      "MaxSpeakerLabels": number,
      "ProfanityCollectionName": "string",
      "ProfanityFilterMethod": "string",
      "ShowAlternatives": boolean,
      "ShowSpeakerLabels": boolean,
      "VocabularyName": "string"
   },
   "TranscriptionJobName": "string"
}
```
Transcribing with a Custom Language Model (AWS CLI)

To transcribe with a custom language model (AWS CLI)

- Run the following code.

```bash
aws transcribe start-transcription-job \
--transcription-job-name "example-job-name" \
--language-code "en-US" \
--media MediaFileUri="s3://example-bucket/example-audio.wav" \
--model-settings LanguageModelName="ExampleLanguageModel"
```

Next step

Step 5: Viewing and Updating Your Custom Language Models (p. 72)

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` operation 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.
# Streaming Transcription

Amazon Transcribe 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.

<table>
<thead>
<tr>
<th>Language</th>
<th>Sample Rate</th>
<th>Available In</th>
</tr>
</thead>
<tbody>
<tr>
<td>US English (en-US)</td>
<td>16 kHz, 8 kHz</td>
<td>Amazon Transcribe console, StartStreamTranscription (p. 319) operation, and WebSocket request</td>
</tr>
<tr>
<td>US Spanish (es-US)</td>
<td>16 kHz, 8 kHz</td>
<td>Amazon Transcribe console, StartStreamTranscription (p. 319) operation, and WebSocket request</td>
</tr>
<tr>
<td>Australian English (en-AU)</td>
<td>8 kHz</td>
<td>StartStreamTranscription (p. 319) operation and WebSocket request</td>
</tr>
<tr>
<td>British English (en-GB)</td>
<td>8 kHz</td>
<td>StartStreamTranscription (p. 319) operation and WebSocket request</td>
</tr>
<tr>
<td>French (fr-FR)</td>
<td>8 kHz</td>
<td>StartStreamTranscription (p. 319) operation and WebSocket request</td>
</tr>
<tr>
<td>Canadian French (fr-CA)</td>
<td>8 kHz</td>
<td>StartStreamTranscription (p. 319) operation and WebSocket request</td>
</tr>
<tr>
<td>German (de-DE)</td>
<td>16 kHz, 8 kHz</td>
<td>Amazon Transcribe console, StartStreamTranscription (p. 319) operation, and WebSocket request</td>
</tr>
<tr>
<td>Japanese (ja-JP)</td>
<td>16 kHz, 8 kHz</td>
<td>Amazon Transcribe console, StartStreamTranscription (p. 319) operation, and WebSocket request</td>
</tr>
<tr>
<td>Korean (ko-KR)</td>
<td>8 kHz</td>
<td>StartStreamTranscription (p. 319) operation and WebSocket request</td>
</tr>
<tr>
<td>Brazilian Portuguese (pt-BR)</td>
<td>16 kHz, 8 kHz</td>
<td>Amazon Transcribe console, StartStreamTranscription (p. 319) operation, and WebSocket request</td>
</tr>
<tr>
<td>Language</td>
<td>Sample Rate</td>
<td>Available In</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Italian (it-IT)</td>
<td>16 kHz, 8 kHz</td>
<td>Amazon Transcribe console, StartStreamTranscription (p. 319) operation and WebSocket request</td>
</tr>
</tbody>
</table>

Amazon Transcribe streaming transcription can be used for a variety of purposes. For example:

- Streaming transcriptions can generate real-time subtitles for live broadcast media.
- Lawyers can make real-time annotations on top of streaming transcriptions during courtroom depositions.
- Video game chat can be transcribed in real time so that hosts can moderate content or run real-time analysis.
- Streaming transcriptions can provide assistance to the hearing impaired.

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 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 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 amazon is the largest
the amazon is the largest
the amazon is the largest
the amazon is the largest rainforest
the amazon is the largest rainforest
the amazon is the largest rainforest
the amazon is the largest rainforest on the
the amazon is the largest rainforest on the
the amazon is the largest rainforest on the planet
the amazon is the largest rainforest on the planet
the amazon is the largest rainforest on the planet covering over
the amazon is the largest rainforest on the planet covering over
the amazon is the largest rainforest on the planet covering over two million
```

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.

```json
{  
  "TranscriptResultStream": {  
    "TranscriptEvent": {  
      "Transcript": {  
        "Results": [  
          {  
            "Alternatives": [  
              {  
                "Items": [  
                  {  
                    "Content": "the",  
                    "EndTime": 0.3799375,  
                    "StartTime": 0.0299375,  
                    "Type": "pronunciation"  
                  },  
                  {  
                    "Content": "amazon",  
                    "EndTime": 0.5899375,  
                    "StartTime": 0.3899375,  
                    "Type": "pronunciation"  
                  },  
                  {  
                    "Content": "is",  
                    "EndTime": 0.7899375,  
                    "StartTime": 0.5999375,  
                    "Type": "pronunciation"  
                  },  
                  {  
                    "Content": "the",  
                    "EndTime": 0.9199375,  
                    "StartTime": 0.7999375,  
                    "Type": "pronunciation"  
                  },  
                  {  
                    "Content": "largest",  
                    "EndTime": 1.0199375,  
                    "StartTime": 0.9299375,  
                    "Type": "pronunciation"  
                  }  
                ]  
              }  
            ]  
            "Transcript": "the amazon is the largest"  
          }  
        ],  
        "EndTime": 1.02,  
        "IsPartial": true,  
        "ResultId": "2db76dc8-d728-11e8-9f8b-f2801f1b9fd1",  
        "StartTime": 0.0199375  
      }  
    }  
  }  
}
```

**Topics**

- Event Stream Encoding (p. 76)
- Using Amazon Transcribe Streaming with WebSockets (p. 77)
- Using Amazon Transcribe Streaming With HTTP/2 (p. 83)
Event Stream Encoding

Event stream encoding provides bidirectional communication using messages 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 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 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 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 type.

• **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 Streaming with WebSockets

When you use the WebSocket protocol to stream audio, Amazon Transcribe transcribes the stream in real time. You encode the audio with event stream encoding, Amazon Transcribe responds with a JSON structure that is also encoded using event stream encoding. For more information, see Event Stream Encoding (p. 76). 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. 77)
- Creating a Pre-Signed URL (p. 78)
- Handling the WebSocket Upgrade Response (p. 81)
- Making a WebSocket Streaming Request (p. 82)
- Handling a WebSocket Streaming Response (p. 82)
- Handling WebSocket Streaming Errors (p. 83)

---

### Adding a Policy for WebSocket Requests to Your IAM Role

To use the WebSocket protocol to call Amazon Transcribe, you need to attach the following policy to the AWS Identity and Access Management (IAM) role that makes the request.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "transcribestreaming",
```
Creating a Pre-Signed URL

Create a URL for your WebSocket request that contains the information needed to set up communication between your application and Amazon Transcribe. WebSocket streaming 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 URL has the following format. Line breaks have been added for readability.

```
GET wss://transcribestreaming.region.amazonaws.com:8443/stream-transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=Signature Version 4 credential scope
&X-Amz-Date=date
&X-Amz-Expires=time in seconds until expiration
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding=mediaEncoding
&sample-rate=mediaSampleRateHertz
&session-id=sessionId
&vocabulary-name=vocabularyName
```

Use the following values for the URL parameters:

- **media-encoding** – The encoding used for the input audio. Valid values are pcm, ogg-opus, and flac.
- **sample-rate** – The sample rate of the input audio in Hertz. We suggest that you use 8000 Hz for low-quality audio and 16000 Hz for high-quality audio. The sample rate must match the sample rate in the audio file.
- **sessionId** – Optional. An identifier for the transcription session. If you don’t provide a session ID, Amazon Transcribe generates one for you and returns it in the response.
- **vocabulary-name** – Optional. The name of the vocabulary to use when processing the transcription job, if any.

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 separated by slashes (“/”) that is formed by concatenating your access key ID and your credential scope components. Credential scope includes the date in YYYYMMDD format, the AWS Region, the service name, and a special termination string (aws4_request).
- **X-Amz-Date** – The date and time that the signature was created. Generate the date and time by following the instructions in Handling Dates in Signature Version 4 in the Amazon Web Services General Reference.
• **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 URL 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 Amazon Web Services General Reference.

1. Define variables for the request in your application.

   ```
   # HTTP verb
   method = "GET"
   # Service name
   service = "transcribe"
   # AWS Region
   region = "AWS Region"
   # Amazon Transcribe streaming endpoint
   endpoint = "wss://transcribestreaming.region.amazonaws.com:8443"
   # Host
   host = "transcribestreaming.region.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 = "/stream-transcription-websocket"
   ```

3. Create the canonical headers and signed headers. Note the trailing "\n" in the canonical headers.

   ```
   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, 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 URL-encoded and sorted by name.
canonical_querystring = "X-Amz-Algorithm=" + algorithm
canonical_querystring += "X-Amz-Credential=" + access_key + "/
canonical_querystring += "X-Amz-Date=" + amz_date
canonical_querystring += "X-Amz-Expires=300"
canonical_querystring += "X-Amz-Security-Token=" + token
canonical_querystring += "X-Amz-SignedHeaders=" + signed_headers
canonical_querystring += "&language-code=en-US&media-encoding=pcm&sample-rate=16000"

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 Amazon Web Services 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 Amazon Web Services
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 Amazon Web Services 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, 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

**Task 4: 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 in the Amazon Web Services General Reference.

1. Add the authentication information to the query string.

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

2. Create the URL for the request.

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

You use the request URL with your WebSocket library to make the request to the Amazon Transcribe service.

**Including WebSocket Request Headers**

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

```
Host: transcribestreaming.region.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 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 to the websocket-location header for readability.

```
HTTP/1.1 101 Web Socket Protocol Handshake
Connection: upgrade
Upgrade: websocket
websocket-origin: wss://transcribestreaming.region.amazonaws.com:8443
```
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. 76).

Each data frame contains three headers combined with a chunk of raw audio bytes. The following table lists and describes the headers.

<table>
<thead>
<tr>
<th>Header Name (String)</th>
<th>Header Name Byte Length</th>
<th>Header Value Type</th>
<th>Value String Byte Length</th>
<th>Value String (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>:content-type</td>
<td>13</td>
<td>7</td>
<td>24</td>
<td>application/octet-stream</td>
</tr>
<tr>
<td>:event-type</td>
<td>11</td>
<td>7</td>
<td>10</td>
<td>AudioEvent</td>
</tr>
<tr>
<td>:message-type</td>
<td>13</td>
<td>7</td>
<td>5</td>
<td>event</td>
</tr>
</tbody>
</table>

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.
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. 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 sends a close frame.

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Header Name (String)</th>
<th>Header Value Type</th>
<th>Value String Byte Length</th>
<th>Value String (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>:content-type</td>
<td>7</td>
<td>24</td>
<td>application/octet-stream</td>
</tr>
<tr>
<td>15</td>
<td>:exception-type</td>
<td>7</td>
<td>varies</td>
<td>varies, see below</td>
</tr>
<tr>
<td>13</td>
<td>:message-type</td>
<td>7</td>
<td>9</td>
<td>exception</td>
</tr>
</tbody>
</table>

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 that 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 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, Amazon Transcribe can return any of the common service errors. For a list, see Common Errors.

Using Amazon Transcribe Streaming With HTTP/2

Amazon Transcribe uses a format called event stream encoding for streaming transcription. This format encoded binary data with header information that describes the contents of each event. For more
When Amazon Transcribe uses the **HTTP/2 protocol** for streaming transcriptions, the key components for a streaming request are:

- A header frame. This contains the HTTP/2 headers for the request, and a signature in the `authorization` header that Amazon Transcribe 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 `StartStreamTranscription (p. 319)` operation.

**Header Frame**

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.

**Required Headers**

The header frame of a request to Amazon Transcribe requires the following HTTP/2 headers:

```
POST /stream-transcription HTTP/2.0
host: transcribestreaming.region.amazonaws.com
authorization: Generated value
content-type: application/vnd.amazon.eventstream
x-amz-target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-EVENTS
x-amz-date: Date
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: media encoding
x-amzn-transcribe-sample-rate: Sample rate
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. For a list of valid regions, see **AWS Regions and Endpoints** in the Amazon Web Services General Reference.
- **authorization**: The Signature Version 4 signature for the request. For more information about creating a signature, see **Signing AWS Requests with Signature Version 4** in the Amazon Web Services General Reference.
- **x-amz-date**: Generate a date and time for the request following the instructions in **Handling Dates in Signature Version 4** in the Amazon Web Services General Reference.

For more information about the headers specific to Amazon Transcribe, see the `StartStreamTranscription (p. 319)` operation.
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.

The following diagram shows how this works.

Create the Audio Event

To create the message to send to Amazon Transcribe, create the audio event. Combine the headers described in the following table with a chunk of audio bytes into an event-encoded message.

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Header Value Type</th>
<th>Value String</th>
<th>Value (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>:content-type</td>
<td>7</td>
<td>24</td>
<td>application/octet-stream</td>
</tr>
<tr>
<td>:event-type</td>
<td>7</td>
<td>10</td>
<td>AudioEvent</td>
</tr>
<tr>
<td>:message-type</td>
<td>7</td>
<td>5</td>
<td>event</td>
</tr>
</tbody>
</table>

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. 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 that the audio stream is complete, send an empty data frame that contains only the date and signature.

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Header Value Type</th>
<th>Value String</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>:chunk-signature</td>
<td>6</td>
<td>varies</td>
<td>Generated signature</td>
</tr>
<tr>
<td>:date</td>
<td>8</td>
<td>8</td>
<td>Timestamp</td>
</tr>
</tbody>
</table>

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, 20190127T223754Z is 22:37:54 on 1/27/2019.

- **KEYPATH**: The signature scope in the format date/region/service/aws4_request. For example, 20190127/us-east-1/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 Amazon Web Services 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 for transcription.

### End Frame

To indicate that the audio stream is complete, send an end frame to Amazon Transcribe. 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 are also sent using event stream encoding. Use this information to decode a response from the StartStreamTranscription (p. 319) operation.
Transcription Response

A transcription response is event stream encoded. It contains the standard prelude and the following headers:

<table>
<thead>
<tr>
<th>Header Name Byte Length</th>
<th>Header Name (String)</th>
<th>Header Value Type</th>
<th>Value String Byte Length</th>
<th>Value String (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>:content-type</td>
<td>7</td>
<td>24</td>
<td>application/octet-stream</td>
</tr>
<tr>
<td>11</td>
<td>:event-type</td>
<td>7</td>
<td>10</td>
<td>AudioEvent</td>
</tr>
<tr>
<td>13</td>
<td>:message-type</td>
<td>7</td>
<td>5</td>
<td>event</td>
</tr>
</tbody>
</table>

For details, see Event Stream Encoding (p. 76).

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

```json
:content-type: "application/json"
:event-type: "TranscriptEvent"
:message-type: "event"
```

For an example of the JSON structure returned by Amazon Transcribe, see Using Amazon Transcribe Streaming With HTTP/2 (p. 83).

Exception Response

If there is an error in processing your transcription stream, Amazon Transcribe sends an exception response. The response is event stream encoded. For details, see Event Stream Encoding (p. 76).

The response contains the standard prelude and the following headers:

<table>
<thead>
<tr>
<th>Header Name Byte Length</th>
<th>Header Name (String)</th>
<th>Header Value Type</th>
<th>Value String Byte Length</th>
<th>Value String (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>:content-type</td>
<td>7</td>
<td>16</td>
<td>application/json</td>
</tr>
<tr>
<td>11</td>
<td>:event-type</td>
<td>7</td>
<td>19</td>
<td>BadRequestException</td>
</tr>
<tr>
<td>13</td>
<td>:message-type</td>
<td>7</td>
<td>9</td>
<td>exception</td>
</tr>
</tbody>
</table>

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

```json
:content-type: "application/json"
:event-type: "BadRequestException"
:message-type: "exception"
```

Example Request and Response

The following is an end-to-end example of a streaming transcription request. In this example, binary data is represented as base64-encoded strings. In an actual response, the data are raw bytes.
Step 1 - Start the Session With Amazon Transcribe

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

```
POST /stream-transcription HTTP/2.0
host: transcribestreaming.region.amazonaws.com
authorization: Generated value
content-type: application/vnd.amazon.eventstream
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-EVENTS
x-amz-date: Date
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: pcm
x-amzn-transcribe-sample-rate: Sample rate
transfer-encoding: chunked
```

Step 2 - Send Authentication Information to Amazon Transcribe

Amazon Transcribe sends the following response:

```
HTTP/2.0 200
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-sample-rate: Sample rate
x-amzn-request-id: 8a08df7d-5998-48bf-a303-484355b4ab4e
x-amzn-transcribe-session-id: b4526fcf-5eee-4361-8192-d1cb9e9d8887
x-amzn-transcribe-media-encoding: pcm
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. 76). The binary data in this request is base64-encoded. In an actual request, the data is raw bytes.

```
:content-type: "application/octet-stream"
:event-type: "AudioEvent"
:message-type: "event"
UklGRjzxPQBXQVZFZm10IBAAAAABAAEAgD4AAAB9AAACABAAZGF0YVTwPQAAAAAAAAAAAAAAD//wIA/f8EAA==
```

Step 4 - Create an Audio Event Message

Create an audio message that contains the audio data to send to Amazon Transcribe. For details, see Event Stream Encoding (p. 76). The audio event data in this example is base64-encoded. In an actual request, the data is raw bytes.

```
:date: 2019-01-29T01:56:17.291Z
:chunk-signature: signature
```
Step 5 - Use the Response from Amazon Transcribe

Amazon Transcribe 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.

The response from Amazon Transcribe is:

AAAAAUwAAAEP1RHpYBTpkYXRlCAAAAWiXUKMLEdpjaHVuy1zaWduYXR1cmUGACct62y+uymwEK1Srlp/zvBI
5eGn83jdBWCarUBJA+eaDafqjyI=

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

```json
:event-type: "TranscriptEvent"
:content-type: "application/json"
:message-type: "event"

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

For an example of the JSON structure returned by Amazon Transcribe, see Event Stream Encoding (p. 76).

Step 6 - End the Transcription Stream

Finally, send an empty audio event to Amazon Transcribe 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:

```plaintext
: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 streaming transcription. The code provides tolerance for intermittent failures in the connection to Amazon Transcribe. 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 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() {
```
/**
 * 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 occured:", 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) {
                    e1.printStackTrace();
                }
                log.debug("Making retry attempt: " + (retryAttempt + 1));
                recursiveStartStream(request, publisher, responseHandler, finalFuture,
retyAttempt + 1);
            } else {
                log.error("Encountered unretriable 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.
                transcriptionBehavior.onError(e);
            })
            .onComplete() -> {
                //Do nothing here. Don't close any streams that shouldn't be cleaned up yet.
                transcriptionBehavior.onComplete();
            }
            .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 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.
   * @param e
   */
  void onError(Throwable e);

  /**
   * Defines how to respond to the Transcript result stream.
   * @param e
   */
  void onStream(TranscriptResultStream e);

  /**
   * Defines what to do on initiating a stream connection with the service.
   * @param r
   */
  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
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 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. 94)

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. 89).
- 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. 92).
- Copy the sample application to your workspace. Build and run the application.
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"))nelStreamPublisher(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);
    }
}
} }
Amazon Transcribe Medical

What Is Amazon Transcribe Medical?

Amazon Transcribe Medical is an automatic speech recognition (ASR) service that enables you to add medical speech-to-text capabilities to an application. Amazon Transcribe Medical is a Health Insurance Portability and Accountability Act of 1996 (HIPAA) compliant service and is available through either real-time streaming or asynchronous API calls. To help with security and compliance, Amazon Transcribe Medical operates with a shared responsibility model. AWS is responsible for protecting the infrastructure that runs Amazon Transcribe Medical and you are responsible for the management of your data. For more information, see https://aws.amazon.com/compliance/shared-responsibility-model/

Use Amazon Transcribe Medical to transcribe medical-related speech found in a variety of use cases. Example uses include physician-dictated notes, drug safety monitoring, telemedicine, or physician-to-patient conversations.

Use Amazon Comprehend Medical for further analysis of your transcript.

Important Notice

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. All operations of Amazon Transcribe Medical should only be used in patient care scenarios after review for accuracy and sound medical judgment by trained medical professionals.

Streaming Audio Transcription

Amazon Transcribe Medical enables you to transcribe medical speech in real time. To do this, use a WebSocket protocol to establish a bi-directional connection to the Amazon Transcribe Medical real-time application programming interface. When you send Amazon Transcribe Medical a stream of audio, it returns a stream of JSON objects containing the audio’s transcription.

For more information about processing audio streams, see Establish a Bi-Directional Connection Using the WebSocket Protocol (p. 120).

API Batch Transcription

Amazon Transcribe Medical supports the transcription of individual audio files through asynchronous API batch calls. For more information about transcribing audio files, see the section called “How It Works” (p. 98)
Batch transcription can improve your transcription results. You can also use channel identification to separate and transcribe channels in audio files that have multiple channels. Channel identification generates separate transcripts for each channel and merges them into a single output. To identify speakers in single-channel audio files, use speaker identification. To generate additional alternative transcription results for the same source audio, use alternative transcriptions.

**Supported Specialties**

You can transcribe medical speech 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

**Are You a First-time User of Amazon Transcribe Medical?**

If you are a first-time user, we recommend that you read the following topic to get an introduction to Amazon Transcribe Medical: How Amazon Transcribe Medical Works (p. 98).

**How Amazon Transcribe Medical Works**

Amazon Transcribe Medical enables you to transcribe individual audio content with medical information into text. You can use either the streaming transcription API or the batch transcription API. Streaming transcription enables you to transcribe real-time streams of audio into text. To transcribe recorded audio files, use batch transcription. For more information on transcribing streaming audio, see Streaming Transcription Overview (p. 98) and Establish a Bi-Directional Connection Using the WebSocket Protocol (p. 120). For more information on batch transcription, see Batch Transcription Overview (p. 99).

**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. 100).
• 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. 120).

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 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 (p. 292) operation. To retrieve information on a medical transcription job, use GetMedicalTranscriptionJob (p. 259). You list medical transcription jobs with ListMedicalTranscriptionJobs (p. 277). You delete a medical transcription job with DeleteMedicalTranscriptionJob (p. 247).

To transcribe an audio file, you use a transcription job. You store the file as an object in an Amazon Simple Storage Service (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 duration or less than 2 GB in size.

For best results:

• Use a lossless format, such as FLAC or WAV.
• Use a sample rate of at least 16000 Hz or higher.

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 S3 buckets see Getting Started with Amazon Simple Storage Service

You can see the minimum number of request parameters to enter in the sample JSON here:

```json
{
```
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:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
</table>
| Convert cardinal numbers greater than 10 to numbers. | • "Fifty five" > 55  
• "a hundred" > 100  
• "One thousand and thirty one" > 1031  
• "One hundred twenty-three million four hundred fifty six thousand seven hundred eight nine" > 123,456,789 |
| Convert cardinal numbers followed by "million" or "billion" to numbers followed by a word when "million" or "billion" is not followed by a number. | • "one hundred million" > 100 million  
• "one billion" > 1 billion  
• "two point three million" > 2.3 million |
| Convert ordinal numbers greater than 10 to numbers. | • "Forty third" > 43rd  
• "twenty sixth avenue" > 26th avenue |
| Convert fractions to their numeric format. | • "a quarter" > 1/4  
• "three sixteenths" > 3/16  
• "a half" > 1/2  
• "a hundredth" > 1/100 |
| Convert numbers less than 10 to digits if there are more than one in a row. | • "three four five" > 345  
• "My phone number is four two five five five one two one two" > 4255551212 |
| Decimals are indicated by "dot" or "point." | • "three hundred and three dot five" > 303.5  
• "three point twenty three" > 3.23  
• "zero point four" > 0.4  
• "point three" > 0.3 |
| Convert the word "percent" after a number to a percent sign. | • "twenty three percent" > 23%  
• "twenty three point four five percent" > 23.45% |
### Rule Description

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert the words &quot;dollar,&quot; &quot;US dollar,&quot; &quot;Australian dollar,&quot; &quot;AUD,&quot; or &quot;USD&quot; after a number to a dollar symbol before the number.</td>
<td>• &quot;one dollar and fifteen cents&quot; &gt; $1.15&lt;br&gt;• &quot;twenty three USD&quot; &gt; $23&lt;br&gt;• &quot;twenty three Australian dollars&quot; &gt; $23</td>
</tr>
<tr>
<td>Convert the words &quot;pounds,&quot; or &quot;milligrams&quot; to &quot;lbs&quot; or &quot;mg&quot;.</td>
<td>• &quot;twenty three pounds&quot; &gt; 23 lbs&lt;br&gt;• &quot;forty-five milligrams&quot; &gt; 45 mg</td>
</tr>
<tr>
<td>Convert the words &quot;rupees,&quot; &quot;Indian rupees,&quot; or &quot;INR&quot; after a number to rupee sign (₹) before the number.</td>
<td>• &quot;twenty three rupees&quot; &gt; ₹23&lt;br&gt;• &quot;fifty rupees thirty paise&quot; &gt; ₹50.30</td>
</tr>
<tr>
<td>Convert times to numbers.</td>
<td>• &quot;seven a.m. eastern standard time&quot; &gt; 7 a.m.&lt;br&gt;&quot;twelve thirty p.m.&quot; &gt; 12:30 p.m.</td>
</tr>
<tr>
<td>Combine years expressed as two digits into four. Only valid for the 20th, 21st, and 22nd centuries.</td>
<td>• &quot;nineteen sixty two&quot; &gt; 1962&lt;br&gt;• &quot;the year is twenty twelve&quot; &gt; the year is 2012&lt;br&gt;• &quot;twenty nineteen&quot; &gt; 2019&lt;br&gt;• &quot;twenty one thirty&quot; &gt; 2130</td>
</tr>
<tr>
<td>Convert dates to numbers.</td>
<td>• &quot;May fifth twenty twelve&quot; &gt; May 5th 2012&lt;br&gt;&quot;May five twenty twelve&quot; &gt; May 5 2012&lt;br&gt;&quot;five May twenty twelve&quot; &gt; 5 May 2012</td>
</tr>
<tr>
<td>Separate spans of numbers by the word &quot;to.&quot;</td>
<td>• &quot;twenty three to thirty seven&quot; &gt; 23 to 37</td>
</tr>
</tbody>
</table>

## 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.

<table>
<thead>
<tr>
<th>Term Spoken in Source Audio</th>
<th>Abbreviation used in Output</th>
<th>Example Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centigrade</td>
<td>C</td>
<td>The patient’s temperature is 37.4°C.</td>
</tr>
<tr>
<td>Celsius</td>
<td>C</td>
<td>The patient’s temperature is 37.4°C.</td>
</tr>
<tr>
<td>Fahrenheit</td>
<td>F</td>
<td>The patient’s temperature is 101°F.</td>
</tr>
<tr>
<td>grams</td>
<td>g</td>
<td>A mass of 100 g was extracted from the patient.</td>
</tr>
<tr>
<td>meters</td>
<td>m</td>
<td>The patient is 1.8 m tall.</td>
</tr>
<tr>
<td>feet</td>
<td>ft</td>
<td>The patient is 6 ft tall.</td>
</tr>
</tbody>
</table>
### 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. 103)
- Getting Started With Console Streaming (p. 104)
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 AWS (p. 103)
2. Create an IAM User (p. 103)

Sign up for AWS

When you sign up for Amazon Web Services (AWS), 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

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 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:
Getting Started With Console Streaming

Get started with Amazon Transcribe Medical streaming transcription by using the 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 Amazon Transcribe console at https://console.aws.amazon.com/transcribe.
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 console.

Next Step

To learn more about WebSocket streaming, go to WebSocket Streaming (p. 120).

Getting Started With Batch Transcription

Get started with Amazon Transcribe Medical batch transcription in the 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 Amazon Transcribe console at https://console.aws.amazon.com/transcribe.
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 a finished specifying the details of your job, choose Create to start a medical transcription job.

Batch Transcription Overview (p. 99)

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.
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 amazon is the largest
the amazon is the largest
the amazon is the largest
the amazon is the largest rainforest
the amazon is the largest rainforest
the amazon is the largest rainforest on the
the amazon is the largest rainforest on the
the amazon is the largest rainforest on the planet
the amazon is the largest rainforest on the planet
the amazon is the largest rainforest on the planet
the amazon is the largest rainforest on the planet covering over
the amazon is the largest rainforest on the planet covering over
the amazon is the largest rainforest on the planet covering over two million
```

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.

```
{
  "TranscriptResultStream": {
    "TranscriptEvent": {
      "Transcript": {
        "Results": [
        {
          "Alternatives": [
            {
              "Items": [
```

<table>
<thead>
<tr>
<th>Language</th>
<th>Sample Rate</th>
<th>Available In</th>
</tr>
</thead>
<tbody>
<tr>
<td>US English (en-US)</td>
<td>16 kHz, 8 kHz</td>
<td>Amazon Transcribe Medical console, StartMedicalStreamTranscription (p. 313) operation, and WebSocket request</td>
</tr>
</tbody>
</table>
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 type.

• **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 encoded binary data with header information that describes the contents of each event. For more information, see *Event Stream Encoding (p. 76)*. You can use this information for applications that call the Amazon Transcribe Medical endpoint without using the Amazon Transcribe Medical SDK.

When Amazon Transcribe Medical uses the **HTTP/2 protocol** for streaming-med transcriptions, the key components for a streaming-med 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 (p. 313)** operation.

#### 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 /stream-transcription HTTP/2.0
host: transcribestreaming.region.amazonaws.com
authorization: Generated value
content-type: application/vnd.amazon.eventstream
x-amz-target: com.amazonaws.transcribe.Transcribe.StartStreamTranscription
x-amz-content-sha256: streaming-med-AWS4-HMAC-SHA256-EVENTS
x-amz-date: Date
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: media encoding
x-amzn-transcribe-sample-rate: Sample rate
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 regions, see [AWS Regions and Endpoints](https://docs.aws.amazon.com/general/latestguide/spn-list.html) in the [Amazon Web Services General Reference](https://docs.aws.amazon.com/en_us/home.html).

• **authorization**: The Signature Version 4 signature for the request. For more information about creating a signature, see [Signing AWS Requests with Signature Version 4](https://docs.aws.amazon.com/general/latestguide/sign-v4-library.html) in the [Amazon Web Services General Reference](https://docs.aws.amazon.com/en_us/home.html).

• **x-amz-date**: Generate a date and time for the request following the instructions in [Handling Dates in Signature Version 4](https://docs.aws.amazon.com/general/latestguide/sign-v4-library.html) in the [Amazon Web Services General Reference](https://docs.aws.amazon.com/en_us/home.html).

For more information about the headers specific to Amazon Transcribe Medical, see the [StartMedicalStreamTranscription](https://docs.aws.amazon.com/medicaltranscribe/latest/dg/api-requests-medical.html) operation.

### 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.

![Diagram of data frame creation](image)

#### 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.

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Header Name (string)</th>
<th>Header Value</th>
<th>Value Type</th>
<th>Value String</th>
<th>Byte Length</th>
<th>Value String (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>:content-type</td>
<td>7</td>
<td>24</td>
<td>application/octet-stream</td>
<td>AudioEvent</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>:event-type</td>
<td>7</td>
<td>10</td>
<td>AudioEvent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>:message-type</td>
<td>7</td>
<td>5</td>
<td>event</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Header Name (String)</th>
<th>Header Value Type</th>
<th>Value String Byte Length</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>:chunk-signature</td>
<td>6</td>
<td>varies</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>:date</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

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:

```java
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, 20190127T223754Z is 22:37:54 on 1/27/2019.
- **KEYPATH**: The signature scope in the format date/region/service/aws4_request. For example, 20190127/us-east-1/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 Amazon Web Services General Reference.

```java
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* (p. 313) operation.

Transcription Response

A transcription response is event stream encoded. It contains the standard prelude and the following headers:

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Header Name (String)</th>
<th>Header Value Type</th>
<th>Value String Byte Length</th>
<th>Value String (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>:content-type</td>
<td>7</td>
<td>24</td>
<td>application/octet-stream</td>
</tr>
<tr>
<td>11</td>
<td>:event-type</td>
<td>7</td>
<td>10</td>
<td>AudioEvent</td>
</tr>
<tr>
<td>13</td>
<td>:message-type</td>
<td>7</td>
<td>5</td>
<td>event</td>
</tr>
</tbody>
</table>

For details, see *Event Stream Encoding* (p. 76).

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

```
:content-type: "application/json"
: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. 108).

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. 76).

The response contains the standard prelude and the following headers:

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Header Name (String)</th>
<th>Header Value Type</th>
<th>Value String Byte Length</th>
<th>Value String (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>:content-type</td>
<td>7</td>
<td>16</td>
<td>application/json</td>
</tr>
<tr>
<td>11</td>
<td>:event-type</td>
<td>7</td>
<td>19</td>
<td>BadRequestException</td>
</tr>
<tr>
<td>13</td>
<td>:message-type</td>
<td>7</td>
<td>9</td>
<td>exception</td>
</tr>
</tbody>
</table>

When the exception response is decoded, it contains the following information:
Example Request and Response

The following is an end-to-end example of a streaming transcription request. In this example, binary data is 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:

```plaintext
POST /stream-transcription HTTP/2.0
host: transcribestreaming.region.amazonaws.com
authorization: Generated value
content-type: application/vnd.amazon.eventstream
x-amz-content-sha256: streaming-med-AWS4-HMAC-SHA256-EVENTS
x-amz-date: Date
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-media-encoding: Media encoding
x-amzn-transcribe-sample-rate: Sample rate
transfer-encoding: chunked
```

Step 2 - Send Authentication Information to Amazon Transcribe Medical

Amazon Transcribe Medical sends the following response:

```plaintext
HTTP/2.0 200
x-amzn-transcribe-language-code: en-US
x-amzn-transcribe-sample-rate: Sample rate
x-amzn-request-id: 8a08df7d-5998-48bf-a303-484355b4ab4e
x-amzn-transcribe-session-id: b4526fcf-5eee-4361-8192-d1cb9e9d6887
x-amzn-transcribe-media-encoding: pcm
x-amzn-Request-Id: 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. 106). The binary data in this request is base64-encoded. In an actual request, the data is raw bytes.

```plaintext
:content-type: "application/octet-stream"
:event-type: "AudioEvent"
:message-type: "event"
UklGRjzxPQBXQVZFWm10IBAAAAABAAEAgD4AAAB9AACABAAZGF0YTtwPQQAAAACAAAAD//wIwAd/4f8EAA==
```
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. 106). The audio event data in this example is base64-encoded. In an actual request, the data is raw bytes.

```
:date: 2019-01-29T01:56:17.291Z
:chunk-signature: signature
AAAA0gAAAIKVoRFcTTcjbj250Z50LXR5cGuHAhhcHBsaWNhdGlvbi9yY31ldC1zI1Y1WLOmV2ZW50LXR5cGUHApbdWRpb0V2ZW50TptZXNzYWdlLXR5cGUHAAVldmVudAxDaDb256ZW50LXR5cGUHApbhcHBsaWNhdGlvbi94LWFtei1lc29uLTBlMVJhYi08T0AV0FWRUWzdCAQAAAAAQADBAAAAAfQAAAgAQAGRhjDGFU8D0AAAAAA/
```

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.

The response from Amazon Transcribe Medical is:

```
AAAAUwAAEP1RHpYBTpkYXRlCAAAAWiUXkJaRVuay1zaWduYXRlcmUGACt6Zy+yuymwEk2SrIp/zVBI5eGn83jdWCaRUBJA+aDafqjqi=
```

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

```
:event-type: "TranscriptEvent"
:content-type: "application/json"
:message-type: "event"
{"Transcript":{"Results": [results]}}
```

For an example of the JSON structure returned by Amazon Transcribe Medical, see Event Stream Encoding (p. 106).

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.

```java
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();
                }
            }
        }
    });
}
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.

```java
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.

```java
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.
 */
```
Next step

Using the HTTP/2 Retry Client (p. 118)

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. 114).
• 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. 117).

• Copy the sample application to your workspace. Build and run the application.

```java
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 {
        TranscribeStreamingRetryClient client = new
                TranscribeStreamingRetryClient(EnvironmentVariableCredentialsProvider.create(), endpoint,
                region);

        StartStreamTranscriptionRequest request = StartStreamTranscriptionRequest.builder()
                .languageCode(language)
                .mediaEncoding(encoding)
                .mediaSampleRateHertz(sample_rate)
                .build();

        CompletableFuture<Void> result = client.startStreamTranscription(
                request,
                new AudioStreamPublisher(
                        new FileInputStream(new File("FileName"))));

        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;
        }
    }
```
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. 106). 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. 120)
- Creating a Pre-Signed URL (p. 121)
- Handling the WebSocket Upgrade Response (p. 124)
- Making a WebSocket Streaming Request (p. 125)
- Handling a WebSocket Streaming Response (p. 125)
- Handling WebSocket Streaming Errors (p. 126)

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 AWS Identity and Access Management (IAM) role that makes the request.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "transcribemedicalstreaming",
      "Effect": "Allow",
      "Action": "transcribe:StartMedicalStreamTranscription",
      "Resource": "*"
    }
  ]
}
```
Creating a Pre-Signed URL

You need to construct a URL 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 URL has the following format. Line breaks have been added for readability.

```
GET https://transcribestreaming.region.amazonaws.com:8443/medical-stream-transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=Signature Version 4 credential scope
&X-Amz-Date=date
&X-Amz-Expires=time in seconds until expiration
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding=mediaEncoding
&sample-rate=mediaSampleRateHertz
&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 URL 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. 16,000 Hz or higher sample rates 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 separated by slashes (“/”) that is formed by concatenating your access key ID and your credential scope components. Credential scope includes the date in YYYYYMMDD format, the AWS Region, the service name, and a special termination string (aws4_request).
- **X-Amz-Date** – The date and time that the signature was created. Generate the date and time by following the instructions in Handling Dates in Signature Version 4 in the Amazon Web Services General Reference.
- **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 URL 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 Amazon Web Services General Reference.

1. Define variables for the request in your application.

   ```
   # HTTP verb
   method = "GET"
   # Service name
   service = "transcribe"
   # AWS Region
   region = "AWS Region"
   # Amazon Transcribe streaming-med endpoint
   endpoint = "https://transcribestreaming.region.amazonaws.com:8443"
   # Host
   host = "transcribestreaming.region.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.

   ```
   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 URL-encoded and sorted by name.
canonical_querystring = "X-Amz-Algorithm=" + algorithm
canonical_querystring += "X-Amz-Credentials=" + access_key + "/" + credential_scope
canonical_querystring += "X-Amz-Date=" + amz_date
canonical_querystring += "X-Amz-Expires=300"
canonical_querystring += "X-Amz-Security-Token=" + token
canonical_querystring += "X-Amz-SignedHeaders=" + signed_headers
canonical_querystring += "&language-code=en-US&media-encoding=pcm&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 + '
' + canonical_uri + '
' + canonical_querystring + '
' + canonical_headers + '
' + signed_headers + '
' + 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 Amazon Web Services 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 Amazon Web Services 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 Amazon Web Services 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, 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

**Task 4: 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 HTTP Request in the Amazon Web Services General Reference.

1. Add the authentication information to the query string.

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

2. Create the URL for the request.

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

You use the request URL 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 headers are managed by your WebSocket client library.

- **Host**: `transcribestreaming.region.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 Web Socket Protocol Handshake
Connection: upgrade
Upgrade: websocket
websocket-origin: https://transcribestreaming.region.amazonaws.com:8443
websocket-location: transcribestreaming.region.amazonaws.com:8443/stream-transcription-websocket?
```
Using WebSocket Streaming

X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIDEXAMPLE%2F20190117%2Fregion%2Ftranscribe
&X-Amz-Date=date and time
&X-Amz-Expires=expiration length
&X-Amz-SignedHeaders=host
&language-code=language code
&media-encoding=media encoding
&sample-rate=media sample rate
&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. For a description of the contents, see Creating a Pre-Signed URL (p. 78).
- **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. 76).

Each data frame contains three headers combined with a chunk of raw audio bytes. The following table lists and describes the headers.

<table>
<thead>
<tr>
<th>Header Name (String)</th>
<th>Header Value Type</th>
<th>Value String Byte Length</th>
<th>Value String (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>:content-type</td>
<td>7</td>
<td>24</td>
<td>application/octet-stream</td>
</tr>
<tr>
<td>:event-type</td>
<td>7</td>
<td>10</td>
<td>AudioEvent</td>
</tr>
<tr>
<td>:message-type</td>
<td>7</td>
<td>5</td>
<td>event</td>
</tr>
</tbody>
</table>

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.
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. 104).

### 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.

<table>
<thead>
<tr>
<th>Header Name Length</th>
<th>Header Name (String)</th>
<th>Header Value Type</th>
<th>Value String Byte Length</th>
<th>Value String Type (UTF-8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>:content-type</td>
<td>7</td>
<td>24</td>
<td>application/octet-stream</td>
</tr>
<tr>
<td>11</td>
<td>:event-type</td>
<td>7</td>
<td>15</td>
<td>TranscriptEvent</td>
</tr>
<tr>
<td>13</td>
<td>:message-type</td>
<td>7</td>
<td>5</td>
<td>event</td>
</tr>
</tbody>
</table>

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 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
  - 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. 151).

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. 161).

For information about how numbers and medical measurements appear in the transcription output, see Transcribing Numbers (p. 100) and Transcribing Medical Terms and Measurements (p. 101).

**Topics**

- Transcribing an Audio File of a Medical Conversation (p. 127)
- Transcribing a Medical Conversation in a Real-Time Stream (p. 131)
- Identifying Speakers and Labeling Their Speech (p. 133)
- Transcribing Multi-Channel Audio (p. 140)

**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 (p. 292) operation or the Amazon Transcribe Medical console.

When you start a medical transcription job with the StartMedicalTranscriptionJob (p. 292) operation, you specify PRIMARYCARE as the value of the Specialty parameter.

**Console**

**To transcribe a clinician-patient dialogue (console)**

To use the console to transcribe a clinician-patient dialogue, create a transcription job and choose Conversation for Audio input type.

1. Sign in to AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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 Simple Storage Service (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)**

- In the **StartMedicalTranscriptionJob** (p. 292) operation, 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 as **PRIMARYCARE**.
  e. For **Type**, specify **CONVERSATION**.
  f. For **OutputBucketName**, specify the Amazon Simple Storage Service (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.

```python
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "medical-conversation-transcription-job-name"
job_uri = "https://DOC-EXAMPLE-BUCKET1.s3-Region.amazonaws.com/example-audio-file.extension"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    OutputBucketName = 'DOC-EXAMPLE-BUCKET2'
)
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.

```json
{
  "jobName": "conversation-medical-transcription-job",
  "accountId": "453794026688",
  "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"
      }
    ]
  }
}
```
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.

```json
{
    "MedicalTranscriptionJobName": "conversation-medical-transcription-job",
    "LanguageCode": "en-US",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName": "the-S3-bucket-where-you-output-the-transcription-results",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET1/example-audio-file.extension"
    }
}
```

The following is the response from running the preceding CLI command.

```json
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "example-conversation-medical-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "en-US",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET1/example-audio-file.extension"
        }
    }
}
```
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. 120). To start an HTTP/2 stream, use the StartMedicalStreamTranscription (p. 313) operation.

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 the section called “StartMedicalStreamTranscription” (p. 313) operation:

- 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.

Console

To transcribe a streaming medical conversation (console)

To use the 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 AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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` operation 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. 151).

To label the speech from different speakers, set the `ShowSpeakerLabel` parameter to `true`. For more information, see [Identifying Speakers and Labeling Their Speech](p. 133).

For more information on setting up an HTTP/2 stream to transcribe a medical conversation, see [Streaming Request](p. 108).

**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 URL. This URL 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. 120).

Use the following template to create your pre-signed URL.

```get
https://transcribestreaming.region.amazonaws.com:8443/medical-stream-transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=Signature Version 4 credential scope
&X-Amz-Date=date
&X-Amz-Expires=time in seconds until expiration
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding=mediaEncoding
&sample-rate=mediaSampleRateHertz
&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. 151).
To label the speech from different speakers, set the `show-speaker-label` parameter to `true`. For more information, see Identifying Speakers and Labeling Their Speech (p. 133).

For more information on creating pre-signed URLs, see Creating a Pre-Signed URL (p. 121).

### Identifying Speakers and Labeling Their Speech

To identify and label the speech from different speakers in Amazon Transcribe Medical, use *speaker identification*. This enables you to see what the patient said and what the clinician said in the transcription output.

When you enable speaker identification, 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 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 identification 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. 133)
- Identifying Speakers and Labeling Their Speech in Real-time Streams (p. 137)

### Identifying Speakers and Labeling Their Speech in Audio Files

You can enable speaker identification in a batch transcription job using either the `StartMedicalTranscriptionJob` (p. 292) operation or the Amazon Transcribe Medical console. This enables you to identify the speakers in a clinician-patient conversation and determine who said what in the transcription output.

#### Console

**To enable speaker identification in a transcription job (console)**

To use the console to enable speaker identification in your transcription job, you enable audio identification and then speaker identification.

1. Sign in to AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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)

- In the StartMedicalTranscriptionJob (p. 292) operation, specify the following.
  - For MedicalTranscriptionJobName, specify a name that is unique in your AWS account.
  - For LanguageCode, specify the language code that corresponds to the language spoken in the audio file.
  - In the MediaFileUri parameter of the Media object, specify the name of the audio file that you want to transcribe.
  - For Specialty, specify the medical specialty of the clinician speaking in the audio file.
  - For Type, specify CONVERSATION.
  - For OutputBucketName, specify the Amazon Simple Storage Service (Amazon S3) bucket to store the transcription results.
  - For the Settings object, specify the following.
    - ShowSpeakerLabels – true.
    - 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.

```python
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "example-diarization-transcription"
job_uri = "https://DOC-EXAMPLE-BUCKET1.s3-Region.amazonaws.com/example-file.mp4"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName=job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    OutputBucketName = 'DOC-EXAMPLE-BUCKET2',
    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"
      }
    ]
  }
}
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.

```bash
aws transcribe start-transcription-job
--cli-input-json file://example-start-command.json
```

The following code shows the contents of `example-start-command.json`.

```json
{
  "MedicalTranscriptionJobName": "speaker-id-conversation-medical-transcription-job",
  "LanguageCode": "language-code",
  "Specialty": "PRIMARYCARE",
  "Type": "CONVERSATION",
  "OutputBucketName": "DOC-EXAMPLE-BUCKET",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
  },
  "Settings":{
    "ShowSpeakerLabels": true,
    "MaxSpeakerLabels": 2
  }
}
```

The following is the response from running the preceding CLI command.

```json
{
  "MedicalTranscriptionJobName": "speaker-id-conversation-medical-transcription-job",
  "LanguageCode": "en-US",
  "Specialty": "PRIMARYCARE",
  "Type": "CONVERSATION",
  "OutputBucketName": "DOC-EXAMPLE-BUCKET1",
  "Media": {
    "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET1/example-file.extension"
  },
  "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 Amazon Transcribe Medical 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 (p. 313) operation. To start a WebSocket request, use a pre-signed URL. The URL 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 (console)

To identify speakers in audio that is spoken into your microphone (console)

You can use the Amazon Transcribe Medical 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 into the AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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 (p. 313) 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. 108).

Identifying Speakers in a WebSocket Request

To identify speakers in WebSocket streams with the API, use the following format to create a pre-signed URL to start a WebSocket request and set show-speaker-label to true.

```
GET https://transcribestreaming.region.amazonaws.com:8443/medical-stream-transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=Signature Version 4 credential scope
&X-Amz-Date=date
&X-Amz-Expires=time in seconds until expiration
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
```
Identifying Speakers

The following code shows the truncated example response of a streaming request.

```json
{
  "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
              },
              ...
            ]
          }
        ]
      }
    ]
  }
}
```
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 above 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 URL (p. 121).

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 – Console and StartMedicalTranscriptionJob (p. 292) operation
- Streaming transcription – WebSocket Streaming and StartMedicalStreamTranscription (p. 313) operation

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 Amazon Transcribe Medical console or the StartMedicalTranscriptionJob (p. 292) operation.

Console

To use the 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 console.

To transcribe a multi-channel audio file (console)

1. Sign in to AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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)**

- In the **StartMedicalTranscriptionJob** (p. 292) operation, 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. In 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).

```python
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "your-transcription-job-name"
job_uri = "the-Amazon-S3-object-URL-of-your-media-file"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName=job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    OutputBucketName = 'DOC-EXAMPLE-BUCKET1',
    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.
The following is the code of example-start-command.json.

```
{
    "MedicalTranscriptionJobName": "multichannel-conversation-medical-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName": "DOC-EXAMPLE-BUCKET",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/example-audio-file.extension"
    },
    "Settings": {
      "ChannelIdentification": true
    }
}
```

The following is the response.

```
{
    "MedicalTranscriptionJob": {
      "MedicalTranscriptionJobName": "multichannel-conversation-medical-transcription-job",
      "TranscriptionJobStatus": "IN_PROGRESS",
      "LanguageCode": "language-code",
      "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/example-audio-file.extension"
      },
      "StartTime": "2020-09-20T23:46:44.081000+00:00",
      "CreationTime": "2020-09-20T23:46:44.053000+00:00",
      "Settings": {
        "ChannelIdentification": true
      },
      "Specialty": "PRIMARYCARE",
      "Type": "CONVERSATION"
    }
}
```

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"
  
  ],
  "channel_label": "ch_1",
  "items": [
  
  "start_time": "12.379",
  "end_time": "12.589",
  "alternatives": [
  
  "confidence": "0.5645",
  "content": "It"
  
  ],
  "type": "pronunciation"
  
  ]
  
  ]
  
  }
Transcribing Multi-Channel Audio Streams

You can transcribe audio from separate channels in either HTTP/2 or WebSocket streams using the StartMedicalStreamTranscription (p. 313) operation.

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 section called “StartMedicalStreamTranscription” (p. 313) operation 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. 108).

Transcribing Multi-Channel Audio in a WebSocket Stream

To identify speakers in WebSocket streams, use the following format to create a pre-signed URL 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 URL contains the information needed to set up bi-directional communication between your application and Amazon Transcribe Medical.

```text
GET wss://transcribestreaming.region.amazonaws.com:8443/stream-transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=Signature Version 4 credential scope
&X-Amz-Date=date
&X-Amz-Expires=time in seconds until expiration
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
```
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. 151).

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. 161).

For information about how numbers and medical measurements appear in the transcription output, see Transcribing Numbers (p. 100) and Transcribing Medical Terms and Measurements (p. 101).

Topics
• Transcribing an Audio File of a Medical Dictation (p. 146)
• Transcribing a Medical Dictation in a Real-Time Stream (p. 150)

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 (p. 292) operation or the Amazon Transcribe Medical console.

When you start a medical transcription job with the StartMedicalTranscriptionJob (p. 292) operation, you specify PRIMARYCARE as the value of the Specialty parameter.

Console

To transcribe a clinician-patient dialogue (console)

To use the console to transcribe a clinician-patient dialogue, create a transcription job and choose Conversation for Audio input type.

1. Sign in to AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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 Simple Storage Service (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)

- In the StartMedicalTranscriptionJob (p. 292) operation, 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 Simple Storage Service (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.

```python
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "your-medical-conversation-transcription-job-name"
job_uri = "https://DOC-EXAMPLE-BUCKET1.s3-Region.amazonaws.com/example-audio-file.extension"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName = job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = '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 of a medical dictation.

```
{
    "jobName": "dictation-medical-transcription-job",
    "accountId": "account-id-number",
    "results": {
        "transcripts": [
            {
```
"transcript": "... came for a follow up visit today..."
To identify the speakers in an audio file using a batch transcription job (AWS CLI)

- Run the following code.

```bash
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`.

```json
{
    "MedicalTranscriptionJobName": "conversation-medical-transcription-job",
    "LanguageCode": "en-US",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName": "DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET1/example-audio-file.extension"
    }
}
```

The following is the response from running the preceding CLI command.

```json
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "example-dictation-medical-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "en-US",
        "Media": {
            "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET1/example-audio-file.extension"
        },
        "StartTime": "2020-09-20T00:35:22.256000+00:00",
        "CreationTime": "2020-09-20T00:35:22.218000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```
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 Amazon Transcribe Medical 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 URL parameter of the WebSocket request, or the Specialty parameter of the "StartMedicalStreamTranscription" operation:

- 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.

To Transcribe a Dictation That Is Spoken Into Your Microphone (Console)

To use the 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.

1. Sign in to AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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 Medical Dictation In an HTTP/2 Stream

To transcribe an HTTP/2 stream of a medical dictation, use the StartMedicalStreamTranscription (p. 313) operation 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. 108).

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 URL. This URL 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. 120).

Use the following template to create your pre-signed URL.

```
GET https://transcribestreaming.region.amazonaws.com:8443/medical-stream-transcription-websocket
?language-code={languageCode}
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=Signature Version 4 credential scope
&X-Amz-Date={date}
&X-Amz-Expires=time in seconds until expiration
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding={mediaEncoding}
&sample-rate={mediaSampleRateHertz}
&session-id={sessionId}
&specialty={medicalSpecialty}
&type={DICTATION}
&vocabulary-name={vocabularyName}
&show-speaker-label={boolean}
```

For more information on creating pre-signed URLs, see Creating a Pre-Signed URL (p. 121).

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. 152)
- Using a Text File To Create a Medical Custom Vocabulary (p. 155)
- Transcribing An Audio File Using a Medical Custom Vocabulary (p. 157)
- Transcribing a Real-Time Stream Using a Medical Custom Vocabulary (p. 158)
- Character Sets for Amazon Transcribe Medical (p. 159)

**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 Simple Storage Service (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. 159).

**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. 159). 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 # l#.

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. 159). 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-pty*. 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. 159).

• **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 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]acute respiratory distress syndrome
A.L.L.[TAB][TAB][TAB]eɪ ɛ l ɛ l[TAB]ALL
```

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]acute respiratory distress syndrome
A.L.L.[TAB][TAB][TAB]eɪ ɛ l ɛ l[TAB]ALL
```

```
DisplayAs[TAB]SoundsLike[TAB]IPA[TAB]Phrase
acute respiratory distress syndrome[TAB][TAB][TAB]acute-respiratory-distress-syndrome
ALL[TAB][TAB][TAB][TAB][TAB]eɪ ɛ l ɛ l[TAB]A.L.L.
[TAB][TAB][TAB][TAB][TAB][TAB]ay-tree-o-ven-trick-u-lar-node-al-re-entr-ant-tack-ih-card-ia[TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TAB][TABLE
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 (p. 235) operation or the Amazon Transcribe Medical console.

**Console**

**To create a custom vocabulary (console)**

To use the 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 AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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 URI that identifies the text file you will use to create your custom vocabulary.
   - For Vocabulary input file location in Amazon 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 console.

**API**

**To create a medical custom vocabulary (API)**

- In the StartTranscriptionJob (p. 297) operation, 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 (p. 262) operation.

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-BUCKET1.s3-Region.amazonaws.com/example_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 your-custom-medical-vocabulary-name
    --language-code en-US

The following is the response from running the preceding CLI command.

    {
        "VocabularyName": "cli-medical-vocab-1","
Transcribing An Audio File Using a Medical Custom Vocabulary

Use the StartMedicalTranscriptionJob (p. 292) or the Amazon Transcribe Medical console to start a transcription job that uses a custom vocabulary to improve transcription accuracy.

Console

To start a transcription job with a custom vocabulary (console)

To use the console to use your custom vocabulary in your transcription job,.

1. Sign in to AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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. Under Customization, enable Custom vocabulary.
7. Under Vocabulary selection, choose a custom vocabulary.
8. Choose Create.

API

To identify speakers in an audio file using a batch transcription job (API)

1. In the StartMedicalTranscriptionJob (p. 292) operation, 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 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 Simple Storage Service (Amazon S3) bucket to store the transcription results.
   g. For the Settings object, specify the following.
      1. 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.

```python
from __future__ import print_function
```
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` operation. You can use a custom vocabulary in real-time using either the Amazon Transcribe Medical console, the `StartMedicalStreamTranscription` operation, or by using the WebSocket protocol.

To Transcribe a Dictation That Is Spoken Into Your Microphone (Console)

To use the 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.

1. Sign in to AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical.
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.
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.

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>aʊ</td>
<td>0061 028A</td>
<td>w</td>
<td>0077</td>
</tr>
<tr>
<td>aɪ</td>
<td>0061 026A</td>
<td>z</td>
<td>007A</td>
</tr>
<tr>
<td>b</td>
<td>0062</td>
<td>æ</td>
<td>00E6</td>
</tr>
<tr>
<td>d</td>
<td>0064</td>
<td>ð</td>
<td>00F0</td>
</tr>
<tr>
<td>eɪ</td>
<td>0065 026A</td>
<td>η</td>
<td>014B</td>
</tr>
<tr>
<td>f</td>
<td>0066</td>
<td>ə</td>
<td>0251</td>
</tr>
<tr>
<td>g</td>
<td>0067</td>
<td>ɔ</td>
<td>0254</td>
</tr>
<tr>
<td>h</td>
<td>0068</td>
<td>ɔɪ</td>
<td>0254 026A</td>
</tr>
<tr>
<td>i</td>
<td>0069</td>
<td>ə</td>
<td>0259</td>
</tr>
<tr>
<td>j</td>
<td>006A</td>
<td>ɛ</td>
<td>025B</td>
</tr>
<tr>
<td>k</td>
<td>006B</td>
<td>ɛɪ</td>
<td>025D</td>
</tr>
<tr>
<td>l</td>
<td>006C</td>
<td>g</td>
<td>0261</td>
</tr>
<tr>
<td>l̩</td>
<td>006C 0329</td>
<td>ɪ</td>
<td>026A</td>
</tr>
<tr>
<td>m</td>
<td>006D</td>
<td>ɭ</td>
<td>0279</td>
</tr>
<tr>
<td>n</td>
<td>006E</td>
<td>ɲ</td>
<td>0283</td>
</tr>
<tr>
<td>η</td>
<td>006E 0329</td>
<td>u</td>
<td>028A</td>
</tr>
<tr>
<td>ou</td>
<td>006F 028A</td>
<td>ʌ</td>
<td>028C</td>
</tr>
<tr>
<td>p</td>
<td>0070</td>
<td>ʍ</td>
<td>028D</td>
</tr>
<tr>
<td>s</td>
<td>0073</td>
<td>ʒ</td>
<td>0292</td>
</tr>
<tr>
<td>t</td>
<td>0074</td>
<td>ʤ</td>
<td>02A4</td>
</tr>
<tr>
<td>u</td>
<td>0075</td>
<td>ʧ</td>
<td>02A7</td>
</tr>
<tr>
<td>v</td>
<td>0076</td>
<td>θ</td>
<td>03B8</td>
</tr>
</tbody>
</table>
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 Amazon Transcribe Medical console or the StartMedicalTranscriptionJob (p. 292) operation.

**Console**

To generate additional alternative transcriptions in a transcription job (console)

To use the console to generate alternative transcriptions, you enable alternative results when you configure your job.

1. Sign in to AWS Management Console and open the Amazon Transcribe Medical console at Amazon Transcribe Medical 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.
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)

- In the StartMedicalTranscriptionJob (p. 292) operation, 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 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 you're transcribing a medical conversation or a dictation.
  f. For OutputBucketName, specify the Amazon Simple Storage Service (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 that 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.

```python
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "example-alternatives-transcription"
job_uri = "https://transcribe-learning-1.s3-us-west-2.amazonaws.com/medasrdemo-Paul.mp4"
transcribe.start_medical_transcription_job(
    MedicalTranscriptionJobName=job_name,
    Media = {'MediaFileUri': job_uri},
    LanguageCode = 'en-US',
    Specialty = 'PRIMARYCARE',
    Type = 'CONVERSATION',
    OutputBucketName = 'transcribe-testing-2',
),
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://example-start-command.json
```

The following code shows the contents of `example-start-command.json`.

```json
{
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-transcription-job",
    "LanguageCode": "en-US",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName": "DOC-EXAMPLE-BUCKET",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
    },
    "Settings":{
        "ShowAlternatives": true,
        "MaxAlternatives": 2
    }
}
```
The following is the response from running the preceding CLI command.

```
{
   "MedicalTranscriptionJob": {
      "MedicalTranscriptionJobName": "alternatives-medical-transcription-job",
      "TranscriptionJobStatus": "IN_PROGRESS",
      "LanguageCode": "en-US",
      "Media": {
         "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.extension"
      },
      "StartTime": "2020-09-21T19:09:18.199000+00:00",
      "CreationTime": "2020-09-21T19:09:18.171000+00:00",
      "Settings": {
         "ShowAlternatives": true,
         "MaxAlternatives": 2
      },
      "Specialty": "PRIMARYCARE",
      "Type": "CONVERSATION"
   }
}
```

---

## Guidelines and Quotas

### Supported Regions

For a list of AWS Regions where Amazon Transcribe Medical is available, see [Amazon Transcribe Endpoints and Quotas](https://docs.aws.amazon.com/transcribe/latest/dg/endpoints.html) in the [Amazon Web Services General Reference](https://docs.aws.amazon.com/awssdk/latest/api/).

### Throttling

You can request a quota increase for the following resources:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of concurrent batch transcription jobs</td>
<td>100</td>
</tr>
<tr>
<td>Transactions per second, StartMedicalTranscriptionJob</td>
<td>10</td>
</tr>
<tr>
<td>Number of StartMedicalStreamTranscription Websocket requests</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second, StartMedicalStreamTranscription operation</td>
<td>5</td>
</tr>
<tr>
<td>Resource</td>
<td>Default</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>Total number of medical vocabularies per account</td>
<td>100</td>
</tr>
<tr>
<td>Number of pending medical vocabularies</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second, GetMedicalTranscriptionJob operation</td>
<td>20</td>
</tr>
<tr>
<td>Transactions per second, DeleteMedicalTranscriptionJob operation</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second, ListMedicalTranscriptionJobs operation</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second, CreateMedicalVocabulary operation</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second, UpdateMedicalVocabulary operation</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second, DeleteMedicalVocabulary operation</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second, GetMedicalVocabulary operation</td>
<td>20</td>
</tr>
<tr>
<td>Transactions per second, ListMedicalVocabularies operation</td>
<td>5</td>
</tr>
</tbody>
</table>

For information about requesting a quota increase, see AWS Service Quotas in the Amazon Web Services 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 16000 Hz or greater.

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

Amazon Transcribe Medical has the following quotas that are not alterable:

<table>
<thead>
<tr>
<th>Description</th>
<th>Quotas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum audio file length</td>
<td>14,400 seconds</td>
</tr>
<tr>
<td>Maximum audio file size</td>
<td>2 GB</td>
</tr>
<tr>
<td>Number of days that job records are retained</td>
<td>90</td>
</tr>
<tr>
<td>Number of channels for channel identification</td>
<td>2</td>
</tr>
<tr>
<td>Minimum audio file duration, in milliseconds (ms)</td>
<td>500</td>
</tr>
</tbody>
</table>

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 Amazon VPC console or the AWS Command Line Interface (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.region.transcribe`

For streaming transcription in Amazon Transcribe Medical, create a VPC endpoint using the following service name:
Amazon Transcribe Developer 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.

```json
{
    "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.

```json
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": ["transcribe:StartMedicalTranscriptionJob"],
            "Resource": "*"
        }
    ]
}
```
Creating a VPC endpoint policy for
Amazon Transcribe Medical Streaming

}
]
Identifying the Language of Your Media Files

When you generate transcriptions, Amazon Transcribe can automatically identify the predominant language in a media file with automatic language identification. This enables you to transcribe your files without specifying a language code for each individual file.

Use automatic language identification to:

- Transcribe customer service recordings in countries where multiple languages are spoken.
- Transcribe a media library that has files in different languages.
- Label media content with the language automatically identified by Amazon Transcribe.
- Identify incorrectly labeled audio and video content for media content operations. For example, you can identify videos and podcasts labeled with the incorrect language.

Media files are transcribed in a single language, even if they contain speech in two or more languages. Amazon Transcribe transcribes the audio according to the primary language found in the file. Amazon Transcribe can automatically identify any language that can be used for batch transcription with the API or the Amazon Transcribe console. For a list of languages, see What Is Amazon Transcribe? (p. 1).

To identify the language with greater accuracy, you can specify a list of languages that you think are present in your collection of media files. From that list, Amazon Transcribe chooses the language with the greatest confidence score to transcribe your audio. A score with a larger value indicates that Amazon Transcribe is more confident that it identified the language correctly. For best results, if you are certain of the language spoken in each of the audio files, specify a language code. For more information, see the StartTranscriptionJob (p. 297) operation.

Some Amazon Transcribe features require you to specify a language code. If you try to automatically identify the language of your audio with the following features enabled, you receive an error:

- Custom language models
- Custom vocabularies
- Vocabulary filtering
- Automatic content redaction

To increase the chance of identifying the language successfully, media files should have at least 30 seconds of speech.

Topics
- Transcribing with Automatic Language Identification (p. 168)
- Getting Notifications Using Automatic Language Identification (p. 173)

Transcribing with Automatic Language Identification

You can use automatic language identification in a batch transcription job with either the Amazon Transcribe console or the StartTranscriptionJob (p. 297) operation.
Console

To start a transcription job with automatic language identification (console)

1. Sign in to the AWS Management Console and open the Amazon Transcribe console at Amazon Transcribe console.
2. In the navigation pane, choose Transcription jobs.
3. For Language settings, choose Automatic language identification.
4. (Optional) For Language options for automatic language identification - optional, choose any languages you think are present in the file that you want to transcribe.
5. For Input file location on S3, under Input data, enter the URI of your media file, or search for it in the Browse S3 search box.
6. For Data location, under Output data, choose the type of S3 bucket you’d like to use to store the transcription output.
7. Choose Next.
8. Choose Create.

API

To start a transcription job with automatic language identification (API)

In the StartTranscriptionJob (p. 297) operation, specify the following:

- For TranscriptionJobName, specify a name that is unique in your AWS account.
- In the MediaFileUri parameter of the Media object, specify the location in S3 object location of the media file that you want to transcribe.
- Set the IdentifyLanguage parameter to true.
- (Optional) For increased accuracy with language identification, enter an array of the languages that are spoken in your file in LanguageOptions. For example, if you're confident your media file is either in US English, US Spanish, or French, provide the following array: ["en-US", "es-US", "fr-FR"].

Don't specify a value for the LanguageCode parameter. Doing so generates a BadRequestException error.

If you specify languages for the LanguageOptions parameter, Amazon Transcribe shows the language codes and their associated confidence scores for the languages that you've specified in the transcription job's output. The audio is transcribed in the language with the highest confidence score. The following example transcription output shows that en-GB and de-DE were specified in LanguageOptions.

```json
{
   "jobName": "your-transcription-job",
   "accountId": "your-account-id",
   "results": {
      "language_code": "en-GB",
      "transcripts": [
         { "transcript": "So I see. Supposed to show some overeager squatting with an itchy trigger finger, that's who. [transcription output shortened for brevity] You know why? Why? Because I love it." }
      ],
      "language_identification": [ ...
```
If you don't specify a set of languages, the response lists the language codes and associated confidence scores of the languages that have the five highest confidence scores. Compare the following example response with the output in the previous example.
Amazon Transcribe always transcribes audio in the language that has the highest confidence score.

For more information about the request parameters used to start a batch transcription job and their data types, see StartTranscriptionJob (p. 297).

**AWS CLI**

To start a transcription job with automatic language identification enabled in the AWS CLI, use the following command.

```bash
aws transcribe start-transcription-job \
  --media MediaFileUri=s3://DOC-EXAMPLE-BUCKET1/your-media-file.mp4 \
  --identify-language \
  --transcription-job-name your-transcription-job-name
```

**Finding the Identified Language of a Transcription Job**

To see the language code and its confidence score for a transcription job where you've identified the language, use the GetTranscriptionJob (p. 265) operation. You can retrieve this information about a transcription job during processing. You don't need to wait for a transcription to complete to get this information. The following AWS CLI request gets information about a specific transcription job, as shown in the example response.

```bash
aws transcribe get-transcription-job \
  --transcription-job-name your-transcription-job
```
To list the transcription jobs that have automatic language identification enabled, use the `ListTranscriptionJobs` operation. The language that Amazon Transcribe identified is represented by its language code in the `LanguageCode` parameter in the response. For transcription jobs with automatic language identification enabled, the `IdentifiedLanguageScore` parameter indicates how confident Amazon Transcribe is that it identified the correct language. Its value ranges between zero and one, with zero meaning no confidence and one meaning absolute confidence. The following AWS CLI command would return a response similar to the one that's shown.

```
aws transcribe list-transcription-jobs
```
Getting Notifications Using Automatic Language Identification

Amazon Transcribe can notify you if it successfully identifies the language of your media file before the transcription job finishes. To receive a notification, enable a rule for an Amazon CloudWatch event. CloudWatch is a service that you can use to monitor your AWS resources and applications in real time. For more information see What Is Amazon CloudWatch.

For more information about the CloudWatch events that show whether the language has been identified successfully in your audio, see Language Identification Events (p. 218). For general information about CloudWatch events in Amazon Transcribe, see Using Amazon CloudWatch Events with Amazon Transcribe (p. 216).
Use Vocabulary Filtering to Filter Unwanted Words

You can mask or remove words that you don't want to appear in your transcription results with **vocabulary filtering**. For example, you can use vocabulary filtering to prevent the display of offensive or profane terms. This enables you to generate family-friendly captions of a TV show or transcripts of conferences that are appropriate for your audiences. Use vocabulary filtering for any word that you consider profane, obscene, offensive, or otherwise unsuitable for the readers of your transcripts.

Vocabulary filtering is available for both real-time streaming and batch processing. For both transcription processing methods, you can either mask unwanted words (replace them with three asterisks (***) in your transcription) or remove them completely. For real-time streaming only, you can use tags to mark words that are listed in your vocabulary filter in your transcription results. 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.

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 to filter words from your transcript.

You can filter unwanted words with the Amazon Transcribe console or the API.

**Topics**
- Step 1: Creating a List of Unwanted Words  (p. 174)
- Step 2: Creating a Vocabulary Filter (p. 175)
- Step 3: Filtering Transcriptions (p. 176)

**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` operation 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` operation 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 operation, see Step 2: Creating a Vocabulary Filter (p. 175).

To create a list of unfiltered words (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 Simple Storage Service (Amazon S3).

Next Step

Step 2: Creating a Vocabulary Filter (p. 175)

Step 2: Creating a Vocabulary Filter

You can create a vocabulary filter with either the CreateVocabularyFilter (p. 242) operation or the Amazon Transcribe console.

If you use the CreateVocabularyFilter (p. 242) operation, 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.

Console

Creating a Vocabulary Filter (Console)

To use the 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. 174). Your file can be saved locally or in Amazon Simple Storage Service (Amazon S3).

To create a vocabulary filter (console)

1. Sign in to AWS Management Console and open the Amazon Transcribe console at Amazon Transcribe 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)

- In the CreateVocabularyFilter (p. 242) operation, 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 Simple Storage Service (Amazon S3) location of the text file in the VocabularyFilterFileUri parameter using this format: s3://DOC-EXAMPLE-BUCKET1/vocabulary-filter-example.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 (p. 289) operation. You can then use that information with the GetVocabularyFilter (p. 271) operation to retrieve the download URI for your vocabulary filter and learn more about that filter.

AWS CLI

The following is an example AWS Command Line Interface (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.

```bash
aws transcribe create-vocabulary-filter 
   --vocabulary-filter-name your-filter-name 
   --language-code en-US 
   --vocabulary-filter-file-uri s3://DOC-EXAMPLE-BUCKET1/vocabulary-filter-example.txt
```

```
{
   "VocabularyFilterName": "your-filter-name",
   "LanguageCode": "en-US"
}
```

Next Step

Step 3: Filtering Transcriptions (p. 176)

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 Streaming Transcription (p. 73). For information about batch transcriptions, see How Amazon Transcribe Works (p. 4).

Topics
• Filtering Batch Transcriptions (p. 177)
• Filtering Streaming Transcriptions (p. 178)

Filtering Batch Transcriptions

Use a vocabulary filter to filter unwanted words from a batch transcription job with either the Amazon Transcribe console or the `StartTranscriptionJob` (p. 297) operation.

The following operation shows the parameters and data types.

```json
{
    "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"
}
```

**Console**

To use the 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. 175).
To filter unwanted words in a transcription job (console)

1. Sign in to AWS Management Console and open the Amazon Transcribe console at Amazon Transcribe 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 audio file or video 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.
10. Choose Create.

API

To filter a batch transcription (API)

- In the StartTranscriptionJob (p. 297) operation, 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. In the MediaFileUri parameter of the Media object, specify the name of the media file that 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.

Filtering Streaming Transcriptions

Use a vocabulary filter to filter unwanted words in real-time streams with either the Amazon Transcribe console or the StartStreamTranscription (p. 319) operation.

The following syntax shows the parameters and their data types.

```json
{
    "LanguageCode" : "enum",
    "MediaSampleRateHertz" : "integer",
    "MediaEncoding" : "enum",
    "VocabularyName" : "string",
    "SessionId" : "string",
    "VocabularyFilterName" : "string",
    "VocabularyFilterMethod" : "enum"
}
```
To filter a streaming transcription (API)

- In the StartStreamTranscription (p. 319) operation, 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 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. 179).

To filter a streaming transcription (console)

1. Sign into the AWS Management Console and open the Amazon Transcribe console at Amazon Transcribe 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. In the StartStreamTranscription (p. 319) operation, 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

- In the StartStreamTranscription (p. 319) operation, specify the following:
  a. For VocabularyFilterName, the name of your vocabulary filter.
b. For VocabularyFilterMethod, specify tag.

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."

The following example JSON output shows this.

```
"Transcript": {
  "Results": [
    {
      "Alternatives": [
        {
          "Items": [
            {
              "Content": "jumps",
              "EndTime": 1.02,
              "StartTime": 0.98,
              "Type": "pronunciation",
              "VocabularyFilterMatch": false
            },
            {
              "Content": "over",
              "EndTime": 1.26,
              "StartTime": 1.03,
              "Type": "pronunciation",
              "VocabularyFilterMatch": false
            },
            {
              "Content": "the",
              "EndTime": 1.41,
              "StartTime": 1.27,
              "Type": "pronunciation",
              "VocabularyFilterMatch": false
            },
            {
              "Content": "lazy",
              "EndTime": 1.81,
              "StartTime": 1.42,
              "Type": "pronunciation",
              "VocabularyFilterMatch": true
            }
          ]
        }
      ]
    }
  ]
}
```
Identifying Speakers

To identify different speakers in Amazon Transcribe, use *speaker identification*. When you enable speaker identification, Amazon Transcribe labels each speaker utterance. You enable speaker identification by using the batch transcription or real-time streaming APIs, or the Amazon Transcribe console.

Speaker identification is available in US English (en-US).

**Topics**
- Identifying Speakers In Audio Files (p. 181)
- Identifying Speakers in Real-time Streams (p. 184)

Identifying Speakers In Audio Files

You can enable speaker identification in a batch transcription job using either the `StartTranscriptionJob` (p. 297) operation or the Amazon Transcribe console.

**Console**

To identify speakers in an audio file (console)

To use the console to enable speaker identification in your transcription job, you enable audio identification and then speaker identification.

1. Sign in to AWS Management Console and open the Amazon Transcribe console at Amazon Transcribe 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.
7. For Audio identification type, choose Speaker identification.
8. For Maximum number of speakers, specify the maximum 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 a speaker label.
9. Choose Create.

**API**

To identify speakers in an audio file using a batch transcription job (API)

- In the `StartTranscriptionJob` (p. 297) operation, 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. In the MediaFileUri parameter of the Media object, specify the name of the media file that you want to transcribe.
d. For the Settings object, specify the following.

i. ShowSpeakerLabels - true.

ii. MaxSpeakerLabels - An integer between 2 and 10 for that indicates 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 a speaker label.

The following syntax shows the request parameters to start a batch transcription job and their data types.

```json
{
    "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.

```json
{
    "jobName": "job ID",
    "accountId": "account ID",
    "results": {
        "transcripts": [
            {
                "transcript": "Professional answer."
            }
        ],
        "speaker_labels": {
            "speakers": 1,
            "segments": [
                {
                    "start_time": "0.000000",
                    "end_time": "1.000000"
                }
            
```
AWS CLI

To identify the speakers in an audio file using a batch transcription job (AWS CLI)

- Run the following code.
Identifying Speakers in Real-time Streams

You can identify different speakers in either HTTP/2 or Websocket streams. Speaker identification 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 identification decreases if you exceed that number. To start an HTTP/2 stream, you specify the `ShowSpeakerLabels` request parameter of the `StartStreamTranscription (p. 319)` operation. To start a Websocket request, you use a pre-signed URL, a URL that contains the information needed to start your stream. To use the 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).

To identify speakers in audio that is spoken into your microphone (console)

You can use the Amazon Transcribe console to start a real-time stream and transcribe any speech picked up by your microphone.
1. Sign into the AWS Management Console and open the Amazon Transcribe console at Amazon Transcribe console.
2. In the navigation pane, choose Real-time transcription.
3. In Language, choose the language of your real-time stream.
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
x-amzn-transcribe-language-code: LanguageCode
x-amzn-transcribe-sample-rate: MediaSampleRateHz
x-amzn-transcribe-media-encoding: MediaEncoding
x-amzn-transcribe-session-id: SessionId
x-amzn-transcribe-vocabulary-filter-name: VocabularyFilterName
x-amzn-transcribe-vocabulary-filter-method: VocabularyFilterMethod
x-amzn-transcribe-show-speaker-label: ShowSpeakerLabel
Content-type: application/json
{
    "AudioStream": {
        "AudioEvent": {
            "AudioChunk": blob
        }
    }
}
```

To identify speakers in an HTTP/2 stream, use the StartStreamTranscription (p. 319) and specify the following:

- For LanguageCode, specify the language code that corresponds to the language spoken in your audio file.
- For MediaSampleRateHz, specify the sample rate of the audio.
- ShowSpeakerLabel - true.

Websocket Streaming

To identify speakers in Websocket streams, use the following format to create a pre-signed URL to start a Websocket request and specify show-speaker-label as true. A pre-signed URL contains the information to set up bi-directional communication between your application and Amazon Transcribe.

```
GET wss://transcribestreaming.region.amazonaws.com:8443/stream-transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=Signature Version 4 credential scope
&X-Amz-Date=date
&X-Amz-Expires=time in seconds until expiration
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
```
&media-encoding=mediaEncoding
&sample-rate=mediaSampleRateHertz
&session-id=sessionId
&show-speaker-label=true

For more information on completing Websocket requests, see Creating a Pre-Signed URL (p. 78).

Streaming Transcription Output

The following code shows the truncated example response of a streaming request.

```json
{
  "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
          },
          ...
        ]
      }
    ]
  }
}
```
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 above 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.
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 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 - Console and StartTranscriptionJob (p. 297) operation
- Streaming transcription - WebSocket Streaming and StartStreamTranscription (p. 319) operation

Transcribing Multi-Channel Audio Files

To transcribe multi-channel audio in a batch transcription job, use the Amazon Transcribe console or the StartTranscriptionJob (p. 297) operation.

**Console**

To use the 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 console.

**To transcribe a multi-channel audio file (console)**

1. Sign in to AWS Management Console and open the Amazon Transcribe console at Amazon Transcribe 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.
7. For Audio identification type, choose Channel identification.
8. Choose Create.

**API**

To transcribe a multi-channel audio file (API)

- In the StartTranscriptionJob (p. 297) operation, 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. In 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).

```python
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe')
job_name = "your-transcription-job-name"
job_uri = "the-Amazon-S3-object-URL-of-your-media-file"
transcribe.start_transcription_job(
    TranscriptionJobName=job_name,
    Media= {'MediaFileUri': job_uri},
    MediaFormat= 'mp4',
    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**

To transcribe a multi-channel audio file using a batch transcription job (AWS CLI)

- Run the following code.

```
aws transcribe start-transcription-job \ 
   --cli-input-json file://example-start-command.json
```

The following is the code of example-start-command.json.

```json
{
    "TranscriptionJobName": "your-transcription-job-name",
    "LanguageCode": "en-US",
    "Media": {
        "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file.mp4"
    },
    "Settings": {
        "ChannelIdentification": true
    }
}
```
The following is the response.

```
{
  "TranscriptionJob": {
    "TranscriptionJobName": "your-transcription-job",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "en-US",
    "Media": {
      "MediaFileUri": "s3://DOC-EXAMPLE-BUCKET/your-audio-file"
    },
    "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 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"
  },
  ...
]"}
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.

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.

Transcribing Multi-Channel Audio Streams

You can transcribe audio from separate channels in either HTTP/2 or WebSocket streams using the StartStreamTranscription (p. 319) operation.

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 section called “StartStreamTranscription” (p. 319) operation and specify the following:

- LanguageCode - The language code of the audio.
- MediaEncoding - pcm
- 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
x-amzn-transcribe-language-code: LanguageCode
x-amzn-transcribe-sample-rate: MediaSampleRateHertz
x-amzn-transcribe-media-encoding: MediaEncoding
x-amzn-transcribe-session-id: SessionId
x-amzn-transcribe-vocabulary-filter-name: VocabularyFilterName
x-amzn-transcribe-vocabulary-filter-method: VocabularyFilterMethod
x-amzn-transcribe-enable-channel-identification: EnableChannelIdentification
x-amzn-transcribe-number-of-channels: NumberOfChannels
Content-type: application/json

{
    "AudioStream": {
        "AudioEvent": {
            "AudioChunk": blob
        }
    }
}
```
Transcribing Multi-Channel Audio in a WebSocket Stream

To identify speakers in WebSocket streams, use the following format to create a pre-signed URL 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 URL contains the information needed to set up bi-directional communication between your application and Amazon Transcribe.

```
GET wss://transcribestreaming.region.amazonaws.com:8443/stream-transcription-websocket
?language-code=languageCode
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=Signature Version 4 credential scope
&X-Amz-Date=date
&X-Amz-Expires=time in seconds until expiration
&X-Amz-Security-Token=security-token
&X-Amz-Signature=Signature Version 4 signature
&X-Amz-SignedHeaders=host
&media-encoding=mediaEncoding
&sample-rate=mediaSampleRateHertz
&sessionId=sessionId
&enable-channel-identification=true
&number-of-channels=number of channels in your audio stream
```

For more information about WebSocket requests, see Creating a Pre-Signed URL (p. 78).

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.
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. 196)
- Identity and Access Management for Amazon Transcribe (p. 201)
- Logging and Monitoring in Amazon Transcribe (p. 214)
- Compliance Validation for Amazon Transcribe (p. 219)
- Resilience in Amazon Transcribe (p. 219)
- Infrastructure Security in Amazon Transcribe (p. 220)

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 Simple Storage Service (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 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. 197)
• Encryption in Transit (p. 197)
• Key Management (p. 197)
• Opting Out of Using Your Data for Service Improvement (p. 198)
• Amazon Transcribe and interface VPC endpoints (AWS PrivateLink) (p. 199)

Encryption at Rest

Amazon Transcribe uses the default Amazon S3 key (SSE-S3) for server-side encryption of transcripts placed in your S3 bucket.

When you use the StartTranscriptionJob (p. 297) operation, you can specify your own AWS Key Management Service 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 AWS Key Management Service (KMS) to provide enhanced encryption for your data. Amazon S3 already enables you to encrypt your input audio when creating a transcription job. Integration with KMS enables you to encrypt the output of the StartTranscriptionJob (p. 297) operation.

If you don’t specify a customer master key (CMK), 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.

KMS Encryption with the AWS Console

To encrypt the output of your transcription job, you can choose between using a customer managed CMK for the account that is making the request, or you can use a CMK from another account.
If you don't specify a CMK, the output of the transcription job is encrypted with the default Amazon S3 key (SSE-S3).

**To enable output result encryption**

1. Under **Output data**, choose **Encryption**.
   - **Encryption**
     - Use key from current account
     - Use key from different account
   - **KMS key ID**
     - Choose key

2. Choose whether the KMS customer managed CMK 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.

### KMS Encryption with the API

To use output encryption with the API, you set the `OutputEncryptionKMSKeyId` parameter of the `StartTranscriptionJob` (p. 297) operation. You can use a customer managed CMK 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 AWS 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"

### 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 (p. 297) operation, 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 (p. 265) operation 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 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 (p. 283) operation.

To move transcripts to your own S3 buckets

1. In the TranscriptionJobName parameter of the GetTranscriptionJob (p. 265) operation, 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 and open the Amazon S3 console at Amazon S3 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 console or the AWS Command Line Interface (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:
For streaming transcription in Amazon Transcribe, create a VPC endpoint using the following service name:

• com.amazonaws.region.transcribestreaming

If you enable private DNS for the endpoint, you can make API requests to Amazon Transcribe using its default DNS name for the 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.

```json
{  
    "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.

```json
{  
    "Statement": [
      {
        "Principal": "*",
        "Effect": "Allow",
```
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. 201)
- Authenticating with Identities (p. 201)
- Managing Access Using Policies (p. 203)
- How Amazon Transcribe Works with IAM (p. 205)
- Amazon Transcribe Identity-Based Policy Examples (p. 208)
- Troubleshooting Amazon Transcribe Identity and Access (p. 212)

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. 212).

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. 205).

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. 208).

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.
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. 205)
- Troubleshooting Amazon Transcribe Identity and Access (p. 212)

### 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. 205)
- Amazon Transcribe IAM Roles (p. 208)

### 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 operation, 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.

**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 IAM 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.

<table>
<thead>
<tr>
<th>Amazon Transcribe Condition Key</th>
<th>Description</th>
<th>Value Type</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>transcribe:OutputBucketName</code></td>
<td>Filters access by the output bucket used to start a transcription job.</td>
<td>String</td>
<td><code>StartTranscriptionJob (p. 297)</code></td>
</tr>
<tr>
<td><code>transcribe:OutputEncryptionKMSKeyId</code></td>
<td>Filters access by the KMS key used to start a transcription job.</td>
<td>String</td>
<td><code>StartTranscriptionJob (p. 297)</code></td>
</tr>
<tr>
<td><code>transcribe:OutputKey</code></td>
<td>Filters access by the output key used to start a transcription job.</td>
<td>String</td>
<td><code>StartTranscriptionJob (p. 297)</code></td>
</tr>
</tbody>
</table>
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 (p. 297) operation, 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 AWS Key Management Service (KMS) key when they use the  
StartTranscriptionJob (p. 297) operation, 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 AWS KMS keys, see Key Management (p. 197).

If you want your users to always use a specific output key when they use the  
StartTranscriptionJob (p. 297) operation, you can use the following policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": ["transcribe:StartTranscriptionJob"],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "transcribe:OutputEncryptionKMSKeyId": "false"
        }
      }
    }
  ]
}
```
"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 (p. 297) operation.

Examples

For examples of Amazon Transcribe identity-based policies, see Amazon Transcribe Identity-Based Policy Examples (p. 208).

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) API operations, 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.

Amazon Transcribe doesn't support service roles.

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. 209)
- Using the Amazon Transcribe Console (p. 209)
- AWS Managed (Predefined) Policies for Amazon Transcribe (p. 210)
- Permissions Required for IAM User Roles (p. 210)
- Permissions Required for Amazon S3 Encryption Keys (p. 211)
- Allow Users to View Their Own Permissions (p. 211)

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 Amazon Transcribe Console

To access the Amazon Transcribe console, you must have a minimum set of permissions for the 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 console won't function as intended for entities (IAM users or roles) with that policy.

To ensure that those entities can use the Amazon Transcribe console, attach the following AWS managed policy to them.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
You don’t need to allow minimum 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 IAM User Guide.

The following AWS managed policies, which you can attach to users, roles, and groups in your account, are specific to Amazon Transcribe:

- **ReadOnly** – Grants read-only access to Amazon Transcribe resources so that you can get and list transcription jobs and custom vocabularies.
- **FullAccess** – Grants full access to create, read, update, delete, and run all Amazon Transcribe resources. It also allows access to Amazon Simple Storage Service (Amazon S3) buckets with `transcribe` in the bucket name.

  **Note**
  
  You can review the managed permission policies by signing in to the IAM console and searching by policy name.

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 S3 bucket and the AWS Key Management Service (AWS 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 KMS Amazon Resource Name (ARN):

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": ["kms:Decrypt"],
            "Resource": "KMS key ARN",
            "Effect": "Allow"
        }
    ]
}
```
The user's IAM policy must have Amazon S3 permissions to access the S3 bucket where audio files are stored and transcriptions are saved.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": "S3 bucket location"
    }
  ]
}
```

Permissions Required for Amazon S3 Encryption Keys

If you are using an AWS KMS key to encrypt an Amazon S3 bucket, include the following in the AWS KMS key policy. This gives Amazon Transcribe access to the contents of the bucket.

```json
{
  "Sid": "Allow-Transcribe",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::account id:root",
  },
  "Action": ["kms:Decrypt"],
  "Resource": "KMS key ARN"
}
```

For more information about allowing access to customer master keys, see Allowing External AWS Accounts to Access a CMK 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.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Resource": ["arn:aws:iam::*:user/${aws:username}"],
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": ["s3:GetObject"
```
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. 212)
- I Am Not Authorized to Perform iam:PassRole (p. 212)
- I Want to View My Access Keys (p. 213)
- I’m an Administrator and Want to Allow Others to Access Amazon Transcribe (p. 213)
- I Want to Allow People Outside of My AWS Account to Access My Amazon Transcribe Resources (p. 213)

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 IAM tries to use the 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 operation using the transcribe:GetTrascriptionJob 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.

212
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 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, `wJalrXUttnFEMI/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. 205).
- 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 CloudWatch Events.

Topics
• Monitoring Amazon Transcribe API Calls with AWS CloudTrail (p. 214)
• Using Amazon CloudWatch Events with Amazon Transcribe (p. 216)

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 Amazon Transcribe 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 Simple Storage Service (Amazon S3) bucket. If you don't create a trail, you can still view the most recent events in the CloudTrail console in Event history. Using the information collected by CloudTrail, you can see each request that was made to Amazon Transcribe, the IP address from which the request was made, who made the request, when it was 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 S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the 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 API Reference (p. 230). For example, calls to the CreateVocabulary, GetTranscriptionJob, and StartTranscriptionJob operations 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 was made with root or IAM user credentials
- Whether the request was made with temporary security credentials for an IAM role or federated user
- Whether the request was 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 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 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 operations create the following entry.

```json
{
  "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": "region",
      "sourceIPAddress": "source IP address",
      "userAgent": "user agent",
      "requestParameters": {
        "mediaFormat": "flac | mp3 | mp4 | wav",
        "languageCode": "en-US | es-US",
        "transcriptionJobName": "unique job name",
        "media": {
          "mediaFileUri": ""
        }
      },
      "responseElements": {
        "transcriptionJob": {
          "transcriptionJobStatus": "IN_PROGRESS",
          "mediaFormat": "flac | mp3 | mp4 | wav",
          "creationTime": "timestamp",
          "transcriptionJobName": "unique job name",
          "languageCode": "en-US | es-US",
          "media": {
            "mediaFileUri": ""
          }
        }
      }
    }
  ]
}
```
Using Amazon CloudWatch Events with Amazon Transcribe

With Amazon CloudWatch Events, you can respond to state changes in your Amazon Transcribe jobs by triggering events in other AWS services. When a transcription job changes state, CloudWatch Events 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 CloudWatch Events 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 CloudWatch Events, 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 CloudWatch Events 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.

For more information about creating and managing CloudWatch Events events, see What is Amazon CloudWatch Events in the Amazon CloudWatch User Guide.

### Defining CloudWatch Events Rules

To define CloudWatch Events rules, use the CloudWatch Events console. When you define a rule, use Amazon Transcribe as the service name. For an example of how to create a CloudWatch Events rule, see Creating a CloudWatch Events Rule That Triggers on an Event in the Amazon CloudWatch User Guide.

The following is an example of a CloudWatch Events rule for Amazon Transcribe. It's triggered when a transcription job's status changes to COMPLETED or FAILED.

```json
{
  "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 `TranscriptionJobStatus` field of the TranscriptionJob (p. 344) data type.

### Amazon Transcribe Events

Amazon CloudWatch logs two kinds of Amazon Transcribe events: transcription job events and automatic language identification events.

#### 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.
The event passed to the target contains the following information:

- **version** – The version of the event data. This value is always 0.
- **id** – A unique identifier generated by CloudWatch Events for the event.
- **detail-type** – An identifier for the details of the event. For Amazon Transcribe, this is either `Transcribe Job State Change` or `Language Identification State Change`.
- **source** – The source of the event. For Amazon Transcribe this is always `aws.transcribe`.
- **account ID** – The AWS account ID of the account that generated the API call.
- **timestamp** – The date and time that the API call was made.
- **region** – The AWS Region where the API call was made.
- **resources** – The resources used by the API call. For Amazon Transcribe, this field is always empty.
- **detail** – Details about the event. It contains the following fields:
  - **TranscriptionJobName** – The unique name that you gave the job.
  - **TranscriptionJobStatus** – The new status of the transcription job. For a list of status values, see the `TranscriptionJobStatus` field of the `TranscriptionJob` data type.

### 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:

```
{
  "version": "0",
  "id": "event ID",
  "detail-type": "Language Identification State Change",
  "source": "aws.transcribe",
  "account": "account ID",
  "time": "timestamp",
  "region": "region",
  "resources": [],
  "detail": {
    "JobType": "TranscriptionJob",
    "JobName": "unique job name",
    "LanguageIdentificationStatus": "status"
  }
}
```

The event passed to the target contains the following information:
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.
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 AWS Identity and Access Management (IAM) principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.
Guidelines and Quotas

Supported Regions

For a list of AWS Regions where Amazon Transcribe is available, see Amazon Transcribe Endpoints and Quotas in the Amazon Web Services General Reference.

Throttling

You can request a quota increase for the following resources:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of concurrent batch transcription jobs</td>
<td>100</td>
</tr>
<tr>
<td>Job queue bandwidth ratio</td>
<td>0.9</td>
</tr>
<tr>
<td>Transactions per second, \texttt{StartTranscriptionJob} operation</td>
<td>10</td>
</tr>
<tr>
<td>Number of concurrent HTTP/2 streams for streaming transcription</td>
<td>5</td>
</tr>
<tr>
<td>Number of \texttt{StartStreamTranscription} Websocket requests</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second, \texttt{StartStreamTranscription} operation</td>
<td>5</td>
</tr>
<tr>
<td>Total number of vocabularies per account</td>
<td>100</td>
</tr>
<tr>
<td>Number of pending vocabularies</td>
<td>10</td>
</tr>
<tr>
<td>Number of concurrently training custom language models</td>
<td>3</td>
</tr>
<tr>
<td>Total number of custom language models per account</td>
<td>10</td>
</tr>
<tr>
<td>Number of channels for channel identification</td>
<td>2</td>
</tr>
<tr>
<td>Resource</td>
<td>Default</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>Transactions per second, GetTranscriptionJob operation</td>
<td>20</td>
</tr>
<tr>
<td>Transactions per second, DeleteTranscriptionJob operation</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second, ListTranscriptionJobs operation</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second, CreateVocabulary and UpdateVocabulary operations</td>
<td>10</td>
</tr>
<tr>
<td>Transactions per second, DeleteVocabulary operation</td>
<td>5</td>
</tr>
<tr>
<td>Transactions per second, GetVocabulary operation</td>
<td>20</td>
</tr>
<tr>
<td>Transactions per second, ListVocabularies operation</td>
<td>5</td>
</tr>
</tbody>
</table>

For information about requesting a quota increase, see AWS Service Quotas in the Amazon Web Services General Reference.

If you don’t need to process all of your transcription jobs concurrently, use Job Queuing (p. 47). 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.

**Guidelines**

For best results:

- Use a lossless format, such as FLAC or WAV, with PCM 16-bit encoding.
- Use a sample rate of 8000 Hz for low-fidelity audio and 16000 Hz for high-fidelity audio.

Amazon Transcribe may store your content to continuously improve the quality of its analysis models. See the Amazon Transcribe FAQ to learn more. To request that we delete content that may have been stored by Amazon Transcribe, open a case with AWS Support.
Quotas

Amazon Transcribe has the following quotas that are not alterable:

<table>
<thead>
<tr>
<th>Description</th>
<th>Quotas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum audio file length</td>
<td>4:00:00 (four) hours, 14,400 seconds</td>
</tr>
<tr>
<td>Maximum audio file size</td>
<td>2 GB</td>
</tr>
<tr>
<td>Maximum size of a custom vocabulary</td>
<td>50 KB</td>
</tr>
<tr>
<td>Maximum length of a custom vocabulary phrase</td>
<td>256 characters</td>
</tr>
<tr>
<td>Maximum size of a vocabulary filter</td>
<td>50 KB</td>
</tr>
<tr>
<td>Maximum number of vocabulary filters</td>
<td>100</td>
</tr>
<tr>
<td>Number of days that job records are retained</td>
<td>90</td>
</tr>
<tr>
<td>Minimum audio file duration, in milliseconds (ms)</td>
<td>500</td>
</tr>
</tbody>
</table>
## Document History for Amazon Transcribe and Amazon Transcribe Medical

- **Latest documentation update:** November 24th, 2020

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.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>New feature</td>
<td>Amazon Transcribe Medical now supports the HTTP/2 streaming protocol.</td>
<td>November 24, 2020</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Transcribe Medical now supports the neurology medical specialty for streaming audio transcription.</td>
<td>November 24, 2020</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Transcribe Medical now supports the urology medical specialty for streaming audio transcription.</td>
<td>November 24, 2020</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Transcribe Medical now supports the radiology medical specialty for streaming audio transcription.</td>
<td>November 24, 2020</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Transcribe Medical now supports the cardiology medical specialty for streaming audio transcription.</td>
<td>November 24, 2020</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Transcribe Medical now supports the oncology medical specialty for streaming audio transcription.</td>
<td>November 24, 2020</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Transcribe now supports OGG/OPUS and FLAC codecs for streaming audio transcription.</td>
<td>November 24, 2020</td>
</tr>
<tr>
<td>Region expansion</td>
<td>Amazon Transcribe is now available in the Frankfurt (eu-central-1) and London (eu-west-2) regions.</td>
<td>November 4, 2020</td>
</tr>
<tr>
<td><strong>New languages</strong></td>
<td>Amazon Transcribe adds support for Italian and German for streaming audio transcription.</td>
<td>November 4, 2020</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Transcribe adds support for interface VPC endpoints in batch transcription. For more information, see Amazon Transcribe and interface VPC endpoints (AWS PrivateLink).</td>
<td>October 9, 2020</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Transcribe adds support for channel identification in streaming. For more information, see Transcribing Multi-Channel Audio.</td>
<td>September 17, 2020</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Transcribe adds support for automatic language identification in batch transcription. For more information, see Identifying the Language.</td>
<td>September 15, 2020</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Transcribe adds support for speaker identification in streaming. For more information, see Filtering Streaming Transcriptions.</td>
<td>August 19, 2020</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Transcribe adds support for custom language models. For more information, see Improving Transcription Accuracy with Custom Models.</td>
<td>August 5, 2020</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Transcribe adds support for interface VPC endpoints in streaming. For more information, see Amazon Transcribe and interface VPC endpoints (AWS PrivateLink).</td>
<td>June 26, 2020</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Transcribe adds support for vocabulary filtering in streaming. For more information, see Filtering Streaming Transcriptions.</td>
<td>May 20, 2020</td>
</tr>
<tr>
<td><strong>New feature</strong></td>
<td>Amazon Transcribe Medical adds support for custom vocabularies in both batch processing and streaming. For more information, see Medical Custom Vocabularies.</td>
<td>April 29, 2020</td>
</tr>
<tr>
<td>Date</td>
<td>Feature Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>April 1, 2020</td>
<td>New feature</td>
<td>Amazon Transcribe Medical adds support for batch processing of audio files. For more information, see <a href="#">Batch Transcription Overview</a>.</td>
</tr>
<tr>
<td>February 26, 2020</td>
<td>New feature</td>
<td>Amazon Transcribe adds support for automatically redacting personally identifiable information. For more information, see <a href="#">Automatic Content Redaction</a>.</td>
</tr>
<tr>
<td>December 20, 2019</td>
<td>New feature</td>
<td>Amazon Transcribe adds support for creating a vocabulary of words to filter from a transcription. For more information, see <a href="#">Vocabulary Filtering</a>.</td>
</tr>
<tr>
<td>December 19, 2019</td>
<td>New feature</td>
<td>Amazon Transcribe adds support for queuing transcription jobs. For more information, see <a href="#">Job Queuing</a>.</td>
</tr>
<tr>
<td>November 21, 2019</td>
<td>Region expansion</td>
<td>Amazon Transcribe is now available in the Asia Pacific (Tokyo) (ap-northeast-1) region.</td>
</tr>
<tr>
<td>November 21, 2019</td>
<td>New languages</td>
<td>Amazon Transcribe adds support for Gulf Arabic, Hebrew, Japanese, Malay, Swiss German, Telugu, and Turkish.</td>
</tr>
<tr>
<td>November 20, 2019</td>
<td>New feature</td>
<td>Amazon Transcribe adds support for alternative transcriptions. For more information, see <a href="#">Alternative Transcriptions</a>.</td>
</tr>
<tr>
<td>November 12, 2019</td>
<td>New languages</td>
<td>Amazon Transcribe adds support for Dutch, Farsi, Indonesian, Irish English, Portuguese, Scottish English, Tamil, and Welsh English.</td>
</tr>
<tr>
<td>October 25, 2019</td>
<td>New language</td>
<td>Amazon Transcribe now supports streaming transcription for Australian English (en-AU).</td>
</tr>
<tr>
<td>October 9, 2019</td>
<td>Region expansion</td>
<td>Amazon Transcribe is now available in the China (Beijing) (cn-north-1) and China (Ningxia) (cn-northwest-1) regions.</td>
</tr>
</tbody>
</table>
## Amazon Transcribe Developer Guide

<table>
<thead>
<tr>
<th>New feature</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Transcribe enables you to provide your</td>
<td>Your own AWS Key Management Service key to encrypt your transcription output files. For more information, see the <a href="https://docs.aws.amazon.com/transcribe/latest/dg/OutputEncryptionKMSKeyId.html">OutputEncryptionKMSKeyId</a> parameter of the StartStreamTranscription operation.</td>
<td>September 24, 2019</td>
</tr>
<tr>
<td>New languages</td>
<td>Amazon Transcribe adds support for Chinese Mandarin-Mainland and Russian.</td>
<td>August 23, 2019</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Transcribe adds support for streaming audio transcription using the WebSocket protocol. For more information, see <a href="https://docs.aws.amazon.com/transcribe/latest/dg/StreamingTranscription.html">Streaming Transcription</a>.</td>
<td>July 19, 2019</td>
</tr>
<tr>
<td>New feature</td>
<td>AWS CloudTrail now records events for the StartStreamTranscription operation.</td>
<td>July 19, 2019</td>
</tr>
<tr>
<td>Region expansion</td>
<td>Amazon Transcribe is now available in the US West (N. California) (us-west-1) region.</td>
<td>June 27, 2019</td>
</tr>
<tr>
<td>New language</td>
<td>Amazon Transcribe adds support for Modern Standard Arabic.</td>
<td>May 28, 2019</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Transcribe now transcribes numeric words into numbers for US English. For example, &quot;forty-two&quot; is transcribed as &quot;42&quot;. For more information, see <a href="https://docs.aws.amazon.com/transcribe/latest/dg/TranscribingNumbers.html">Transcribing Numbers</a>.</td>
<td>May 23, 2019</td>
</tr>
<tr>
<td>New language</td>
<td>Amazon Transcribe adds support for Hindi and Indian English.</td>
<td>May 15, 2019</td>
</tr>
<tr>
<td>New SDK</td>
<td>The AWS SDK for C++ now supports Amazon Transcribe.</td>
<td>May 8, 2019</td>
</tr>
<tr>
<td>New language</td>
<td>Amazon Transcribe adds support for Spanish.</td>
<td>April 19, 2019</td>
</tr>
<tr>
<td>Region expansion</td>
<td>Amazon Transcribe is now available in the EU (Frankfurt) (eu-central-1) and Asia Pacific (Seoul) (ap-northeast-2) regions.</td>
<td>April 18, 2019</td>
</tr>
<tr>
<td>New language</td>
<td>Amazon Transcribe adds support for streaming transcription in British English, French, and Canadian French.</td>
<td>April 5, 2019</td>
</tr>
<tr>
<td>New Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>New feature</td>
<td>The AWS SDK for Ruby V3 now supports Amazon Transcribe</td>
<td>March 25, 2019</td>
</tr>
<tr>
<td>New feature</td>
<td>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 Custom Vocabularies.</td>
<td>March 25, 2019</td>
</tr>
<tr>
<td>New languages</td>
<td>Amazon Transcribe adds support for German and Korean.</td>
<td>March 22, 2019</td>
</tr>
<tr>
<td>Region expansion</td>
<td>Amazon Transcribe is now available in the South America (Sao Paulo) (sa-east-1) region.</td>
<td>February 7, 2019</td>
</tr>
<tr>
<td>New language</td>
<td>Amazon Transcribe now supports streaming transcription for US Spanish (es-US).</td>
<td>February 7, 2019</td>
</tr>
<tr>
<td>Region expansion</td>
<td>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) regions.</td>
<td>January 24, 2019</td>
</tr>
<tr>
<td>New languages</td>
<td>Amazon Transcribe adds support for French, Italian, and Brazilian Portuguese.</td>
<td>December 20, 2018</td>
</tr>
<tr>
<td>New feature</td>
<td>Amazon Transcribe now supports transcription of audio streams. For more information, see Streaming Transcription.</td>
<td>November 19, 2018</td>
</tr>
<tr>
<td>New languages</td>
<td>Amazon Transcribe adds support for Australian English, British English, and Canadian French.</td>
<td>November 15, 2018</td>
</tr>
<tr>
<td>Region expansion</td>
<td>Amazon Transcribe is now available in Canada (Central) (ca-central-1) and Asia Pacific (Sydney) (ap-southeast-2).</td>
<td>July 17, 2018</td>
</tr>
<tr>
<td>New feature</td>
<td>You can now specify your own location to store the output from a transcription job. For more information, see the TranscriptionJobSummary data type.</td>
<td>July 11, 2018</td>
</tr>
<tr>
<td>New feature</td>
<td>Added AWS CloudTrail and Amazon CloudWatch Events integration. For more information, see Monitoring Amazon Transcribe.</td>
<td>June 28, 2018</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>New feature (p. 224)</td>
<td>Amazon Transcribe adds support for custom vocabularies. For more information, see Create a Custom Vocabulary.</td>
<td>April 4, 2018</td>
</tr>
<tr>
<td>New guide (p. 224)</td>
<td>This is the first release of the Amazon Transcribe Developer Guide.</td>
<td>November 29, 2017</td>
</tr>
</tbody>
</table>
API Reference

This section contains the API Reference documentation.

Topics

- Actions (p. 230)
- Data Types (p. 324)

Actions

The following actions are supported by Amazon Transcribe Service:

- CreateLanguageModel (p. 232)
- CreateMedicalVocabulary (p. 235)
- CreateVocabulary (p. 238)
- CreateVocabularyFilter (p. 242)
- DeleteLanguageModel (p. 245)
- DeleteMedicalTranscriptionJob (p. 247)
- DeleteMedicalVocabulary (p. 249)
- DeleteTranscriptionJob (p. 251)
- DeleteVocabulary (p. 253)
- DeleteVocabularyFilter (p. 255)
- DescribeLanguageModel (p. 257)
- GetMedicalTranscriptionJob (p. 259)
- GetMedicalVocabulary (p. 262)
- GetTranscriptionJob (p. 265)
- GetVocabulary (p. 268)
- GetVocabularyFilter (p. 271)
- ListLanguageModels (p. 274)
- ListMedicalTranscriptionJobs (p. 277)
- ListMedicalVocabularies (p. 280)
- ListTranscriptionJobs (p. 283)
- ListVocabularies (p. 286)
- ListVocabularyFilters (p. 289)
- StartMedicalTranscriptionJob (p. 292)
- StartTranscriptionJob (p. 297)
- UpdateMedicalVocabulary (p. 303)
- UpdateVocabulary (p. 306)
- UpdateVocabularyFilter (p. 310)

The following actions are supported by Amazon Transcribe Streaming Service:

- StartMedicalStreamTranscription (p. 313)
• StartStreamTranscription (p. 319)

Amazon Transcribe Service

The following actions are supported by Amazon Transcribe Service:

- CreateLanguageModel (p. 232)
- CreateMedicalVocabulary (p. 235)
- CreateVocabulary (p. 238)
- CreateVocabularyFilter (p. 242)
- DeleteLanguageModel (p. 245)
- DeleteMedicalTranscriptionJob (p. 247)
- DeleteMedicalVocabulary (p. 249)
- DeleteTranscriptionJob (p. 251)
- DeleteVocabulary (p. 253)
- DeleteVocabularyFilter (p. 255)
- DescribeLanguageModel (p. 257)
- GetMedicalTranscriptionJob (p. 259)
- GetMedicalVocabulary (p. 262)
- GetTranscriptionJob (p. 265)
- GetVocabulary (p. 268)
- GetVocabularyFilter (p. 271)
- ListLanguageModels (p. 274)
- ListMedicalTranscriptionJobs (p. 277)
- ListMedicalVocabularies (p. 280)
- ListTranscriptionJobs (p. 283)
- ListVocabularies (p. 286)
- ListVocabularyFilters (p. 289)
- StartMedicalTranscriptionJob (p. 292)
- StartTranscriptionJob (p. 297)
- UpdateMedicalVocabulary (p. 303)
- UpdateVocabulary (p. 306)
- UpdateVocabularyFilter (p. 310)
CreateLanguageModel
Service: Amazon Transcribe Service

Creates a new custom language model. Use Amazon S3 prefixes to provide the location of your input files. The time it takes to create your model depends on the size of your training data.

Request Syntax

```
{
  "BaseModelName": "string",
  "InputDataConfig": {
    "DataAccessRoleArn": "string",
    "S3Uri": "string",
    "TuningDataS3Uri": "string"
  },
  "LanguageCode": "string",
  "ModelName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**BaseModelName (p. 232)**

The Amazon Transcribe standard language model, or base model used to create your custom language model.

If you want to use your custom language model to transcribe audio with a sample rate of 16 kHz or greater, choose Wideband.

If you want to use your custom language model to transcribe audio with a sample rate that is less than 16 kHz, choose Narrowband.

Type: String

Valid Values: NarrowBand | WideBand

Required: Yes

**InputDataConfig (p. 232)**

Contains the data access role and the Amazon S3 prefixes to read the required input files to create a custom language model.

Type: InputDataConfig (p. 327) object

Required: Yes

**LanguageCode (p. 232)**

The language of the input text you're using to train your custom language model.

Type: String

Valid Values: en-US

Required: Yes
**ModelName (p. 232)**

The name you choose for your custom language model when you create it.

Type: String


Pattern: `^[0-9a-zA-Z._-]+$`

Required: Yes

**Response Syntax**

```json
{
    "BaseModelName": "string",
    "InputDataConfig": {
        "DataAccessRoleArn": "string",
        "S3Uri": "string",
        "TuningDataS3Uri": "string"
    },
    "LanguageCode": "string",
    "ModelName": "string",
    "ModelStatus": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**BaseModelName (p. 233)**

The Amazon Transcribe standard language model, or base model you've used to create a custom language model.

Type: String

Valid Values: NarrowBand | WideBand

**InputDataConfig (p. 233)**

The data access role and Amazon S3 prefixes you've chosen to create your custom language model.

Type: InputDataConfig (p. 327) object

**LanguageCode (p. 233)**

The language code of the text you've used to create a custom language model.

Type: String

Valid Values: en-US

**ModelName (p. 233)**

The name you've chosen for your custom language model.

Type: String

ModelStatus (p. 233)

The status of the custom language model. When the status is COMPLETED the model is ready to use.

Type: String

Valid Values: IN_PROGRESS | FAILED | COMPLETED

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

ConflictException

There is already a resource with that name.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateMedicalVocabulary
Service: Amazon Transcribe Service

Creates a new custom vocabulary that you can use to change how Amazon Transcribe Medical transcribes your audio file.

Request Syntax

```
{
   "LanguageCode": "string",
   "VocabularyFileUri": "string",
   "VocabularyName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**LanguageCode (p. 235)**

The language code for the language used for the entries in your custom vocabulary. The language code of your custom vocabulary must match the language code of your transcription job. US English (en-US) is the only language code available for Amazon Transcribe Medical.

Type: String


Required: Yes

**VocabularyFileUri (p. 235)**

The location in Amazon S3 of the text file you use to define your custom vocabulary. The URI must be in the same AWS Region as the resource that you're calling. Enter information about your VocabularyFileUri in the following format:

https://s3.<aws-region>.amazonaws.com/<bucket-name>/<keyprefix>/<objectkey>

The following is an example URI for a vocabulary file that is stored in Amazon S3:

https://s3.us-east-1.amazonaws.com/AWSDOC-EXAMPLE-BUCKET/vocab.txt

For more information about Amazon S3 object names, see Object Keys in the Amazon S3 Developer Guide.

For more information about custom vocabularies, see Medical Custom Vocabularies.

Type: String


Pattern: (s3://|http(s*)://)+

Required: Yes
VocabularyName (p. 235)

The name of the custom vocabulary. This case-sensitive name must be unique within an AWS account. If you try to create a vocabulary with the same name as a previous vocabulary, you get a ConflictException error.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Syntax

```
{
  "FailureReason": "string",
  "LanguageCode": "string",
  "LastModifiedTime": number,
  "VocabularyName": "string",
  "VocabularyState": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

FailureReason (p. 236)

If the VocabularyState field is FAILED, this field contains information about why the job failed.

Type: String

LanguageCode (p. 236)

The language code for the entries in your custom vocabulary. US English (en-US) is the only valid language code for Amazon Transcribe Medical.

Type: String


LastModifiedTime (p. 236)

The date and time that you created the vocabulary.

Type: Timestamp

VocabularyName (p. 236)

The name of the vocabulary. The name must be unique within an AWS account and is case sensitive.

Type: String

Pattern: ^[0-9a-zA-Z._-]+$  

**VocabularyState (p. 236)**

The processing state of your custom vocabulary in Amazon Transcribe Medical. If the state is **READY**, you can use the vocabulary in a `StartMedicalTranscriptionJob` request.

Type: String

Valid Values: PENDING | READY | FAILED

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

**ConflictException**

There is already a resource with that name.

HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateVocabulary
Service: Amazon Transcribe Service

Creates a new custom vocabulary that you can use to change the way Amazon Transcribe handles transcription of an audio file.

Request Syntax

```json
{
   "LanguageCode": "string",
   "Phrases": [ "string" ],
   "VocabularyFileUri": "string",
   "VocabularyName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**LanguageCode (p. 238)**

The language code of the vocabulary entries.

Type: String


Required: Yes

**Phrases (p. 238)**

An array of strings that contains the vocabulary entries.

Type: Array of strings

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: .+

Required: No

**VocabularyFileUri (p. 238)**

The S3 location of the text file that contains the definition of the custom vocabulary. The URI must be in the same region as the API endpoint that you are calling. The general form is


For example:

https://s3.us-east-1.amazonaws.com/AWSDOC-EXAMPLE-BUCKET/vocab.txt

For more information about S3 object names, see Object Keys in the Amazon S3 Developer Guide.
For more information about custom vocabularies, see Custom Vocabularies.

**VocabularyName (p. 238)**

The name of the vocabulary. The name must be unique within an AWS account. The name is case sensitive. If you try to create a vocabulary with the same name as a previous vocabulary you will receive a ConflictException error.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

**Response Syntax**

```
{
  "FailureReason": "string",
  "LanguageCode": "string",
  "LastModifiedTime": number,
  "VocabularyName": "string",
  "VocabularyState": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**FailureReason (p. 239)**

If the VocabularyState field is FAILED, this field contains information about why the job failed.

Type: String

**LanguageCode (p. 239)**

The language code of the vocabulary entries.

Type: String


**LastModifiedTime (p. 239)**

The date and time that the vocabulary was created.

Type: Timestamp
VocabularyName (p. 239)

The name of the vocabulary.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

VocabularyState (p. 239)

The processing state of the vocabulary. When the VocabularyState field contains READY the vocabulary is ready to be used in a StartTranscriptionJob request.

Type: String

Valid Values: PENDING | READY | FAILED

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

ConflictException

There is already a resource with that name.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

• AWS Command Line Interface
• AWS SDK for .NET
• AWS SDK for C++
• AWS SDK for C
• AWS SDK for Java V2
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
CreateVocabularyFilter

Service: Amazon Transcribe Service

Creates a new vocabulary filter that you can use to filter words, such as profane words, from the output of a transcription job.

Request Syntax

```
{
    "LanguageCode": "string",
    "VocabularyFilterFileUri": "string",
    "VocabularyFilterName": "string",
    "Words": [ "string" ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

LanguageCode (p. 242)

The language code of the words in the vocabulary filter. All words in the filter must be in the same language. The vocabulary filter can only be used with transcription jobs in the specified language.

Type: String


Required: Yes

VocabularyFilterFileUri (p. 242)

The Amazon S3 location of a text file used as input to create the vocabulary filter. Only use characters from the character set defined for custom vocabularies. For a list of character sets, see Character Sets for Custom Vocabularies.

The specified file must be less than 50 KB of UTF-8 characters.

If you provide the location of a list of words in the VocabularyFilterFileUri parameter, you can't use the Words parameter.

Type: String


Pattern: (s3://|http(s*)://).+

Required: No

VocabularyFilterName (p. 242)

The vocabulary filter name. The name must be unique within the account that contains it. If you try to create a vocabulary filter with the same name as another vocabulary filter, you get a ConflictException error.
Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Words (p. 242)

The words to use in the vocabulary filter. Only use characters from the character set defined for
custom vocabularies. For a list of character sets, see Character Sets for Custom Vocabularies.

If you provide a list of words in the Words parameter, you can't use the
VocabularyFilterFileUri parameter.

Type: Array of strings

Array Members: Minimum number of 1 item.

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

Response Syntax

```json
{
  "LanguageCode": "string",
  "LastModifiedTime": number,
  "VocabularyFilterName": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

LanguageCode (p. 243)

The language code of the words in the collection.

Type: String


LastModifiedTime (p. 243)

The date and time that the vocabulary filter was modified.

Type: Timestamp

VocabularyFilterName (p. 243)

The name of the vocabulary filter.

Type: String

Errors
For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException
Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception message field for more information.

HTTP Status Code: 400

ConflictException
There is already a resource with that name.

HTTP Status Code: 400

InternalFailureException
There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException
Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteLanguageModel
Service: Amazon Transcribe Service

Deletes a custom language model using its name.

Request Syntax

```json
{
   "ModelName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**ModelName (p. 245)**

The name of the model you're choosing to delete.

- Type: String
- Pattern: `^[0-9a-zA-Z._-]+$`
- Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception message field for more information.

- HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

- HTTP Status Code: 500

**LimitExceededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

- HTTP Status Code: 400
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteMedicalTranscriptionJob
Service: Amazon Transcribe Service

Deletes a transcription job generated by Amazon Transcribe Medical and any related information.

Request Syntax

```json
{
    "MedicalTranscriptionJobName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**MedicalTranscriptionJobName (p. 247)**

The name you provide to the DeleteMedicalTranscriptionJob object to delete a transcription job.

Type: String


Pattern: ^[0-9a-zA-Z_.-]+$

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteMedicalVocabulary
Service: Amazon Transcribe Service

Deletes a vocabulary from Amazon Transcribe Medical.

Request Syntax

```json
{
   "VocabularyName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**VocabularyName (p. 249)**

The name of the vocabulary that you want to delete.

Type: String


Pattern: `^[0-9a-zA-Z._-]+$`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceedededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400
NotFoundException

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteTranscriptionJob
Service: Amazon Transcribe Service

Deletes a previously submitted transcription job along with any other generated results such as the transcription, models, and so on.

Request Syntax

```
{
   "TranscriptionJobName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**TranscriptionJobName (p. 251)**

The name of the transcription job to be deleted.

- Type: String
- Pattern: ^[0-9a-zA-Z._-]+$
- Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteVocabulary
Service: Amazon Transcribe Service

Deletes a vocabulary from Amazon Transcribe.

Request Syntax

```json
{
    "VocabularyName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**VocabularyName (p. 253)**

The name of the vocabulary to delete.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception message field for more information.

HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400
NotFoundException

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteVocabularyFilter
Service: Amazon Transcribe Service

Removes a vocabulary filter.

Request Syntax

```json
{
  "VocabularyFilterName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**VocabularyFilterName (p. 255)**

The name of the vocabulary filter to remove.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400
NotFoundException

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeLanguageModel
Service: Amazon Transcribe Service

Gets information about a single custom language model. Use this information to see details about the language model in your AWS account. You can also see whether the base language model used to create your custom language model has been updated. If Amazon Transcribe has updated the base model, you can create a new custom language model using the updated base model. If the language model wasn't created, you can use this operation to understand why Amazon Transcribe couldn't create it.

Request Syntax

```json
{
   "ModelName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**ModelName (p. 257)**

The name of the custom language model you submit to get more information.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Syntax

```json
{
   "LanguageModel": {
      "BaseModelName": "string",
      "CreateTime": number,
      "FailureReason": "string",
      "InputDataConfig": {
         "DataAccessRoleArn": "string",
         "S3Uri": "string",
         "TuningDataS3Uri": "string"
      },
      "LanguageCode": "string",
      "LastModifiedTime": number,
      "ModelName": "string",
      "ModelStatus": "string",
      "UpgradeAvailability": boolean
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.
The following data is returned in JSON format by the service.

**LanguageModel (p. 257)**

The name of the custom language model you requested more information about.

Type: LanguageModel (p. 329) object

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

**NotFoundException**

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
GetMedicalTranscriptionJob
Service: Amazon Transcribe Service

Returns information about a transcription job from Amazon Transcribe Medical. To see the status of the job, check the TranscriptionJobStatus field. If the status is COMPLETED, the job is finished. You find the results of the completed job in the TranscriptFileUri field.

Request Syntax

```json
{
    "MedicalTranscriptionJobName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

MedicalTranscriptionJobName (p. 259)

The name of the medical transcription job.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Syntax

```json
{
    "MedicalTranscriptionJob": {
        "CompletionTime": number,
        "CreationTime": number,
        "FailureReason": "string",
        "LanguageCode": "string",
        "Media": {
            "MediaFileUri": "string"
        },
        "MediaFormat": "string",
        "MediaSampleRateHertz": number,
        "MedicalTranscriptionJobName": "string",
        "Settings": {
            "ChannelIdentification": boolean,
            "MaxAlternatives": number,
            "MaxSpeakerLabels": number,
            "ShowAlternatives": boolean,
            "ShowSpeakerLabels": boolean,
            "VocabularyName": "string"
        },
        "Specialty": "string",
        "StartTime": number,
        "Transcript": {
            "TranscriptFileUri": "string"
        }
    }
}
```
"TranscriptionJobStatus": "string",
"Type": "string"
}
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

MedicalTranscriptionJob (p. 259)

An object that contains the results of the medical transcription job.

Type: MedicalTranscriptionJob (p. 333) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

NotFoundException

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
GetMedicalVocabulary
Service: Amazon Transcribe Service

Retrieves information about a medical vocabulary.

Request Syntax

```json
{
  "VocabularyName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

VocabularyName (p. 262)

The name of the vocabulary that you want information about. The value is case sensitive.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Syntax

```json
{
  "DownloadUri": "string",
  "FailureReason": "string",
  "LanguageCode": "string",
  "LastModifiedTime": number,
  "VocabularyName": "string",
  "VocabularyState": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

DownloadUri (p. 262)

The location in Amazon S3 where the vocabulary is stored. Use this URI to get the contents of the vocabulary. You can download your vocabulary from the URI for a limited time.

Type: String


Pattern: (s3://|http(s*)://).+
FailureReason (p. 262)

If the VocabularyState is FAILED, this field contains information about why the job failed.

Type: String

LanguageCode (p. 262)

The valid language code for your vocabulary entries.

Type: String


LastModifiedTime (p. 262)

The date and time that the vocabulary was last modified with a text file different from the one that was previously used.

Type: Timestamp

VocabularyName (p. 262)

The name of the vocabulary returned by Amazon Transcribe Medical.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

VocabularyState (p. 262)

The processing state of the vocabulary. If the VocabularyState is READY then you can use it in the StartMedicalTranscriptionJob operation.

Type: String

Valid Values: PENDING | READY | FAILED

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceeded Exception

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.
HTTP Status Code: 400

**NotFoundException**

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
GetTranscriptionJob
Service: Amazon Transcribe Service

Returns information about a transcription job. To see the status of the job, check the TranscriptionJobStatus field. If the status is COMPLETED, the job is finished and you can find the results at the location specified in the TranscriptFileUri field. If you enable content redaction, the redacted transcript appears in RedactedTranscriptFileUri.

Request Syntax

```json
{
  "TranscriptionJobName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

TranscriptionJobName (p. 265)

The name of the job.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Syntax

```json
{
  "TranscriptionJob": {
    "CompletionTime": number,
    "ContentRedaction": {
      "RedactionOutput": "string",
      "RedactionType": "string"
    },
    "CreationTime": number,
    "FailureReason": "string",
    "IdentifiedLanguageScore": number,
    "IdentifyLanguage": boolean,
    "JobExecutionSettings": {
      "AllowDeferredExecution": boolean,
      "DataAccessRoleArn": "string"
    },
    "LanguageCode": "string",
    "LanguageOptions": [ "string" ],
    "Media": {
      "MediaFileUri": "string"
    },
    "MediaFormat": "string",
    "MediaSampleRateHertz": number,
    "ModelSettings": {
      "LanguageModelName": "string"
    }
  }
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

TranscriptionJob (p. 265)

An object that contains the results of the transcription job.

Type: TranscriptionJob (p. 344) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

NotFoundException

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
GetVocabulary
Service: Amazon Transcribe Service

Gets information about a vocabulary.

Request Syntax

```json
{
   "VocabularyName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

VocabularyName (p. 268)

The name of the vocabulary to return information about. The name is case sensitive.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Syntax

```json
{
   "DownloadUri": "string",
   "FailureReason": "string",
   "LanguageCode": "string",
   "LastModifiedTime": number,
   "VocabularyName": "string",
   "VocabularyState": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

DownloadUri (p. 268)

The S3 location where the vocabulary is stored. Use this URI to get the contents of the vocabulary. The URI is available for a limited time.

Type: String


Pattern: (s3://|http(s*)://)+
FailureReason (p. 268)

If the VocabularyState field is FAILED, this field contains information about why the job failed.

Type: String

LanguageCode (p. 268)

The language code of the vocabulary entries.

Type: String


LastModifiedTime (p. 268)

The date and time that the vocabulary was last modified.

Type: Timestamp

VocabularyName (p. 268)

The name of the vocabulary to return.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

VocabularyState (p. 268)

The processing state of the vocabulary.

Type: String

Valid Values: PENDING | READY | FAILED

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400
NotFoundException

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
GetVocabularyFilter
Service: Amazon Transcribe Service

Returns information about a vocabulary filter.

Request Syntax

```json
{
   "VocabularyFilterName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

VocabularyFilterName (p. 271)

The name of the vocabulary filter for which to return information.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Syntax

```json
{
   "DownloadUri": "string",
   "LanguageCode": "string",
   "LastModifiedTime": number,
   "VocabularyFilterName": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

DownloadUri (p. 271)

The URI of the list of words in the vocabulary filter. You can use this URI to get the list of words.

Type: String


Pattern: (s3://|http(s*)://).+

LanguageCode (p. 271)

The language code of the words in the vocabulary filter.
Type: String

Valid Values: af-ZA | ar-AE | ar-SA | cy-GB | da-DK | de-CH | de-DE | en-AB |
fr-FR | ga-IE | gd-GB | he-IL | hi-IN | id-ID | it-IT | ja-JP | ko-KR |
ms-MY | nl-NL | pt-BR | pt-PT | ru-RU | ta-IN | te-IN | tr-TR | zh-CN

LastModifiedTime (p. 271)

The date and time that the contents of the vocabulary filter were updated.

Type: Timestamp

VocabularyFilterName (p. 271)

The name of the vocabulary filter.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn’t pass one or more validation tests. For example, if the entity that you’re trying to delete doesn’t exist or if it is in a non-terminal state (for example, it’s “in progress”). See the exception message field for more information.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

NotFoundException

We can’t find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
• AWS SDK for Java V2
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
ListLanguageModels
Service: Amazon Transcribe Service

Provides more information about the custom language models you've created. You can use the information in this list to find a specific custom language model. You can then use the DescribeLanguageModel (p. 257) operation to get more information about it.

Request Syntax

```json
{
   "MaxResults": number,
   "NameContains": "string",
   "NextToken": "string",
   "StatusEquals": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

MaxResults (p. 274)

The maximum number of language models to return in the response. If there are fewer results in the list, the response contains only the actual results.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

NameContains (p. 274)

When specified, the custom language model names returned contain the substring you've specified.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: No

NextToken (p. 274)

When included, fetches the next set of jobs if the result of the previous request was truncated.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

Required: No

StatusEquals (p. 274)

When specified, returns only custom language models with the specified status. Language models are ordered by creation date, with the newest models first. If you don't specify a status, Amazon Transcribe returns all custom language models ordered by date.

Type: String
Valid Values: IN_PROGRESS | FAILED | COMPLETED

Required: No

Response Syntax

```
{
   "Models": [
      {
         "BaseModelName": "string",
         "CreateTime": number,
         "FailureReason": "string",
         "InputDataConfig": {
            "DataAccessRoleArn": "string",
            "S3Uri": "string",
            "TuningDataS3Uri": "string"
         },
         "LanguageCode": "string",
         "LastModifiedTime": number,
         "ModelName": "string",
         "ModelStatus": "string",
         "UpgradeAvailability": boolean
      }
   ],
   "NextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Models (p. 275)

A list of objects containing information about custom language models.

Type: Array of LanguageModel (p. 329) objects

NextToken (p. 275)

The ListLanguageModels (p. 274) operation returns a page of jobs at a time. The maximum size of the list is set by the MaxResults parameter. If there are more language models in the list than the page size, Amazon Transcribe returns the NextPage token. Include the token in the next request to the ListLanguageModels (p. 274) operation to return the next page of language models.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.
HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
**ListMedicalTranscriptionJobs**

Service: Amazon Transcribe Service

Lists medical transcription jobs with a specified status or substring that matches their names.

**Request Syntax**

```json
{
    "JobNameContains": "string",
    "MaxResults": number,
    "NextToken": "string",
    "Status": "string"
}
```

**Request Parameters**

For information about the parameters that are common to all actions, see [Common Parameters](p. 374).

The request accepts the following data in JSON format.

**JobNameContains (p. 277)**

When specified, the jobs returned in the list are limited to jobs whose name contains the specified string.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: No

**MaxResults (p. 277)**

The maximum number of medical transcription jobs to return in the response. If there are fewer results in the list, this response contains only the actual results.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**NextToken (p. 277)**

If you receive a truncated result in the previous request of `ListMedicalTranscriptionJobs`, include `NextToken` to fetch the next set of jobs.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

Required: No

**Status (p. 277)**

When specified, returns only medical transcription jobs with the specified status. Jobs are ordered by creation date, with the newest jobs returned first. If you don't specify a status, Amazon Transcribe Medical returns all transcription jobs ordered by creation date.

Type: String
Valid Values: QUEUED | IN_PROGRESS | FAILED | COMPLETED

Required: No

Response Syntax

```json
{
    "MedicalTranscriptionJobSummaries": [
    {
        "CompletionTime": number,
        "CreationTime": number,
        "FailureReason": "string",
        "LanguageCode": "string",
        "MedicalTranscriptionJobName": "string",
        "OutputLocationType": "string",
        "Specialty": "string",
        "StartTime": number,
        "TranscriptionJobStatus": "string",
        "Type": "string"
    }
    ],
    "NextToken": "string",
    "Status": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**MedicalTranscriptionJobSummaries (p. 278)**

A list of objects containing summary information for a transcription job.

Type: Array of **MedicalTranscriptionJobSummary (p. 336)** objects

**NextToken (p. 278)**

The ListMedicalTranscriptionJobs operation returns a page of jobs at a time. The maximum size of the page is set by the MaxResults parameter. If the number of jobs exceeds what can fit on a page, Amazon Transcribe Medical returns the NextPage token. Include the token in the next request to the ListMedicalTranscriptionJobs operation to return in the next page of jobs.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

**Status (p. 278)**

The requested status of the medical transcription jobs returned.

Type: String

Valid Values: QUEUED | IN_PROGRESS | FAILED | COMPLETED

Errors

For information about the errors that are common to all actions, see **Common Errors (p. 373)**.
BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListMedicalVocabularies
Service: Amazon Transcribe Service

Returns a list of vocabularies that match the specified criteria. If you don't enter a value in any of the request parameters, returns the entire list of vocabularies.

Request Syntax

```json
{
   "MaxResults": number,
   "NameContains": "string",
   "NextToken": "string",
   "StateEquals": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

MaxResults (p. 280)

The maximum number of vocabularies to return in the response.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

NameContains (p. 280)

Returns vocabularies whose names contain the specified string. The search is not case sensitive. ListMedicalVocabularies returns both "vocabularyname" and "VocabularyName".

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: No

NextToken (p. 280)

If the result of your previous request to ListMedicalVocabularies was truncated, include the NextToken to fetch the next set of vocabularies.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

Required: No

StateEquals (p. 280)

When specified, returns only vocabularies with the VocabularyState equal to the specified vocabulary state. Use this field to see which vocabularies are ready for your medical transcription jobs.
Type: String

Valid Values: PENDING | READY | FAILED

Required: No

**Response Syntax**

```json
{
  "NextToken": "string",
  "Status": "string",
  "Vocabularies": [
    {
      "LanguageCode": "string",
      "LastModifiedTime": number,
      "VocabularyName": "string",
      "VocabularyState": "string"
    }
  ]
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**NextToken (p. 281)**

The `ListMedicalVocabularies` operation returns a page of vocabularies at a time. You set the maximum number of vocabularies to return on a page with the `MaxResults` parameter. If there are more jobs in the list that will fit on a page, Amazon Transcribe Medical returns the `NextPage` token. To return the next page of vocabularies, include the token in the next request to the `ListMedicalVocabularies` operation.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

**Status (p. 281)**

The requested vocabulary state.

Type: String

Valid Values: PENDING | READY | FAILED

**Vocabularies (p. 281)**

A list of objects that describe the vocabularies that match your search criteria.

Type: Array of `VocabularyInfo (p. 352)` objects

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 373).
**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceededException**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListTranscriptionJobs
Service: Amazon Transcribe Service

Lists transcription jobs with the specified status.

Request Syntax

```
{
  "JobNameContains": "string",
  "MaxResults": number,
  "NextToken": "string",
  "Status": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

JobNameContains (p. 283)

When specified, the jobs returned in the list are limited to jobs whose name contains the specified string.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: No

MaxResults (p. 283)

The maximum number of jobs to return in the response. If there are fewer results in the list, this response contains only the actual results.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

NextToken (p. 283)

If the result of the previous request to ListTranscriptionJobs was truncated, include the NextToken to fetch the next set of jobs.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

Required: No

Status (p. 283)

When specified, returns only transcription jobs with the specified status. Jobs are ordered by creation date, with the newest jobs returned first. If you don't specify a status, Amazon Transcribe returns all transcription jobs ordered by creation date.
Response Syntax

```json
{
    "NextToken": "string",
    "Status": "string",
    "TranscriptionJobSummaries": [
        {
            "CompletionTime": number,
            "ContentRedaction": {
                "RedactionOutput": "string",
                "RedactionType": "string"
            },
            "CreationTime": number,
            "FailureReason": "string",
            "IdentifiedLanguageScore": number,
            "IdentifyLanguage": boolean,
            "LanguageCode": "string",
            "ModelSettings": {
                "LanguageModelName": "string"
            },
            "OutputLocationType": "string",
            "StartTime": number,
            "TranscriptionJobName": "string",
            "TranscriptionJobStatus": "string"
        }
    ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**NextToken (p. 284)**

The ListTranscriptionJobs operation returns a page of jobs at a time. The maximum size of the page is set by the MaxResults parameter. If there are more jobs in the list than the page size, Amazon Transcribe returns the NextPage token. Include the token in the next request to the ListTranscriptionJobs operation to return in the next page of jobs.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

**Status (p. 284)**

The requested status of the jobs returned.

Type: String

Valid Values: QUEUED | IN_PROGRESS | FAILED | COMPLETED
TranscriptionJobSummaries (p. 284)

A list of objects containing summary information for a transcription job.

Type: Array of TranscriptionJobSummary (p. 348) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListVocabularies
Service: Amazon Transcribe Service

Returns a list of vocabularies that match the specified criteria. If no criteria are specified, returns the entire list of vocabularies.

Request Syntax

```json
{
    "MaxResults": number,
    "NameContains": "string",
    "NextToken": "string",
    "StateEquals": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**MaxResults (p. 286)**

The maximum number of vocabularies to return in the response. If there are fewer results in the list, this response contains only the actual results.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**NameContains (p. 286)**

When specified, the vocabularies returned in the list are limited to vocabularies whose name contains the specified string. The search is not case sensitive, ListVocabularies returns both "vocabularyname" and "VocabularyName" in the response list.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: No

**NextToken (p. 286)**

If the result of the previous request to ListVocabularies was truncated, include the NextToken to fetch the next set of jobs.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

Required: No
StateEquals (p. 286)

When specified, only returns vocabularies with the VocabularyState field equal to the specified state.

Type: String

Valid Values: PENDING | READY | FAILED

Required: No

Response Syntax

```
{
  "NextToken": "string",
  "Status": "string",
  "Vocabularies": [
    {
      "LanguageCode": "string",
      "LastModifiedTime": number,
      "VocabularyName": "string",
      "VocabularyState": "string"
    }
  ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken (p. 287)

The ListVocabularies operation returns a page of vocabularies at a time. The maximum size of the page is set in the MaxResults parameter. If there are more jobs in the list than will fit on the page, Amazon Transcribe returns the NextPage token. To return in the next page of jobs, include the token in the next request to the ListVocabularies operation.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

Status (p. 287)

The requested vocabulary state.

Type: String

Valid Values: PENDING | READY | FAILED

Vocabularies (p. 287)

A list of objects that describe the vocabularies that match the search criteria in the request.

Type: Array of VocabularyInfo (p. 352) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).
BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying

to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the

exception Message field for more information.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceeded Exception

Either you have sent too many requests or your input file is too long. Wait before you resend your

request, or use a smaller file and resend the request.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListVocabularyFilters
Service: Amazon Transcribe Service

Gets information about vocabulary filters.

Request Syntax

```json
{
    "MaxResults": number,
    "NameContains": "string",
    "NextToken": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

MaxResults (p. 289)

The maximum number of filters to return in the response. If there are fewer results in the list, this response contains only the actual results.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

NameContains (p. 289)

Filters the response so that it only contains vocabulary filters whose name contains the specified string.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: No

NextToken (p. 289)

If the result of the previous request to ListVocabularyFilters was truncated, include the NextToken to fetch the next set of collections.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

Required: No

Response Syntax

```json
{
}
```
"NextToken": "string",
"VocabularyFilters": [
  {
    "LanguageCode": "string",
    "LastModifiedTime": number,
    "VocabularyFilterName": "string"
  }
]

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

NextToken (p. 289)

The ListVocabularyFilters operation returns a page of collections at a time. The maximum size of the page is set by the MaxResults parameter. If there are more jobs in the list than the page size, Amazon Transcribe returns the NextPage token. Include the token in the next request to the ListVocabularyFilters operation to return in the next page of jobs.

Type: String

Length Constraints: Maximum length of 8192.

Pattern: .+

VocabularyFilters (p. 289)

The list of vocabulary filters. It contains at most MaxResults number of filters. If there are more filters, call the ListVocabularyFilters operation again with the NextToken parameter in the request set to the value of the NextToken field in the response.

Type: Array of VocabularyFilterInfo (p. 351) objects

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
StartMedicalTranscriptionJob

Service: Amazon Transcribe Service

Starts a batch job to transcribe medical speech to text.

Request Syntax

```
{
  "LanguageCode": "string",
  "Media": {
    "MediaFileUri": "string"
  },
  "MediaFormat": "string",
  "MediaSampleRateHertz": number,
  "MedicalTranscriptionJobName": "string",
  "OutputBucketName": "string",
  "OutputEncryptionKMSKeyId": "string",
  "OutputKey": "string",
  "Settings": {
    "ChannelIdentification": boolean,
    "MaxAlternatives": number,
    "MaxSpeakerLabels": number,
    "ShowAlternatives": boolean,
    "ShowSpeakerLabels": boolean,
    "VocabularyName": "string"
  },
  "Specialty": "string",
  "Type": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

LanguageCode (p. 292)

The language code for the language spoken in the input media file. US English (en-US) is the valid value for medical transcription jobs. Any other value you enter for language code results in a BadRequestException error.

Type: String


Required: Yes

Media (p. 292)

Describes the input media file in a transcription request.

Type: Media (p. 331) object

Required: Yes

MediaFormat (p. 292)

The audio format of the input media file.
Type: String
Valid Values: mp3 | mp4 | wav | flac | ogg | amr | webm
Required: No

**MediaSampleRateHertz (p. 292)**

The sample rate, in Hertz, of the audio track in the input media file.

If you do not specify the media sample rate, Amazon Transcribe Medical determines the sample rate. If you specify the sample rate, it must match the rate detected by Amazon Transcribe Medical. In most cases, you should leave the `MediaSampleRateHertz` field blank and let Amazon Transcribe Medical determine the sample rate.

Type: Integer
Required: No

**MedicalTranscriptionJobName (p. 292)**

The name of the medical transcription job. You can't use the strings "." or ".\" by themselves as the job name. The name must also be unique within an AWS account. If you try to create a medical transcription job with the same name as a previous medical transcription job, you get a ConflictException error.

Type: String
Pattern: `^[0-9a-zA-Z._-]+`
Required: Yes

**OutputBucketName (p. 292)**

The Amazon S3 location where the transcription is stored.

You must set `OutputBucketName` for Amazon Transcribe Medical to store the transcription results. Your transcript appears in the S3 location you specify. When you call the `GetMedicalTranscriptionJob` (p. 259), the operation returns this location in the `TranscriptFileUri` field. The S3 bucket must have permissions that allow Amazon Transcribe Medical to put files in the bucket. For more information, see Permissions Required for IAM User Roles.

You can specify an AWS Key Management Service (KMS) key to encrypt the output of your transcription using the `OutputEncryptionKeyId` parameter. If you don't specify a KMS key, Amazon Transcribe Medical uses the default Amazon S3 key for server-side encryption of transcripts that are placed in your S3 bucket.

Type: String
Length Constraints: Maximum length of 64.
Pattern: `[a-z0-9][\._a-z0-9]{1,61}[a-z0-9]`
Required: Yes

**OutputEncryptionKeyId (p. 292)**

The Amazon Resource Name (ARN) of the AWS Key Management Service (KMS) key used to encrypt the output of the transcription job. The user calling the `StartMedicalTranscriptionJob` (p. 292) operation must have permission to use the specified KMS key.
You 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 in the current account or another account:
  "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"

If you don't specify an encryption key, the output of the medical transcription job is encrypted with the default Amazon S3 key (SSE-S3).

If you specify a KMS key to encrypt your output, you must also specify an output location in the `OutputBucketName` parameter.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2048.

Pattern: `^[A-Za-z0-9\[A-Za-z0-9:_/+=,@.-]{0,2048}$`

Required: No

OutputKey (p. 292)

You can specify a location in an Amazon S3 bucket to store the output of your medical transcription job.

If you don't specify an output key, Amazon Transcribe Medical stores the output of your transcription job in the Amazon S3 bucket you specified. By default, the object key is "your-transcription-job-name.json".

You can use output keys to specify the Amazon S3 prefix and file name of the transcription output. For example, specifying the Amazon S3 prefix, "folder1/folder2/", as an output key would lead to the output being stored as "folder1/folder2/your-transcription-job-name.json". If you specify "my-other-job-name.json" as the output key, the object key is changed to "my-other-job-name.json". You can use an output key to change both the prefix and the file name, for example "folder/my-other-job-name.json".

If you specify an output key, you must also specify an S3 bucket in the `OutputBucketName` parameter.

Type: String


Pattern: `[a-zA-Z0-9_\-\.\!*\(\)]\{1,1024}$`

Required: No

Settings (p. 292)

Optional settings for the medical transcription job.

Type: `MedicalTranscriptionSetting (p. 338)` object

Required: No

Specialty (p. 292)

The medical specialty of any clinician speaking in the input media.

294
Type: String

Valid Values: PRIMARYCARE

Required: Yes

**Type (p. 292)**

The type of speech in the input audio. CONVERSATION refers to conversations between two or more speakers, e.g., a conversations between doctors and patients. DICTATION refers to single-speaker dictated speech, e.g., for clinical notes.

Type: String

Valid Values: CONVERSATION | DICTATION

Required: Yes

**Response Syntax**

```json
{
    "MedicalTranscriptionJob": {
        "CompletionTime": number,
        "CreationTime": number,
        "FailureReason": "string",
        "LanguageCode": "string",
        "Media": {
            "MediaFileUri": "string"
        },
        "MediaFormat": "string",
        "MediaSampleRateHertz": number,
        "MedicalTranscriptionJobName": "string",
        "Settings": {
            "ChannelIdentification": boolean,
            "MaxAlternatives": number,
            "MaxSpeakerLabels": number,
            "ShowAlternatives": boolean,
            "ShowSpeakerLabels": boolean,
            "VocabularyName": "string"
        },
        "Specialty": "string",
        "StartTime": number,
        "Transcript": {
            "TranscriptFileUri": "string"
        },
        "TranscriptionJobStatus": "string",
        "Type": "string"
    }
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**MedicalTranscriptionJob (p. 295)**

A batch job submitted to transcribe medical speech to text.

Type: MedicalTranscriptionJob (p. 333) object
Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400

ConflictException

There is already a resource with that name.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
StartTranscriptionJob

Service: Amazon Transcribe Service

Starts an asynchronous job to transcribe speech to text.

Request Syntax

```
{
    "ContentRedaction": {
        "RedactionOutput": "string",
        "RedactionType": "string"
    },
    "IdentifyLanguage": boolean,
    "JobExecutionSettings": {
        "AllowDeferredExecution": boolean,
        "DataAccessRoleArn": "string"
    },
    "LanguageCode": "string",
    "LanguageOptions": [ "string" ],
    "Media": {
        "MediaFileUri": "string"
    },
    "MediaFormat": "string",
    "MediaSampleRateHertz": number,
    "ModelSettings": {
        "LanguageModelName": "string"
    },
    "OutputBucketName": "string",
    "OutputEncryptionKMSKeyId": "string",
    "OutputKey": "string",
    "Settings": {
        "ChannelIdentification": boolean,
        "MaxAlternatives": number,
        "MaxSpeakerLabels": number,
        "ShowAlternatives": boolean,
        "ShowSpeakerLabels": boolean,
        "VocabularyFilterMethod": "string",
        "VocabularyFilterName": "string",
        "VocabularyName": "string"
    },
    "TranscriptionJobName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

ContentRedaction (p. 297)

An object that contains the request parameters for content redaction.

Type: ContentRedaction (p. 326) object

Required: No

IdentifyLanguage (p. 297)

Set this field to true to enable automatic language identification. Automatic language identification is disabled by default. You receive a BadRequestException error if you enter a value for a LanguageCode.
Type: Boolean
Required: No

**JobExecutionSettings (p. 297)**

Provides information about how a transcription job is executed. Use this field to indicate that the job can be queued for deferred execution if the concurrency limit is reached and there are no slots available to immediately run the job.

Type: JobExecutionSettings (p. 328) object
Required: No

**LanguageCode (p. 297)**

The language code for the language used in the input media file.

Type: String


Required: No

**LanguageOptions (p. 297)**

An object containing a list of languages that might be present in your collection of audio files. Automatic language identification chooses a language that best matches the source audio from that list.

Type: Array of strings

Array Members: Minimum number of 2 items.


Required: No

**Media (p. 297)**

An object that describes the input media for a transcription job.

Type: Media (p. 331) object
Required: Yes

**MediaFormat (p. 297)**

The format of the input media file.

Type: String

Valid Values: mp3 | mp4 | wav | flac | ogg | amr | webm

Required: No

**MediaSampleRateHertz (p. 297)**

The sample rate, in Hertz, of the audio track in the input media file.
If you do not specify the media sample rate, Amazon Transcribe determines the sample rate. If you specify the sample rate, it must match the sample rate detected by Amazon Transcribe. In most cases, you should leave the `MediaSampleRateHertz` field blank and let Amazon Transcribe determine the sample rate.

Type: Integer  
Required: No

**ModelSettings (p. 297)**

Choose the custom language model you use for your transcription job in this parameter.

Type: `ModelSettings (p. 340)` object  
Required: No

**OutputBucketName (p. 297)**

The location where the transcription is stored.

If you set the `OutputBucketName`, Amazon Transcribe puts the transcript in the specified S3 bucket. When you call the `GetTranscriptionJob (p. 265)` operation, the operation returns this location in the `TranscriptFileUri` field. If you enable content redaction, the redacted transcript appears in `RedactedTranscriptFileUri`. If you enable content redaction and choose to output an unredacted transcript, that transcript's location still appears in the `TranscriptFileUri`. The S3 bucket must have permissions that allow Amazon Transcribe to put files in the bucket. For more information, see Permissions Required for IAM User Roles.

You can specify an AWS Key Management Service (KMS) key to encrypt the output of your transcription using the `OutputEncryptionKMSKeyId` parameter. If you don't specify a KMS key, Amazon Transcribe uses the default Amazon S3 key for server-side encryption of transcripts that are placed in your S3 bucket.

If you don't set the `OutputBucketName`, Amazon Transcribe generates a pre-signed URL, a shareable URL that provides secure access to your transcription, and returns it in the `TranscriptFileUri` field. Use this URL to download the transcription.

Type: String  
Length Constraints: Maximum length of 64.  
Pattern: `[a-z0-9][\./\-a-z0-9]{1,61}[a-z0-9]`  
Required: No

**OutputEncryptionKMSKeyId (p. 297)**

The Amazon Resource Name (ARN) of the AWS Key Management Service (KMS) key used to encrypt the output of the transcription job. The user calling the `StartTranscriptionJob` operation must have permission to use the specified 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"

If you don't specify an encryption key, the output of the transcription job is encrypted with the default Amazon S3 key (SSE-S3).

If you specify a KMS key to encrypt your output, you must also specify an output location in the OutputBucketName parameter.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 2048.
Pattern: ^[A-Za-z0-9]([A-Za-z0-9:_/+@.]+[^0-9a-zA-Z._-])]{0,2048}$
Required: No

OutputKey (p. 297)

You can specify a location in an Amazon S3 bucket to store the output of your transcription job.

If you don't specify an output key, Amazon Transcribe stores the output of your transcription job in the Amazon S3 bucket you specified. By default, the object key is "your-transcription-job-name.json".

You can use output keys to specify the Amazon S3 prefix and file name of the transcription output. For example, specifying the Amazon S3 prefix, "folder1/folder2/", as an output key would lead to the output being stored as "folder1/folder2/your-transcription-job-name.json". If you specify "my-other-job-name.json" as the output key, the object key is changed to "my-other-job-name.json". You can use an output key to change both the prefix and the file name, for example "folder/my-other-job-name.json".

If you specify an output key, you must also specify an S3 bucket in the OutputBucketName parameter.

Type: String
Pattern: [a-zA-Z0-9-.]*[a-zA-Z0-9_.-]*/1,1024}$
Required: No

Settings (p. 297)

A Settings object that provides optional settings for a transcription job.

Type: Settings (p. 341) object
Required: No

TranscriptionJobName (p. 297)

The name of the job. You can't use the strings "." or ".." by themselves as the job name. The name must also be unique within an AWS account. If you try to create a transcription job with the same name as a previous transcription job, you get a ConflictException error.

Type: String
Pattern: ^[0-9a-zA-Z_-.]+$
Required: Yes
Response Syntax

```json
{
  "TranscriptionJob": {
    "CompletionTime": number,
    "ContentRedaction": {
      "RedactionOutput": "string",
      "RedactionType": "string"
    },
    "CreationTime": number,
    "FailureReason": "string",
    "IdentifiedLanguageScore": number,
    "IdentifyLanguage": boolean,
    "JobExecutionSettings": {
      "AllowDeferredExecution": boolean,
      "DataAccessRoleArn": "string"
    },
    "LanguageCode": "string",
    "LanguageOptions": [ "string" ],
    "Media": {
      "MediaFileUri": "string"
    },
    "MediaFormat": "string",
    "MediaSampleRateHertz": number,
    "ModelSettings": {
      "LanguageModelName": "string"
    },
    "Settings": {
      "ChannelIdentification": boolean,
      "MaxAlternatives": number,
      "MaxSpeakerLabels": number,
      "ShowAlternatives": boolean,
      "ShowSpeakerLabels": boolean,
      "VocabularyFilterMethod": "string",
      "VocabularyFilterName": "string",
      "VocabularyName": "string"
    },
    "StartTime": number,
    "Transcript": {
      "RedactedTranscriptFileUri": "string",
      "TranscriptFileUri": "string"
    },
    "TranscriptionJobName": "string",
    "TranscriptionJobStatus": "string"
  }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**TranscriptionJob (p. 301)**

An object containing details of the asynchronous transcription job.

Type: TranscriptionJob (p. 344) object

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).
BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying
to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the
exception Message field for more information.

HTTP Status Code: 400

ConflictException

There is already a resource with that name.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your
request, or use a smaller file and resend the request.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
UpdateMedicalVocabulary
Service: Amazon Transcribe Service

Updates a vocabulary with new values that you provide in a different text file from the one you used to create the vocabulary. The UpdateMedicalVocabulary operation overwrites all of the existing information with the values that you provide in the request.

Request Syntax

```
{
   "LanguageCode": "string",
   "VocabularyFileUri": "string",
   "VocabularyName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

**LanguageCode (p. 303)**

The language code of the language used for the entries in the updated vocabulary. US English (en-US) is the only valid language code in Amazon Transcribe Medical.

Type: String


Required: Yes

**VocabularyFileUri (p. 303)**

The location in Amazon S3 of the text file that contains the you use for your custom vocabulary. The URI must be in the same AWS Region as the resource that you are calling. The following is the format for a URI:

https://s3.<aws-region>.amazonaws.com/<bucket-name>/<keyprefix>/<objectkey>

For example:

https://s3.us-east-1.amazonaws.com/AWSDOC-EXAMPLE-BUCKET/vocab.txt

For more information about Amazon S3 object names, see Object Keys in the Amazon S3 Developer Guide.

For more information about custom vocabularies in Amazon Transcribe Medical, see Medical Custom Vocabularies.

Type: String


Pattern: (s3://|http(s*)://)+
VocabularyName (p. 303)

The name of the vocabulary to update. The name is case sensitive. If you try to update a vocabulary with the same name as a vocabulary you've already made, you get a ConflictException error.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

Response Syntax

```
{
  "LanguageCode": "string",
  "LastModifiedTime": number,
  "VocabularyName": "string",
  "VocabularyState": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

LanguageCode (p. 304)

The language code for the language of the text file used to update the custom vocabulary. US English (en-US) is the only language supported in Amazon Transcribe Medical.

Type: String

Valid Values:
- af-ZA
- ar-AE
- ar-SA
- cy-GB
- da-DK
- de-CH
- de-DE
- en-AB
- en-AU
- en-GB
- en-IE
- en-IN
- en-US
- en-WL
- es-ES
- es-US
- fa-IR
- fr-CA
- fr-FR
- ga-IE
- gd-GB
- he-IL
- hi-IN
- id-ID
- it-IT
- ja-JP
- ko-KR
- ms-MY
- nl-NL
- pt-BR
- pt-PT
- ru-RU
- ta-IN
- te-IN
- tr-TR
- zh-CN

LastModifiedTime (p. 304)

The date and time that the vocabulary was updated.

Type: Timestamp

VocabularyName (p. 304)

The name of the updated vocabulary.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

VocabularyState (p. 304)

The processing state of the update to the vocabulary. When the VocabularyState field is READY, the vocabulary is ready to be used in a StartMedicalTranscriptionJob request.
Type: String

Valid Values: PENDING | READY | FAILED

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException

Your request didn't pass one or more validation tests. For example, if the entity that you're trying
to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the
exception Message field for more information.

HTTP Status Code: 400

ConflictException

There is already a resource with that name.

HTTP Status Code: 400

InternalFailureException

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException

Either you have sent too many requests or your input file is too long. Wait before you resend your
request, or use a smaller file and resend the request.

HTTP Status Code: 400

NotFoundException

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
UpdateVocabulary
Service: Amazon Transcribe Service

Updates an existing vocabulary with new values. The UpdateVocabulary operation overwrites all of the existing information with the values that you provide in the request.

Request Syntax

```
{
  "LanguageCode": "string",
  "Phrases": [ "string" ],
  "VocabularyFileUri": "string",
  "VocabularyName": "string"
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

LanguageCode (p. 306)

The language code of the vocabulary entries.

Type: String


Required: Yes

Phrases (p. 306)

An array of strings containing the vocabulary entries.

Type: Array of strings

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: .+

Required: No

VocabularyFileUri (p. 306)

The S3 location of the text file that contains the definition of the custom vocabulary. The URI must be in the same region as the API endpoint that you are calling. The general form is

```
```

For example:

```
https://s3.us-east-1.amazonaws.com/AWSDOC-EXAMPLE-BUCKET/vocab.txt
```

For more information about S3 object names, see Object Keys in the Amazon S3 Developer Guide.
For more information about custom vocabularies, see Custom Vocabularies.

Type: String


Pattern: (s3://|http(s*)://).+

Required: No

**VocabularyName (p. 306)**

The name of the vocabulary to update. The name is case sensitive. If you try to update a vocabulary with the same name as a previous vocabulary you will receive a ConflictException error.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: Yes

**Response Syntax**

```json
{
    "LanguageCode": "string",
    "LastModifiedTime": number,
    "VocabularyName": "string",
    "VocabularyState": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**LanguageCode (p. 307)**

The language code of the vocabulary entries.

Type: String


**LastModifiedTime (p. 307)**

The date and time that the vocabulary was updated.

Type: Timestamp

**VocabularyName (p. 307)**

The name of the vocabulary that was updated.

Type: String

Pattern: ^[0-9a-zA-Z._-]+$  

VocabularyState (p. 307)  

The processing state of the vocabulary. When the VocabularyState field contains READY the vocabulary is ready to be used in a StartTranscriptionJob request.

Type: String  

Valid Values: PENDING | READY | FAILED

Errors

For information about the errors that are common to all actions, see Common Errors (p. 373).

BadRequestException  

Your request didn’t pass one or more validation tests. For example, if the entity that you’re trying to delete doesn’t exist or if it is in a non-terminal state (for example, it’s “in progress”). See the exception Message field for more information.

HTTP Status Code: 400

ConflictException  

There is already a resource with that name.

HTTP Status Code: 400

InternalServerError  

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

LimitExceededException  

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

NotFoundException  

We can’t find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface  
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
• AWS SDK for Ruby V3
UpdateVocabularyFilter
Service: Amazon Transcribe Service

Updates a vocabulary filter with a new list of filtered words.

Request Syntax

```
{
   "VocabularyFilterFileUri": "string",
   "VocabularyFilterName": "string",
   "Words": [ "string" ]
}
```

Request Parameters

For information about the parameters that are common to all actions, see Common Parameters (p. 374).

The request accepts the following data in JSON format.

VocabularyFilterFileUri (p. 310)

The Amazon S3 location of a text file used as input to create the vocabulary filter. Only use characters from the character set defined for custom vocabularies. For a list of character sets, see Character Sets for Custom Vocabularies.

The specified file must be less than 50 KB of UTF-8 characters.

If you provide the location of a list of words in the VocabularyFilterFileUri parameter, you can't use the Words parameter.

Type: String


Pattern: (s3://|http(s*)://).+

Required: No

VocabularyFilterName (p. 310)

The name of the vocabulary filter to update. If you try to update a vocabulary filter with the same name as another vocabulary filter, you get a ConflictException error.

Type: String


Pattern: ^[0-9a-zA-Z.\-_]+$

Required: Yes

Words (p. 310)

The words to use in the vocabulary filter. Only use characters from the character set defined for custom vocabularies. For a list of character sets, see Character Sets for Custom Vocabularies.

If you provide a list of words in the Words parameter, you can't use the VocabularyFilterFileUri parameter.

Type: Array of strings
Array Members: Minimum number of 1 item.
Length Constraints: Minimum length of 1. Maximum length of 256.
Required: No

**Response Syntax**

```json
{
    "LanguageCode": "string",
    "LastModifiedTime": number,
    "VocabularyFilterName": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**LanguageCode (p. 311)**

The language code of the words in the vocabulary filter.
Type: String

**LastModifiedTime (p. 311)**

The date and time that the vocabulary filter was updated.
Type: Timestamp

**VocabularyFilterName (p. 311)**

The name of the updated vocabulary filter.
Type: String
Pattern: ^[0-9a-zA-Z._-]+$

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

Your request didn't pass one or more validation tests. For example, if the entity that you're trying to delete doesn't exist or if it is in a non-terminal state (for example, it's "in progress"). See the exception Message field for more information.

HTTP Status Code: 400
**InternalFailureException**

There was an internal error. Check the error message and try your request again.

HTTP Status Code: 500

**LimitExceeded Exception**

Either you have sent too many requests or your input file is too long. Wait before you resend your request, or use a smaller file and resend the request.

HTTP Status Code: 400

**NotFoundException**

We can't find the requested resource. Check the name and try your request again.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

**Amazon Transcribe Streaming Service**

The following actions are supported by Amazon Transcribe Streaming Service:

- StartMedicalStreamTranscription (p. 313)
- StartStreamTranscription (p. 319)
StartMedicalStreamTranscription
Service: Amazon Transcribe Streaming Service

Starts a bidirectional HTTP/2 stream where audio is streamed to Amazon Transcribe Medical and the transcription results are streamed to your application.

Request Syntax

```
POST /medical-stream-transcription HTTP/2
x-amzn-transcribe-language-code: LanguageCode
x-amzn-transcribe-sample-rate: MediaSampleRateHertz
x-amzn-transcribe-media-encoding: MediaEncoding
x-amzn-transcribe-vocabulary-name: VocabularyName
x-amzn-transcribe-specialty: Specialty
x-amzn-transcribe-type: Type
x-amzn-transcribe-show-speaker-label: ShowSpeakerLabel
x-amzn-transcribe-session-id: SessionId
x-amzn-transcribe-enable-channel-identification: EnableChannelIdentification
x-amzn-transcribe-number-of-channels: NumberOfChannels
Content-type: application/json

{
  "AudioStream": {
    "AudioEvent": {
      "AudioChunk": blob
    }
  }
}
```

URI Request Parameters

The request uses the following URI parameters.

**EnableChannelIdentification (p. 313)**

When `true`, instructs Amazon Transcribe Medical to process each audio channel separately and then merge the transcription output of each channel into a single transcription.

Amazon Transcribe Medical also produces a transcription of each item. An item includes the start time, end time, and any alternative transcriptions.

You can't set both `ShowSpeakerLabel` and `EnableChannelIdentification` in the same request. If you set both, your request returns a `BadRequestException`.

**LanguageCode (p. 313)**

Indicates the source language used in the input audio stream. For Amazon Transcribe Medical, this is US English (en-US).


Required: Yes

**MediaEncoding (p. 313)**

The encoding used for the input audio.

Valid Values: `pcm | ogg-opus | flac`

Required: Yes
MediaSampleRateHertz (p. 313)

The sample rate of the input audio in Hertz. Sample rates of 16000 Hz or higher are accepted.


Required: Yes

NumberOfChannels (p. 313)

The number of channels that are in your audio stream.

Valid Range: Minimum value of 2.

SessionId (p. 313)

Optional. An identifier for the transcription session. If you don't provide a session ID, Amazon Transcribe generates one for you and returns it in the response.

Length Constraints: Fixed length of 36.

Pattern: [a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}

ShowSpeakerLabel (p. 313)

When true, enables speaker identification in your real-time stream.

Specialty (p. 313)

The medical specialty of the clinician or provider.

Valid Values: PRIMARYCARE | CARDIOLOGY | NEUROLOGY | ONCOLOGY | RADIOLOGY | UROLOGY

Required: Yes

Type (p. 313)

The type of input audio. Choose DICTATION for a provider dictating patient notes. Choose CONVERSATION for a dialogue between a patient and one or more medical professionals.

Valid Values: CONVERSATION | DICTATION

Required: Yes

VocabularyName (p. 313)

The name of the medical custom vocabulary to use when processing the real-time stream.


Pattern: ^[0-9a-zA-Z_.-]+$
Response Syntax

HTTP/2 200
x-amzn-request-id: RequestId
x-amzn-transcribe-language-code: LanguageCode
x-amzn-transcribe-sample-rate: MediaSampleRateHertz
x-amzn-transcribe-media-encoding: MediaEncoding
x-amzn-transcribe-vocabulary-name: VocabularyName
x-amzn-transcribe-specialty: Specialty
x-amzn-transcribe-type: Type
x-amzn-transcribe-show-speaker-label: ShowSpeakerLabel
x-amzn-transcribe-session-id: SessionId
x-amzn-transcribe-enable-channel-identification: EnableChannelIdentification
x-amzn-transcribe-number-of-channels: NumberOfChannels
Content-type: application/json

{
    "TranscriptResultStream": {
        "BadRequestException": {
        },
        "ConflictException": {
        },
        "InternalFailureException": {
        },
        "LimitExceededException": {
        },
        "ServiceUnavailableException": {
        },
        "TranscriptEvent": {
            "Transcript": {
                "Results": [
                    {
                        "Alternatives": [
                            {
                                "Items": [
                                    {
                                        "Confidence": number,
                                        "Content": "string",
                                        "EndTime": number,
                                        "Speaker": "string",
                                        "StartTime": number,
                                        "Type": "string"
                                    }
                                ],
                                "Transcript": "string"
                            }
                        ],
                        "ChannelId": "string",
                        "EndTime": number,
                        "IsPartial": boolean,
                        "ResultId": "string",
                        "StartTime": number
                    }
                }
            }
        }
    }
}
EnableChannelIdentification (p. 315)

Shows whether channel identification has been enabled in the stream.

LanguageCode (p. 315)

The language code for the response transcript. For Amazon Transcribe Medical, this is US English (en-US).


MediaEncoding (p. 315)

The encoding used for the input audio stream.

Valid Values: pcm | ogg-opus | flac

MediaSampleRateHertz (p. 315)

The sample rate of the input audio in Hertz. Valid value: 16000 Hz.


NumberOfChannels (p. 315)

The number of channels identified in the stream.

Valid Range: Minimum value of 2.

RequestId (p. 315)

An identifier for the streaming transcription.

SessionId (p. 315)

Optional. An identifier for the transcription session. If you don't provide a session ID, Amazon Transcribe generates one for you and returns it in the response.

Length Constraints: Fixed length of 36.

Pattern: ^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$

ShowSpeakerLabel (p. 315)

Shows whether speaker identification was enabled in the stream.

Specialty (p. 315)

The specialty in the medical domain.

Valid Values: PRIMARYCARE | CARDIOLOGY | NEUROLOGY | ONCOLOGY | RADIOLGY | UROLOGY

Type (p. 315)

The type of audio that was transcribed.

Valid Values: CONVERSATION | DICTATION

VocabularyName (p. 315)

The name of the vocabulary used when processing the stream.


Pattern: ^[0-9a-zA-Z._-]+$
The following data is returned in JSON format by the service.

**TranscriptResultStream (p. 315)**

Represents the stream of transcription events from Amazon Transcribe Medical to your application.

Type: `MedicalTranscriptResultStream (p. 366)` object

**Errors**

For information about the errors that are common to all actions, see [Common Errors (p. 373)](#).

**BadRequestException**

One or more arguments to the `StartStreamTranscription` or `StartMedicalStreamTranscription` operation was invalid. For example, `MediaEncoding` was not set to a valid encoding, or `LanguageCode` was not set to a valid code. Check the parameters and try your request again.

HTTP Status Code: 400

**ConflictException**

A new stream started with the same session ID. The current stream has been terminated.

HTTP Status Code: 409

**InternalFailureException**

A problem occurred while processing the audio. Amazon Transcribe or Amazon Transcribe Medical terminated processing. Try your request again.

HTTP Status Code: 500

**LimitExceedededException**

You have exceeded the maximum number of concurrent transcription streams, are starting transcription streams too quickly, or the maximum audio length of 4 hours. Wait until a stream has finished processing, or break your audio stream into smaller chunks and try your request again.

HTTP Status Code: 429

**ServiceUnavailableException**

Service is currently unavailable. Try your request later.

HTTP Status Code: 503

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- AWS SDK for Python
- AWS SDK for Ruby V3
StartStreamTranscription

Service: Amazon Transcribe Streaming Service

Starts a bidirectional HTTP2 stream where audio is streamed to Amazon Transcribe and the transcription results are streamed to your application.

The following are encoded as HTTP2 headers:

- x-amzn-transcribe-language-code
- x-amzn-transcribe-media-encoding
- x-amzn-transcribe-sample-rate
- x-amzn-transcribe-session-id

Request Syntax

```plaintext
POST /stream-transcription HTTP/2
x-amzn-transcribe-language-code: LanguageCode
x-amzn-transcribe-sample-rate: MediaSampleRateHertz
x-amzn-transcribe-media-encoding: MediaEncoding
x-amzn-transcribe-vocabulary-name: VocabularyName
x-amzn-transcribe-session-id: SessionId
x-amzn-transcribe-vocabulary-filter-name: VocabularyFilterName
x-amzn-transcribe-vocabulary-filter-method: VocabularyFilterMethod
x-amzn-transcribe-show-speaker-label: ShowSpeakerLabel
x-amzn-transcribe-enable-channel-identification: EnableChannelIdentification
x-amzn-transcribe-number-of-channels: NumberOfChannels
Content-type: application/json

{
    "AudioStream": {
        "AudioEvent": {
            "AudioChunk": blob
        }
    }
}
```

URI Request Parameters

The request uses the following URI parameters.

**EnableChannelIdentification (p. 319)**

When true, instructs Amazon Transcribe to process each audio channel separately and then merge the transcription output of each channel into a single transcription.

Amazon Transcribe also produces a transcription of each item. An item includes the start time, end time, and any alternative transcriptions.

You can’t set both ShowSpeakerLabel and EnableChannelIdentification in the same request. If you set both, your request returns a BadRequestException.

**LanguageCode (p. 319)**

Indicates the source language used in the input audio stream.


Required: Yes
**MediaEncoding (p. 319)**

The encoding used for the input audio.

Valid Values: `pcm` | `ogg-opus` | `flac`

Required: Yes

**MediaSampleRateHertz (p. 319)**

The sample rate, in Hertz, of the input audio. We suggest that you use 8000 Hz for low quality audio and 16000 Hz for high quality audio.


Required: Yes

**NumberOfChannels (p. 319)**

The number of channels that are in your audio stream.

Valid Range: Minimum value of 2.

**SessionId (p. 319)**

A identifier for the transcription session. Use this parameter when you want to retry a session. If you don't provide a session ID, Amazon Transcribe will generate one for you and return it in the response.

Length Constraints: Fixed length of 36.

Pattern: `^[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}$`

**ShowSpeakerLabel (p. 319)**

When `true`, enables speaker identification in your real-time stream.

**VocabularyFilterMethod (p. 319)**

The manner in which you use your vocabulary filter to filter words in your transcript. `Remove` removes filtered words from your transcription results. `Mask` masks those words with a `***` in your transcription results. `Tag` keeps the filtered words in your transcription results and tags them. The tag appears as `VocabularyFilterMatch` equal to `True`.

Valid Values: `remove` | `mask` | `tag`

**VocabularyFilterName (p. 319)**

The name of the vocabulary filter you've created that is unique to your AWS account. Provide the name in this field to successfully use it in a stream.


Pattern: `^[0-9a-zA-Z._-]+$`

**VocabularyName (p. 319)**

The name of the vocabulary to use when processing the transcription job.


Pattern: `^[0-9a-zA-Z._-]+$`

**Request Body**

The request accepts the following data in JSON format.
AudioStream (p. 319)

PCM-encoded stream of audio blobs. The audio stream is encoded as an HTTP2 data frame.

Type: AudioStream (p. 356) object

Required: Yes

Response Syntax

HTTP/2 200
x-amzn-request-id: RequestId
x-amzn-transcribe-language-code: LanguageCode
x-amzn-transcribe-sample-rate: MediaSampleRateHertz
x-amzn-transcribe-media-encoding: MediaEncoding
x-amzn-transcribe-vocabulary-name: VocabularyName
x-amzn-transcribe-session-id: SessionId
x-amzn-transcribe-vocabulary-filter-name: VocabularyFilterName
x-amzn-transcribe-vocabulary-filter-method: VocabularyFilterMethod
x-amzn-transcribe-show-speaker-label: ShowSpeakerLabel
x-amzn-transcribe-enable-channel-identification: EnableChannelIdentification
x-amzn-transcribe-number-of-channels: NumberOfChannels
Content-type: application/json

```json
{
  "TranscriptResultStream": {
    "BadRequestException": {},
    "ConflictException": {},
    "InternalFailureException": {},
    "LimitExceeded.Exception": {},
    "ServiceUnavailableException": {},
    "TranscriptEvent": {
      "Transcript": {
        "Results": [
          {
            "Alternatives": [
              {
                "Items": [
                  {
                    "Content": "string",
                    "EndTime": number,
                    "Speaker": "string",
                    "StartTime": number,
                    "Type": "string",
                    "VocabularyFilterMatch": boolean
                  }
                ],
                "Transcript": "string"
              }
            ],
            "ChannelId": "string",
            "EndTime": number,
            "IsPartial": boolean,
            "ResultId": "string",
            "StartTime": number
          }
        ]
      }
    }
  }
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The response returns the following HTTP headers.

EnableChannelIdentification (p. 321)

Shows whether channel identification has been enabled in the stream.

LanguageCode (p. 321)

The language code for the input audio stream.


MediaEncoding (p. 321)

The encoding used for the input audio stream.

Valid Values: pcm | ogg-opus | flac

MediaSampleRateHertz (p. 321)

The sample rate for the input audio stream. Use 8000 Hz for low quality audio and 16000 Hz for high quality audio.


NumberOfChannels (p. 321)

The number of channels identified in the stream.

Valid Range: Minimum value of 2.

RequestId (p. 321)

An identifier for the streaming transcription.

SessionId (p. 321)

An identifier for a specific transcription session.

Length Constraints: Fixed length of 36.

Pattern: [a-zA-F0-9]{8}-[a-zA-F0-9]{4}-[a-zA-F0-9]{4}-[a-zA-F0-9]{4}-[a-zA-F0-9]{12}

ShowSpeakerLabel (p. 321)

Shows whether speaker identification was enabled in the stream.

VocabularyFilterMethod (p. 321)

The vocabulary filtering method used in the real-time stream.

Valid Values: remove | mask | tag

VocabularyFilterName (p. 321)

The name of the vocabulary filter used in your real-time stream.

VocabularyName (p. 321)

The name of the vocabulary used when processing the stream.

**Length Constraints:** Minimum length of 1. Maximum length of 200.

**Pattern:** `^[0-9a-zA-Z._-]+$`

The following data is returned in JSON format by the service.

**TranscriptResultStream (p. 321)**

Represents the stream of transcription events from Amazon Transcribe to your application.

Type: TranscriptResultStream (p. 372) object

**Errors**

For information about the errors that are common to all actions, see Common Errors (p. 373).

**BadRequestException**

One or more arguments to the `StartStreamTranscription` or `StartMedicalStreamTranscription` operation was invalid. For example, `MediaEncoding` was not set to a valid encoding, or `LanguageCode` was not set to a valid code. Check the parameters and try your request again.

HTTP Status Code: 400

**ConflictException**

A new stream started with the same session ID. The current stream has been terminated.

HTTP Status Code: 409

**InternalFailureException**

A problem occurred while processing the audio. Amazon Transcribe or Amazon Transcribe Medical terminated processing. Try your request again.

HTTP Status Code: 500

**LimitExceededException**

You have exceeded the maximum number of concurrent transcription streams, are starting transcription streams too quickly, or the maximum audio length of 4 hours. Wait until a stream has finished processing, or break your audio stream into smaller chunks and try your request again.

HTTP Status Code: 429

**ServiceUnavailableException**

Service is currently unavailable. Try your request later.

HTTP Status Code: 503

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:
Data Types

The following data types are supported by Amazon Transcribe Service:

- ContentRedaction (p. 326)
- InputDataConfig (p. 327)
- JobExecutionSettings (p. 328)
- LanguageModel (p. 329)
- Media (p. 331)
- MedicalTranscript (p. 332)
- MedicalTranscriptionJob (p. 333)
- MedicalTranscriptionJobSummary (p. 336)
- MedicalTranscriptionSetting (p. 338)
- ModelSettings (p. 340)
- Settings (p. 341)
- Transcript (p. 343)
- TranscriptionJob (p. 344)
- TranscriptionJobSummary (p. 348)
- VocabularyFilterInfo (p. 351)
- VocabularyInfo (p. 352)

The following data types are supported by Amazon Transcribe Streaming Service:

- Alternative (p. 354)
- AudioEvent (p. 355)
- AudioStream (p. 356)
- Item (p. 357)
- MedicalAlternative (p. 359)
- MedicalItem (p. 360)
- MedicalResult (p. 362)
- MedicalTranscript (p. 364)
- MedicalTranscriptEvent (p. 365)
- MedicalTranscriptResultStream (p. 366)
- Result (p. 368)
- Transcript (p. 370)
- TranscriptEvent (p. 371)
- TranscriptResultStream (p. 372)
• ContentRedaction (p. 326)
• InputDataConfig (p. 327)
• JobExecutionSettings (p. 328)
• LanguageModel (p. 329)
• Media (p. 331)
• MedicalTranscript (p. 332)
• MedicalTranscriptionJob (p. 333)
• MedicalTranscriptionJobSummary (p. 336)
• MedicalTranscriptionSetting (p. 338)
• ModelSettings (p. 340)
• Settings (p. 341)
• Transcript (p. 343)
• TranscriptionJob (p. 344)
• TranscriptionJobSummary (p. 348)
• VocabularyFilterInfo (p. 351)
• VocabularyInfo (p. 352)
ContentRedaction
Service: Amazon Transcribe Service

Settings for content redaction within a transcription job.

Contents

RedactionOutput
The output transcript file stored in either the default S3 bucket or in a bucket you specify.
When you choose redacted Amazon Transcribe outputs only the redacted transcript.
When you choose redacted_and_unredacted Amazon Transcribe outputs both the redacted and unredacted transcripts.
Type: String
Valid Values: redacted | redacted_and_unredacted
Required: Yes

RedactionType
Request parameter that defines the entities to be redacted. The only accepted value is PII.
Type: String
Valid Values: PII
Required: Yes

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
**InputDataConfig**

Service: Amazon Transcribe Service

The object that contains the Amazon S3 object location and access role required to train and tune your custom language model.

**Contents**

**DataAccessRoleArn**

The Amazon Resource Name (ARN) that uniquely identifies the permissions you've given Amazon Transcribe to access your Amazon S3 buckets containing your media files or text data.

Type: String


Pattern: `^arn:(aws|aws-cn|aws-us-gov|aws-iso-{0,1}[a-z]{0,1}):iam::[0-9]{0,63}:role/[A-Za-z0-9:_=,+@.-]{0,1024}$`

Required: Yes

**S3Uri**

The Amazon S3 prefix you specify to access the plain text files that you use to train your custom language model.

Type: String


Pattern: `(s3://|http(s*)://).+`

Required: Yes

**TuningDataS3Uri**

The Amazon S3 prefix you specify to access the plain text files that you use to tune your custom language model.

Type: String


Pattern: `(s3://|http(s*)://).+`

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
JobExecutionSettings
Service: Amazon Transcribe Service

Provides information about when a transcription job should be executed.

Contents

AllowDeferredExecution

Indicates whether a job should be queued by Amazon Transcribe when the concurrent execution limit is exceeded. When the AllowDeferredExecution field is true, jobs are queued and executed when the number of executing jobs falls below the concurrent execution limit. If the field is false, Amazon Transcribe returns a LimitExceededException exception.

If you specify the AllowDeferredExecution field, you must specify the DataAccessRoleArn field.

Type: Boolean
Required: No

DataAccessRoleArn

The Amazon Resource Name (ARN) of a role that has access to the S3 bucket that contains the input files. Amazon Transcribe assumes this role to read queued media files. If you have specified an output S3 bucket for the transcription results, this role should have access to the output bucket as well.

If you specify the AllowDeferredExecution field, you must specify the DataAccessRoleArn field.

Type: String

Pattern: ^arn:(aws|aws-cn|aws-us-gov|aws-iso-{0,1}[a-z]{0,1}):iam::[0-9]{0,63}:role/[A-Za-z0-9-_+/?=,\@]{0,1024}$

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
**LanguageModel**
Service: Amazon Transcribe Service

The structure used to describe a custom language model.

**Contents**

**BaseModelName**

The Amazon Transcribe standard language model, or base model used to create the custom language model.

Type: String

Valid Values: NarrowBand | WideBand

Required: No

**CreateTime**

The time the custom language model was created.

Type: Timestamp

Required: No

**FailureReason**

The reason why the custom language model couldn't be created.

Type: String

Required: No

**InputDataConfig**

The data access role and Amazon S3 prefixes for the input files used to train the custom language model.

Type: InputDataConfig (p. 327) object

Required: No

**LanguageCode**

The language code you used to create your custom language model.

Type: String

Valid Values: en-US

Required: No

**LastModifiedTime**

The most recent time the custom language model was modified.

Type: Timestamp

Required: No

**ModelName**

The name of the custom language model.
Type: String
Pattern: \^[0-9a-zA-Z._-]+\$
Required: No

**ModelStatus**

The creation status of a custom language model. When the status is **COMPLETED** the model is ready for use.

Type: String
Valid Values: IN_PROGRESS | FAILED | COMPLETED
Required: No

**UpgradeAvailability**

Whether the base model used for the custom language model is up to date. If this field is true then you are running the most up-to-date version of the base model in your custom language model.

Type: Boolean
Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
Media
Service: Amazon Transcribe Service

Describes the input media file in a transcription request.

Contents

MediaFileUri

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. The general form is:

s3://<AWSDOC-EXAMPLE-BUCKET>/<keyprefix>/<objectkey>

For example:

s3://AWSDOC-EXAMPLE-BUCKET/example.mp4
s3://AWSDOC-EXAMPLE-BUCKET/mediadocs/example.mp4

For more information about S3 object names, see Object Keys in the Amazon S3 Developer Guide.

Type: String


Pattern: (s3://|http(s*)://).+

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
MedicalTranscript
Service: Amazon Transcribe Service

Identifies the location of a medical transcript.

Contents

TranscriptFileUri

The S3 object location of the medical transcript.

Use this URI to access the medical transcript. This URI points to the S3 bucket you created to store the medical transcript.

Type: String


Pattern: (s3://|http(s*)://).+

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
MedicalTranscriptionJob
Service: Amazon Transcribe Service

The data structure that contains the information for a medical transcription job.

Contents

CompletionTime
A timestamp that shows when the job was completed.
Type: Timestamp
Required: No

CreationTime
A timestamp that shows when the job was created.
Type: Timestamp
Required: No

FailureReason
If the TranscriptionJobStatus field is FAILED, this field contains information about why the job failed.

The FailureReason field contains one of the following values:
- Unsupported media format- The media format specified in the MediaFormat field of the request isn't valid. See the description of the MediaFormat field for a list of valid values.
- The media format provided does not match the detected media format- The media format of the audio file doesn't match the format specified in the MediaFormat field in the request. Check the media format of your media file and make sure the two values match.
- Invalid sample rate for audio file- The sample rate specified in the MediaSampleRateHertz of the request isn't valid. The sample rate must be between 8000 and 48000 Hertz.
- The sample rate provided does not match the detected sample rate- The sample rate in the audio file doesn't match the sample rate specified in the MediaSampleRateHertz field in the request. Check the sample rate of your media file and make sure that the two values match.
- Invalid file size: file size too large- The size of your audio file is larger than what Amazon Transcribe Medical can process. For more information, see Guidelines and Quotas in the Amazon Transcribe Medical Guide
- Invalid number of channels: number of channels too large- Your audio contains more channels than Amazon Transcribe Medical is configured to process. To request additional channels, see Amazon Transcribe Medical Endpoints and Quotas in the Amazon Web Services General Reference

Type: String
Required: No

LanguageCode
The language code for the language spoken in the source audio file. US English (en-US) is the only supported language for medical transcriptions. Any other value you enter for language code results in a BadRequestException error.

Type: String
Valid Values: af-ZA | ar-AE | ar-SA | cy-GB | da-DK | de-CH | de-DE | en-AB |
| fr-FR | ga-IE | gd-GB | he-IL | hi-IN | id-ID | it-IT | ja-JP | ko-KR |
| ms-MY | nl-NL | pt-BR | pt-PT | ru-RU | ta-IN | te-IN | tr-TR | zh-CN

Required: No

Media

Describes the input media file in a transcription request.

Type: Media (p. 331) object

Required: No

MediaFormat

The format of the input media file.

Type: String

Valid Values: mp3 | mp4 | wav | flac | ogg | amr | webm

Required: No

MediaSampleRateHertz

The sample rate, in Hertz, of the source audio containing medical information.

If you don't specify the sample rate, Amazon Transcribe Medical determines it for you. If you choose to specify the sample rate, it must match the rate detected by Amazon Transcribe Medical. In most cases, you should leave the MediaSampleRateHertz blank and let Amazon Transcribe Medical determine the sample rate.

Type: Integer


Required: No

MedicalTranscriptionJobName

The name for a given medical transcription job.

Type: String


Pattern: ^[0-9a-zA-Z-_]+$

Required: No

Settings

Object that contains MedicalTranscriptionSetting (p. 338) object.

Type: MedicalTranscriptionSetting (p. 338) object

Required: No

Specialty

The medical specialty of any clinicians providing a dictation or having a conversation. PRIMARYCARE is the only available setting for this object. This specialty enables you to generate transcriptions for the following medical fields:
• Family Medicine
  
  Type: String
  
  Valid Values: PRIMARYCARE
  
  Required: No

**StartTime**

A timestamp that shows when the job started processing.

Type: Timestamp

Required: No

**Transcript**

An object that contains the MedicalTranscript. The MedicalTranscript contains the TranscriptFileUri.

Type: MedicalTranscript (p. 332) object

Required: No

**TranscriptionJobStatus**

The completion status of a medical transcription job.

Type: String

Valid Values: QUEUED | IN_PROGRESS | FAILED | COMPLETED

Required: No

**Type**

The type of speech in the transcription job. CONVERSATION is generally used for patient-physician dialogues. DICTATION is the setting for physicians speaking their notes after seeing a patient. For more information, see How Amazon Transcribe Medical Works (p. 98)

Type: String

Valid Values: CONVERSATION | DICTATION

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
MedicalTranscriptionJobSummary
Service: Amazon Transcribe Service

Provides summary information about a transcription job.

Contents

CompletionTime
A timestamp that shows when the job was completed.
Type: Timestamp
Required: No

CreationTime
A timestamp that shows when the medical transcription job was created.
Type: Timestamp
Required: No

FailureReason
If the TranscriptionJobStatus field is FAILED, a description of the error.
Type: String
Required: No

LanguageCode
The language of the transcript in the source audio file.
Type: String
Required: No

MedicalTranscriptionJobName
The name of a medical transcription job.
Type: String
Pattern: ^[0-9a-zA-Z._-]+$
Required: No

OutputLocationType
Indicates the location of the transcription job's output.
The CUSTOMER_BUCKET is the S3 location provided in the OutputBucketName field when the
Type: String
Valid Values: CUSTOMER_BUCKET | SERVICE_BUCKET
Required: No

Specialty
The medical specialty of the transcription job. Primary care is the only valid value.
Type: String
Valid Values: PRIMARYCARE
Required: No

StartTime
A timestamp that shows when the job began processing.
Type: Timestamp
Required: No

TranscriptionJobStatus
The status of the medical transcription job.
Type: String
Valid Values: QUEUED | IN_PROGRESS | FAILED | COMPLETED
Required: No

Type
The speech of the clinician in the input audio.
Type: String
Valid Values: CONVERSATION | DICTATION
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
MedicalTranscriptionSetting
Service: Amazon Transcribe Service

Optional settings for the StartMedicalTranscriptionJob (p. 292) operation.

Contents

ChannelIdentification

Instructs Amazon Transcribe Medical to process each audio channel separately and then merge the transcription output of each channel into a single transcription.

Amazon Transcribe Medical also produces a transcription of each item detected on an audio channel, including the start time and end time of the item and alternative transcriptions of item. The alternative transcriptions also come with confidence scores provided by Amazon Transcribe Medical.

You can't set both ShowSpeakerLabels and ChannelIdentification in the same request. If you set both, your request returns a BadRequestException.

Type: Boolean
Required: No

MaxAlternatives

The maximum number of alternatives that you tell the service to return. If you specify the MaxAlternatives field, you must set the ShowAlternatives field to true.

Type: Integer
Required: No

MaxSpeakerLabels

The maximum number of speakers to identify in the input audio. If there are more speakers in the audio than this number, multiple speakers are identified as a single speaker. If you specify the MaxSpeakerLabels field, you must set the ShowSpeakerLabels field to true.

Type: Integer
Required: No

ShowAlternatives

Determines whether alternative transcripts are generated along with the transcript that has the highest confidence. If you set ShowAlternatives field to true, you must also set the maximum number of alternatives to return in the MaxAlternatives field.

Type: Boolean
Required: No

ShowSpeakerLabels

Determines whether the transcription job uses speaker recognition to identify different speakers in the input audio. Speaker recognition labels individual speakers in the audio file. If you set the ShowSpeakerLabels field to true, you must also set the maximum number of speaker labels in the MaxSpeakerLabels field.
You can't set both `ShowSpeakerLabels` and `ChannelIdentification` in the same request. If you set both, your request returns a `BadRequestException`.

Type: Boolean
Required: No

**VocabularyName**

The name of the vocabulary to use when processing a medical transcription job.

Type: String
Pattern: `^[0-9a-zA-Z._-]+$`
Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
ModelSettings

Service: Amazon Transcribe Service

The object used to call your custom language model to your transcription job.

Contents

LanguageModelName

The name of your custom language model.

Type: String


Pattern: /^[\-09a-zA-Z_\-]+$/

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
Settings
Service: Amazon Transcribe Service

Provides optional settings for the StartTranscriptionJob operation.

Contents

ChannelIdentification

Instructs Amazon Transcribe to process each audio channel separately and then merge the transcription output of each channel into a single transcription.

Amazon Transcribe also produces a transcription of each item detected on an audio channel, including the start time and end time of the item and alternative transcriptions of the item including the confidence that Amazon Transcribe has in the transcription.

You can't set both ShowSpeakerLabels and ChannelIdentification in the same request. If you set both, your request returns a BadRequestException.

Type: Boolean
Required: No

MaxAlternatives

The number of alternative transcriptions that the service should return. If you specify the MaxAlternatives field, you must set the ShowAlternatives field to true.

Type: Integer
Required: No

MaxSpeakerLabels

The maximum number of speakers to identify in the input audio. If there are more speakers in the audio than this number, multiple speakers are identified as a single speaker. If you specify the MaxSpeakerLabels field, you must set the ShowSpeakerLabels field to true.

Type: Integer
Required: No

ShowAlternatives

Determines whether the transcription contains alternative transcriptions. If you set the ShowAlternatives field to true, you must also set the maximum number of alternatives to return in the MaxAlternatives field.

Type: Boolean
Required: No

ShowSpeakerLabels

Determines whether the transcription job uses speaker recognition to identify different speakers in the input audio. Speaker recognition labels individual speakers in the audio file. If you set the ShowSpeakerLabels field to true, you must also set the maximum number of speaker labels MaxSpeakerLabels field.
You can't set both `ShowSpeakerLabels` and `ChannelIdentification` in the same request. If you set both, your request returns a `BadRequestException`.

**Type**: Boolean

Required: No

**VocabularyFilterMethod**

Set to `mask` to remove filtered text from the transcript and replace it with three asterisks ("***") as placeholder text. Set to `remove` to remove filtered text from the transcript without using placeholder text.

**Type**: String

Required: No

Valid Values: remove | mask

**VocabularyFilterName**

The name of the vocabulary filter to use when transcribing the audio. The filter that you specify must have the same language code as the transcription job.

**Type**: String


Pattern: `^[0-9a-zA-Z._-]+$`

Required: No

**VocabularyName**

The name of a vocabulary to use when processing the transcription job.

**Type**: String


Pattern: `^[0-9a-zA-Z._-]+$`

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
Transcript
Service: Amazon Transcribe Service

Identifies the location of a transcription.

Contents

RedactedTranscriptFileUri

The S3 object location of the redacted transcript.

Use this URI to access the redacted transcript. If you specified an S3 bucket in the OutputBucketName field when you created the job, this is the URI of that bucket. If you chose to store the transcript in Amazon Transcribe, this is a shareable URL that provides secure access to that location.

Type: String


Pattern: (s3://|http(s*)://).+

Required: No

TranscriptFileUri

The S3 object location of the transcript.

Use this URI to access the transcript. If you specified an S3 bucket in the OutputBucketName field when you created the job, this is the URI of that bucket. If you chose to store the transcript in Amazon Transcribe, this is a shareable URL that provides secure access to that location.

Type: String


Pattern: (s3://|http(s*)://).+

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
TranscriptionJob
Service: Amazon Transcribe Service

Describes an asynchronous transcription job that was created with the StartTranscriptionJob operation.

Contents

CompletionTime
A timestamp that shows when the job was completed.
Type: Timestamp
Required: No

ContentRedaction
An object that describes content redaction settings for the transcription job.
Type: ContentRedaction (p. 326) object
Required: No

CreationTime
A timestamp that shows when the job was created.
Type: Timestamp
Required: No

FailureReason
If the TranscriptionJobStatus field is FAILED, this field contains information about why the job failed.

The FailureReason field can contain one of the following values:
• Unsupported media format - The media format specified in the MediaFormat field of the request isn't valid. See the description of the MediaFormat field for a list of valid values.
• The media format provided does not match the detected media format - The media format of the audio file doesn't match the format specified in the MediaFormat field in the request. Check the media format of your media file and make sure that the two values match.
• Invalid sample rate for audio file - The sample rate specified in the MediaSampleRateHertz of the request isn't valid. The sample rate must be between 8000 and 48000 Hertz.
• The sample rate provided does not match the detected sample rate - The sample rate in the audio file doesn't match the sample rate specified in the MediaSampleRateHertz field in the request. Check the sample rate of your media file and make sure that the two values match.
• Invalid file size: file size too large - The size of your audio file is larger than Amazon Transcribe can process. For more information, see Limits in the Amazon Transcribe Developer Guide.
• Invalid number of channels: number of channels too large - Your audio contains more channels than Amazon Transcribe is configured to process. To request additional channels, see Amazon Transcribe Limits in the Amazon Web Services General Reference.

Type: String
Required: No
IdentifiedLanguageScore

A value between zero and one that Amazon Transcribe assigned to the language that it identified in the source audio. Larger values indicate that Amazon Transcribe has higher confidence in the language it identified.

Type: Float

Required: No

IdentifyLanguage

A value that shows if automatic language identification was enabled for a transcription job.

Type: Boolean

Required: No

JobExecutionSettings

Provides information about how a transcription job is executed.

Type: JobExecutionSettings (p. 328) object

Required: No

LanguageCode

The language code for the input speech.

Type: String


Required: No

LanguageOptions

An object that shows the optional array of languages inputted for transcription jobs with automatic language identification enabled.

Type: Array of strings

Array Members: Minimum number of 2 items.


Required: No

Media

An object that describes the input media for the transcription job.

Type: Media (p. 331) object

Required: No

MediaFormat

The format of the input media file.
Type: String

Valid Values: mp3 | mp4 | wav | flac | ogg | amr | webm

Required: No

**MediaSampleRateHertz**

The sample rate, in Hertz, of the audio track in the input media file.

Type: Integer


Required: No

**ModelSettings**

An object containing the details of your custom language model.

Type: ModelSettings (p. 340) object

Required: No

**Settings**

Optional settings for the transcription job. Use these settings to turn on speaker recognition, to set the maximum number of speakers that should be identified and to specify a custom vocabulary to use when processing the transcription job.

Type: Settings (p. 341) object

Required: No

**StartTime**

A timestamp that shows when the job was started processing.

Type: Timestamp

Required: No

**Transcript**

An object that describes the output of the transcription job.

Type: Transcript (p. 343) object

Required: No

**TranscriptionJobName**

The name of the transcription job.

Type: String


Pattern: ^[0-9a-zA-Z_\-.]+$

Required: No

**TranscriptionJobStatus**

The status of the transcription job.

Type: String
Valid Values: QUEUED | IN_PROGRESS | FAILED | COMPLETED

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
TranscriptionJobSummary
Service: Amazon Transcribe Service

Provides a summary of information about a transcription job.

Contents

CompletionTime
A timestamp that shows when the job was completed.
Type: Timestamp
Required: No

ContentRedaction
The content redaction settings of the transcription job.
Type: ContentRedaction (p. 326) object
Required: No

CreationTime
A timestamp that shows when the job was created.
Type: Timestamp
Required: No

FailureReason
If the TranscriptionJobStatus field is FAILED, a description of the error.
Type: String
Required: No

IdentifiedLanguageScore
A value between zero and one that Amazon Transcribe assigned to the language it identified in the source audio. A higher score indicates that Amazon Transcribe is more confident in the language it identified.
Type: Float
Required: No

IdentifyLanguage
Whether automatic language identification was enabled for a transcription job.
Type: Boolean
Required: No

LanguageCode
The language code for the input speech.
Type: String

ModelSettings

The object used to call your custom language model to your transcription job.

Type: ModelSettings (p. 340) object

Required: No

OutputLocationType

Indicates the location of the output of the transcription job.

If the value is CUSTOMER_BUCKET then the location is the S3 bucket specified in the outputBucketName field when the transcription job was started with the StartTranscriptionJob operation.

If the value is SERVICE_BUCKET then the output is stored by Amazon Transcribe and can be retrieved using the URI in the GetTranscriptionJob response's TranscriptFileUri field.

Type: String

Valid Values: CUSTOMER_BUCKET | SERVICE_BUCKET

Required: No

StartTime

A timestamp that shows when the job started processing.

Type: Timestamp

Required: No

TranscriptionJobName

The name of the transcription job.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: No

TranscriptionJobStatus

The status of the transcription job. When the status is COMPLETED, use the GetTranscriptionJob operation to get the results of the transcription.

Type: String

Valid Values: QUEUED | IN_PROGRESS | FAILED | COMPLETED

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
VocabularyFilterInfo
Service: Amazon Transcribe Service

Provides information about a vocabulary filter.

Contents

LanguageCode
The language code of the words in the vocabulary filter.
Type: String
Required: No

LastModifiedTime
The date and time that the vocabulary was last updated.
Type: Timestamp
Required: No

VocabularyFilterName
The name of the vocabulary filter. The name must be unique in the account that holds the filter.
Type: String
Pattern: ^[0-9a-zA-Z._-]+$
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
VocabularyInfo
Service: Amazon Transcribe Service

Provides information about a custom vocabulary.

Contents

LanguageCode

The language code of the vocabulary entries.

Type: String


Required: No

LastModifiedTime

The date and time that the vocabulary was last modified.

Type: Timestamp

Required: No

VocabularyName

The name of the vocabulary.

Type: String


Pattern: ^[0-9a-zA-Z._-]+$

Required: No

VocabularyState

The processing state of the vocabulary. If the state is READY you can use the vocabulary in a StartTranscriptionJob request.

Type: String

Valid Values: PENDING | READY | FAILED

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
Amazon Transcribe Streaming Service

The following data types are supported by Amazon Transcribe Streaming Service:

- Alternative (p. 354)
- AudioEvent (p. 355)
- AudioStream (p. 356)
- Item (p. 357)
- MedicalAlternative (p. 359)
- MedicalItem (p. 360)
- MedicalResult (p. 362)
- MedicalTranscript (p. 364)
- MedicalTranscriptEvent (p. 365)
- MedicalTranscriptResultStream (p. 366)
- Result (p. 368)
- Transcript (p. 370)
- TranscriptEvent (p. 371)
- TranscriptResultStream (p. 372)
Alternative
Service: Amazon Transcribe Streaming Service
A list of possible transcriptions for the audio.

Contents

Items
One or more alternative interpretations of the input audio.
Type: Array of Item (p. 357) objects
Required: No

Transcript
The text that was transcribed from the audio.
Type: String
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
AudioEvent
Service: Amazon Transcribe Streaming Service

Provides a wrapper for the audio chunks that you are sending.

Contents

AudioChunk

An audio blob that contains the next part of the audio that you want to transcribe. The maximum audio chunk size is 32 KB.

Type: Base64-encoded binary data object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
**AudioStream**

Service: Amazon Transcribe Streaming Service

Represents the audio stream from your application to Amazon Transcribe.

**Contents**

**AudioEvent**

A blob of audio from your application. You audio stream consists of one or more audio events. For information on audio encoding formats, see *Speech Input* (p. 5). For more information on stream encoding, see *Event Stream Encoding* (p. 76).

Type: AudioEvent (p. 355) object

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
Item
Service: Amazon Transcribe Streaming Service

A word or phrase transcribed from the input audio.

Contents

Content
The word or punctuation that was recognized in the input audio.
Type: String
Required: No

EndTime
The offset from the beginning of the audio stream to the end of the audio that resulted in the item.
Type: Double
Required: No

Speaker
If speaker identification is enabled, shows the speakers identified in the real-time stream.
Type: String
Required: No

StartTime
The offset from the beginning of the audio stream to the beginning of the audio that resulted in the item.
Type: Double
Required: No

Type
The type of the item. PRONUNCIATION indicates that the item is a word that was recognized in the input audio. PUNCTUATION indicates that the item was interpreted as a pause in the input audio.
Type: String
Valid Values: pronunciation | punctuation
Required: No

VocabularyFilterMatch
Indicates whether a word in the item matches a word in the vocabulary filter you've chosen for your real-time stream. If true then a word in the item matches your vocabulary filter.
Type: Boolean
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java V2
• AWS SDK for Ruby V3
MedicalAlternative
Service: Amazon Transcribe Streaming Service

A list of possible transcriptions for the audio.

Contents

Items

A list of objects that contains words and punctuation marks that represents one or more interpretations of the input audio.

Type: Array of MedicalItem (p. 360) objects

Required: No

Transcript

The text that was transcribed from the audio.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
MedicalItem
Service: Amazon Transcribe Streaming Service

A word or punctuation that is transcribed from the input audio.

Contents

Confidence

A value between 0 and 1 for an item that is a confidence score that Amazon Transcribe Medical assigns to each word that it transcribes.

Type: Double
Required: No

Content

The word or punctuation mark that was recognized in the input audio.

Type: String
Required: No

EndTime

The number of seconds into an audio stream that indicates the creation time of an item.

Type: Double
Required: No

Speaker

If speaker identification is enabled, shows the integer values that correspond to the different speakers identified in the stream. For example, if the value of Speaker in the stream is either a 0 or a 1, that indicates that Amazon Transcribe Medical has identified two speakers in the stream. The value of 0 corresponds to one speaker and the value of 1 corresponds to the other speaker.

Type: String
Required: No

StartTime

The number of seconds into an audio stream that indicates the creation time of an item.

Type: Double
Required: No

Type

The type of the item. PRONUNCIATION indicates that the item is a word that was recognized in the input audio. PUNCTUATION indicates that the item was interpreted as a pause in the input audio, such as a period to indicate the end of a sentence.

Type: String
Valid Values: pronunciation | punctuation
Required: No
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
MedicalResult
Service: Amazon Transcribe Streaming Service

The results of transcribing a portion of the input audio stream.

Contents

Alternatives

A list of possible transcriptions of the audio. Each alternative typically contains one Item that contains the result of the transcription.

Type: Array of MedicalAlternative (p. 359) objects

Required: No

ChannelId

When channel identification is enabled, Amazon Transcribe Medical transcribes the speech from each audio channel separately.

You can use ChannelId to retrieve the transcription results for a single channel in your audio stream.

Type: String

Required: No

EndTime

The time, in seconds, from the beginning of the audio stream to the end of the result.

Type: Double

Required: No

IsPartial

Amazon Transcribe Medical divides the incoming audio stream into segments at natural points in the audio. Transcription results are returned based on these segments.

The IsPartial field is true to indicate that Amazon Transcribe Medical has additional transcription data to send. The IsPartial field is false to indicate that this is the last transcription result for the segment.

Type: Boolean

Required: No

ResultId

A unique identifier for the result.

Type: String

Required: No

StartTime

The time, in seconds, from the beginning of the audio stream to the beginning of the result.

Type: Double

Required: No
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
MedicalTranscript
Service: Amazon Transcribe Streaming Service

The medical transcript in a MedicalTranscriptEvent (p. 365).

Contents

Results

MedicalResult (p. 362) objects that contain the results of transcribing a portion of the input audio stream. The array can be empty.

Type: Array of MedicalResult (p. 362) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
MedicalTranscriptEvent
Service: Amazon Transcribe Streaming Service

Represents a set of transcription results from the server to the client. It contains one or more segments of the transcription.

Contents

Transcript

The transcription of the audio stream. The transcription is composed of all of the items in the results list.

Type: MedicalTranscript (p. 364) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
MedicalTranscriptResultStream
Service: Amazon Transcribe Streaming Service

Represents the transcription result stream from Amazon Transcribe Medical to your application.

Contents

BadRequestException

One or more arguments to the StartStreamTranscription or StartMedicalStreamTranscription operation was invalid. For example, MediaEncoding was not set to a valid encoding, or LanguageCode was not set to a valid code. Check the parameters and try your request again.

Type: Exception
HTTP Status Code: 400

Required: No

ConflictException

A new stream started with the same session ID. The current stream has been terminated.

Type: Exception
HTTP Status Code: 409

Required: No

InternalFailureException

A problem occurred while processing the audio. Amazon Transcribe or Amazon Transcribe Medical terminated processing. Try your request again.

Type: Exception
HTTP Status Code: 500

Required: No

LimitExceededException

You have exceeded the maximum number of concurrent transcription streams, are starting transcription streams too quickly, or the maximum audio length of 4 hours. Wait until a stream has finished processing, or break your audio stream into smaller chunks and try your request again.

Type: Exception
HTTP Status Code: 429

Required: No

ServiceUnavailableException

Service is currently unavailable. Try your request later.

Type: Exception
HTTP Status Code: 503

Required: No

TranscriptEvent

A portion of the transcription of the audio stream. Events are sent periodically from Amazon Transcribe Medical to your application. The event can be a partial transcription of a section of the audio stream, or it can be the entire transcription of that portion of the audio stream.
Type: MedicalTranscriptEvent (p. 365) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
Result
Service: Amazon Transcribe Streaming Service

The result of transcribing a portion of the input audio stream.

Contents

Alternatives

A list of possible transcriptions for the audio. Each alternative typically contains one item that contains the result of the transcription.

Type: Array of Alternative (p. 354) objects

Required: No

ChannelId

When channel identification is enabled, Amazon Transcribe transcribes the speech from each audio channel separately.

You can use ChannelId to retrieve the transcription results for a single channel in your audio stream.

Type: String

Required: No

EndTime

The offset in seconds from the beginning of the audio stream to the end of the result.

Type: Double

Required: No

IsPartial

Amazon Transcribe divides the incoming audio stream into segments at natural points in the audio. Transcription results are returned based on these segments.

The IsPartial field is true to indicate that Amazon Transcribe has additional transcription data to send, false to indicate that this is the last transcription result for the segment.

Type: Boolean

Required: No

ResultId

A unique identifier for the result.

Type: String

Required: No

StartTime

The offset in seconds from the beginning of the audio stream to the beginning of the result.

Type: Double

Required: No
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
Transcript
Service: Amazon Transcribe Streaming Service

The transcription in a TranscriptEvent (p. 371).

Contents

Results

Result (p. 368) objects that contain the results of transcribing a portion of the input audio stream. The array can be empty.

Type: Array of Result (p. 368) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
TranscriptEvent
Service: Amazon Transcribe Streaming Service

Represents a set of transcription results from the server to the client. It contains one or more segments of the transcription.

Contents

Transcript

The transcription of the audio stream. The transcription is composed of all of the items in the results list.

Type: Transcript (p. 370) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3
TranscriptResultStream

Service: Amazon Transcribe Streaming Service

Represents the transcription result stream from Amazon Transcribe to your application.

Contents

BadRequestException

A client error occurred when the stream was created. Check the parameters of the request and try your request again.

Type: Exception
HTTP Status Code: 400

Required: No

ConflictException

A new stream started with the same session ID. The current stream has been terminated.

Type: Exception
HTTP Status Code: 409

Required: No

InternalFailureException

A problem occurred while processing the audio. Amazon Transcribe terminated processing.

Type: Exception
HTTP Status Code: 500

Required: No

LimitExceededException

Your client has exceeded one of the Amazon Transcribe limits, typically the limit on audio length. Break your audio stream into smaller chunks and try your request again.

Type: Exception
HTTP Status Code: 429

Required: No

ServiceUnavailableException

Service is currently unavailable. Try your request later.

Type: Exception
HTTP Status Code: 503

Required: No

TranscriptEvent

A portion of the transcription of the audio stream. Events are sent periodically from Amazon Transcribe to your application. The event can be a partial transcription of a section of the audio stream, or it can be the entire transcription of that portion of the audio stream.

Type: TranscriptEvent (p. 371) object

Required: No
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

InvalidParameterCombination

Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

InvalidQueryParameter

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400
MalformedQueryString
The query string contains a syntax error.
HTTP Status Code: 404

MissingAction
The request is missing an action or a required parameter.
HTTP Status Code: 400

MissingAuthenticationToken
The request must contain either a valid (registered) AWS access key ID or X.509 certificate.
HTTP Status Code: 403

MissingParameter
A required parameter for the specified action is not supplied.
HTTP Status Code: 400

NotAuthorized
You do not have permission to perform this action.
HTTP Status Code: 400

OptInRequired
The AWS access key ID needs a subscription for the service.
HTTP Status Code: 403

RequestExpired
The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.
HTTP Status Code: 400

ServiceUnavailable
The request has failed due to a temporary failure of the server.
HTTP Status Code: 503

ThrottlingException
The request was denied due to request throttling.
HTTP Status Code: 400

ValidationException
The input fails to satisfy the constraints specified by an AWS service.
HTTP Status Code: 400

Common Parameters
The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more
information about Signature Version 4, see Signature Version 4 Signing Process in the Amazon Web Services General Reference.

**Action**

The action to be performed.

**Type:** string

**Required:** Yes

**Version**

The API version that the request is written for, expressed in the format YYYY-MM-DD.

**Type:** string

**Required:** Yes

**X-Amz-Algorithm**

The hash algorithm that you used to create the request signature.

**Condition:** Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

**Type:** string

**Valid Values:** AWS4-HMAC-SHA256

**Required:** Conditional

**X-Amz-Credential**

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: access_key/YYYYMMDD/region/service/aws4_request.

For more information, see Task 2: Create a String to Sign for Signature Version 4 in the Amazon Web Services General Reference.

**Condition:** Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

**Type:** string

**Required:** Conditional

**X-Amz-Date**

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: 20120325T120000Z.

**Condition:** X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see Handling Dates in Signature Version 4 in the Amazon Web Services General Reference.

**Type:** string

**Required:** Conditional
X-Amz-Security-Token

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS Security Token Service, go to AWS Services That Work with IAM in the IAM User Guide.

Condition: If you're using temporary security credentials from the AWS Security Token Service, you must include the security token.

Type: string

Required: Conditional

X-Amz-Signature

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-SignedHeaders

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see Task 1: Create a Canonical Request For Signature Version 4 in the Amazon Web Services General Reference.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional
AWS glossary

For the latest AWS terminology, see the AWS glossary in the AWS General Reference.