Developer Guide

Amazon Transcribe

Amazon Transcribe: Developer Guide

Table of Contents

What is Amazon Transcribe?

Amazon Transcribe is an automatic speech recognition service that uses machine learning models to convert audio to text. You can use Amazon Transcribe as a standalone transcription service or to add speech-to-text capabilities to any application.

With Amazon Transcribe, you can improve accuracy for your specific use case with language customization, filter content to ensure customer privacy or audience-appropriate language, analyze content in multi-channel audio, partition the speech of individual speakers, and more.

You can transcribe media in real time (streaming) or you can transcribe media files located in an Amazon S3 bucket (batch). To see which languages are supported for each type of transcription, refer to the Supported languages and language-specific features table.

Topics

To learn more, see How Amazon Transcribe works and Getting started with Amazon Transcribe.

Tip

Information on the **Amazon Transcribe API** is located in the API Reference.

Amazon Transcribe and HIPAA eligibility

Amazon Transcribe is covered under Amazon's HIPAA eligibility and BAA which requires BAA customers to encrypt all PHI at rest and in transit when in use. Automatic PHI identification is available at no additional charge and in all regions where Amazon Transcribe operates. For more information, refer to HIPAA eligibility and BAA.

Pricing

Amazon Transcribe is a pay-as-you-go service; pricing is based on seconds of transcribed audio, billed on a monthly basis.

Usage is billed in one-second increments, with a minimum per request charge of 15 seconds. Note that additional charges apply for features such as PII content redaction and custom language models.

For cost information for each Amazon Web Services Region, refer to Amazon Transcribe Pricing.

Region availability and quotas

Amazon Transcribe is supported in the following Amazon Web Services Regions:

Region	Transcription type
af-south-1 (Cape Town)	batch, streaming
ap-east-1 (Hong Kong)	batch
ap-northeast-1 (Tokyo)	batch, streaming
ap-northeast-2 (Seoul)	batch, streaming
ap-south-1 (Mumbai)	batch, streaming
ap-southeast-1 (Singapore)	batch, streaming
ap-southeast-2 (Sydney)	batch, streaming
ap-southeast-5 (Malaysia)	streaming
ap-southeast-7 (Thailand)	streaming
ca-central-1 (Canada, Central)	batch, streaming
cn-north-1	batch
cn-northwest-1	batch
eu-central-1 (Frankfurt)	batch, streaming
eu-north-1 (Stockholm)	batch
eu-west-1 (Ireland)	batch, streaming

Region	Transcription type
eu-west-2 (London)	batch, streaming
eu-west-3 (Paris)	batch
me-south-1 (Bahrain)	batch
sa-east-1 (São Paulo)	batch, streaming
us-east-1 (N. Virginia)	batch, streaming
us-east-2 (Ohio)	batch, streaming
us-gov-east-1 (GovCloud, US-East)	batch, streaming
us-gov-west-1 (GovCloud, US-West)	batch, streaming
us-west-1 (San Francisco)	batch
us-west-2 (Oregon)	batch, streaming

Important

Region support differs for Amazon Transcribe, Amazon Transcribe Medical, and Call Analytics.

To get the endpoints for each supported Region, see Service endpoints in the *Amazon Web Services General Reference*.

For a list of quotas that pertain to your transcriptions, refer to the Service quotas in the *Amazon Web Services General Reference*. Some quotas can be changed upon request. If the **Adjustable** column contains '**Yes**', you can request an increase. To do so, select the provided link.

Supported languages and language-specific features

The languages supported by Amazon Transcribe are listed in the following table; also listed are the features that are language-specific. Please verify that the feature you want to use is supported for the language in your media before proceeding with your transcription.

In the following table, 'batch' refers to transcribing a media file located in an Amazon S3 bucket and 'streaming' refers to transcribing streamed media in real time. For Call Analytics transcriptions, 'post-call' refers to transcribing a media file located in an Amazon S3 bucket and 'real-time' refers to transcribing streamed media in real time.

Language	Language code	Data input	Transcribing numbers	Acronyms	Custom language models*	Redaction *	Call Analytics *
Abkhaz	`ab-GE`	batch	no	batch	no	no	no
Afrikaans	`af-ZA`	batch, streaming	no	batch, streaming	no	no	no
Arabic, Gulf	`ar-AE`	batch, streaming	no	no	no	no	post-call
Arabic, Modern Standard	`ar-SA`	batch, streaming	no	no	no	no	no
Armenian	`hy-AM`	batch	no	batch	no	no	no
Asturian	`ast-ES`	batch	no	batch	no	no	no
Azerbaijani	`az-AZ`	batch	no	batch	no	no	no
Bashkir	`ba-RU`	batch	no	batch	no	no	no
Basque	`eu-ES`	batch, streaming	no	batch, streaming	no	no	no

Language	Language code	Data input	Transcribing numbers	Acronyms	Custom language models*	Redaction *	Call Analytics *
Belarusian	be-BY	batch	no	batch	no	no	no
Bengali	bn-IN	batch	no	batch	no	no	no
Bosnian	bs-BA	batch	no	batch	no	no	no
Bulgarian	bg-BG	batch	no	batch	no	no	no
Catalan	ca-ES	batch, streaming	streaming	batch, streaming	no	no	no
Central Kurdish, Iran	ckb-IR	batch	no	batch	no	no	no
Central Kurdish, Iraq	ckb-IQ	batch	no	batch	no	no	no
Chinese, Cantonese	zh-HK (yue-HK)	batch, streaming	no	no	no	no	no
Chinese, Simplified	zh-CN	batch, streaming	no	no	no	no	post-call
Chinese, Traditional	zh-TW	batch, streaming	no	no	no	no	no
Croatian	hr-HR	batch, streaming	no	batch, streaming	no	no	no

Language	Language code	Data input	Transcribing numbers	Acronyms	Custom language models*	Redaction *	Call Analytics *
Czech	cs-CZ	batch, streaming	no	batch, streaming	no	no	no
Danish	da-DK	batch, streaming	no	batch, streaming	no	no	no
Dutch	nl-NL	batch, streaming	no	batch, streaming	no	no	no
English, Australian	en-AU	batch, streaming	batch, streaming	batch, streaming	batch, streaming	streaming	post-call, real-time
English, British	en-GB	batch, streaming	batch, streaming	batch, streaming	batch, streaming	streaming	post-call, real-time
English, Indian	en-IN	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call
English, Irish	en-IE	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call
English, New Zealand	en-NZ	batch, streaming	batch, streaming	batch, streaming	no	streaming	no
English, Scottish	en-AB	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call
English, South African	en-ZA	batch, streaming	batch, streaming	batch, streaming	no	streaming	no
English, US	en-US	batch, streaming	batch, streaming	batch, streaming	batch, streaming	batch, streaming	post-call, real-time

Language	Language code	Data input	Transcribing numbers	Acronyms	Custom language models*	Redaction *	Call Analytics *
English, Welsh	en-WL	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call
Estonian	et-EE	batch	no	batch	no	no	no
Estonian	et-ET	batch	no	batch	no	no	no
Farsi	fa-IR	batch, streaming	no	no	no	no	no
Finnish	fi-FI	batch, streaming	no	batch, streaming	no	no	no
French	fr-FR	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call, real-time
French, Canadian	fr-CA	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call, real-time
Galician	gl-ES	batch, streaming	no	batch, streaming	no	no	no
Georgian	ka-GE	batch	no	batch	no	no	no
German	de-DE	batch, streaming	batch, streaming	batch, streaming	batch, streaming	streaming	post-call, real-time
German, Swiss	de-CH	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call
Greek	el-GR	batch, streaming	no	batch, streaming	no	no	no
Gujarati	gu-IN	batch	no	batch	no	no	no
Hausa	ha-NG	batch	no	batch	no	no	no

Language	Language code	Data input	Transcribing numbers	Acronyms	Custom language models*	Redaction *	Call Analytics *
Hebrew	`he-IL`	batch, streaming	no	no	no	no	no
Hindi, Indian	`hi-IN`	batch, streaming	no	batch, streaming	batch	no	post-call
Hungarian	`hu-HU`	batch	no	batch	no	no	no
Icelandic	`is-IS`	batch	no	batch	no	no	no
Indonesian	`id-ID`	batch, streaming	no	batch, streaming	no	no	no
Italian	`it-IT`	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call, real-time
Japanese	`ja-JP`	batch, streaming	batch, streaming	no	batch, streaming	no	post-call
Kabyle	`kab-DZ`	batch	no	batch	no	no	no
Kannada	`kn-IN`	batch	no	batch	no	no	no
Kazakh	`kk-KZ`	batch	no	batch	no	no	no
Kinyarwanda	`rw-RW`	batch	no	batch	no	no	no
Korean	`ko-KR`	batch, streaming	no	no	no	no	post-call
Kyrgyz	`ky-KG`	batch	no	batch	no	no	no
Latvian	`lv-LV`	batch, streaming	no	batch, streaming	no	no	no

Language	Language code	Data input	Transcribing numbers	Acronyms	Custom language models*	Redaction *	Call Analytics *
Lithuanian	`lt-LT`	batch	no	batch	no	no	no
Luganda	`lg-IN`	batch	no	batch	no	no	no
Macedonian	`mk-MK`	batch	no	batch	no	no	no
Malay	`ms-MY`	batch, streaming	no	batch, streaming	no	no	no
Malayalam	`ml-IN`	batch	no	batch	no	no	no
Maltese	`mt-MT`	batch	no	batch	no	no	no
Marathi	`mr-IN`	batch	no	batch	no	no	no
Meadow Mari	`mhr-RU`	batch	no	batch	no	no	no
Mongolian	`mn-MN`	batch	no	batch	no	no	no
Norwegian Bokmål	`no-NO`	batch, streaming	no	batch, streaming	no	no	no
Odia/ Oriya	`or-IN`	batch	no	batch	no	no	no
Pashto	`ps-AF`	batch	no	batch	no	no	no
Polish	`pl-PL`	batch, streaming	no	batch, streaming	no	no	no
Portuguese	`pt-PT`	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call

Language	Language code	Data input	Transcribing numbers	Acronyms	Custom language models*	Redaction *	Call Analytics *
Portuguese, Brazilian	pt-BR	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call, real-time
Punjabi	pa-IN	batch	no	batch	no	no	no
Romanian	ro-RO	batch, streaming	no	batch, streaming	no	no	no
Russian	ru-RU	batch, streaming	no	no	no	no	no
Serbian	sr-RS	batch, streaming	no	batch, streaming	no	no	no
Sinhala	si-LK	batch	no	batch	no	no	no
Slovak	sk-SK	batch, streaming	no	batch, streaming	no	no	no
Slovenian	sl-SI	batch	no	batch	no	no	no
Somali	so-SO	batch, streaming	no	batch, streaming	no	no	no
Spanish	es-ES	batch, streaming	batch, streaming	batch, streaming	no	streaming	post-call
Spanish, US	es-US	batch, streaming	batch, streaming	batch, streaming	batch, streaming	batch, streaming	post-call, real-time
Sundanese	su-ID	batch	no	batch	no	no	no
Swahili, Kenya	sw-KE	batch	no	batch	no	no	no

Language	Language code	Data input	Transcribing numbers	Acronyms	Custom language models*	Redaction *	Call Analytics *
Swahili, Burundi	sw-BI	batch	no	batch	no	no	no
Swahili, Rwanda	sw-RW	batch	no	batch	no	no	no
Swahili, Tanzania	sw-TZ	batch	no	batch	no	no	no
Swahili, Uganda	sw-UG	batch	no	batch	no	no	no
Swedish	sv-SE	batch, streaming	no	batch, streaming	no	no	no
Tagalog/ Filipino	tl-PH	batch, streaming	no	batch, streaming	no	no	no
Tamil	ta-IN	batch	no	no	no	no	no
Tatar	tt-RU	batch	no	batch	no	no	no
Telugu	te-IN	batch	no	no	no	no	no
Thai	th-TH	batch, streaming	no	batch, streaming	no	no	no
Turkish	tr-TR	batch	no	batch	no	no	no
Ukrainian	uk-UA	batch, streaming	no	batch, streaming	no	no	no
Uyghur	ug-CN	batch	no	batch	no	no	no
Uzbek	uz-UZ	batch	no	batch	no	no	no

Language	Language code	Data input	Transcribing numbers	Acronyms	Custom language models*	Redaction *	Call Analytics *
Vietnamese	`vi-VN`	batch, streaming	no	batch, streaming	no	no	no
Welsh	`cy-WL`	batch	no	batch	no	no	no
Wolof	`wo-SN`	batch	no	batch	no	no	no
Zulu	`zu-ZA`	batch, streaming	no	batch, streaming	no	no	no

- Call summarization: `en-*` (all English dialects)
- Issue detection: `en-AU`, `en-GB`, `en-US`

* This feature is not available in all regions.

Supported programming languages

Amazon Transcribe supports the following Amazon SDKs:

Batch transcriptions	Streaming transcriptions
.NET	.NET (Transcribe Medical and HealthScribe is not supported)
Amazon Command Line Interface (CLI)	The CLI is not supported for streaming.
C++	C++
Go	Go
Java V2	Java V2
JavaScript	JavaScript V3

Batch transcriptions	Streaming transcriptions
PHP V3	The SDK is not supported for streaming.
Python Boto3	Python Streaming SDK for Amazon Transcribe
Ruby V3	Ruby V3
Rust	Rust

For information on using SDKs with Amazon Transcribe, refer to Transcribing with the Amazon SDKs.

For more information on all available Amazon SDKs and builder tools, refer to Tools to Build on Amazon.

Tip

You can find SDK code samples in these GitHub repositories:

- Amazon Code Examples
- Amazon Transcribe Examples

Character sets for custom vocabularies and vocabulary filters

For each language Amazon Transcribe supports, there is a specific set of characters Amazon Transcribe can recognize. When you create a custom vocabulary or vocabulary filter, use only the characters listed in your language's character set. If you use unsupported characters, your custom vocabulary or vocabulary filter fails.

Important

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

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

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

Topics

- Marathi character set
- Meadow Mari character set
- Mongolian character set
- Norwegian Bokmål character set
- Odia/Oriya character set
- Pashto character set
- Polish character set
- Portuguese character set
- Punjabi character set
- Romanian character set
- Russian character set
- Serbian character set
- Sinhala character set
- Slovak character set
- Slovenian character set
- Somali character set
- Spanish character set
- Sundanese character set
- Swahili character set
- Swedish character set
- Tagalog/Filipino character set
- Tamil character set
- Tatar character set
- Telugu character set
- Thai character set
- Turkish character set
- Ukrainian character set
- Uyghur character set
- Uzbek character set
- Vietnamese character set

- Welsh character set
- Wolof character set
- Zulu character set

Abkhaz character set

For Abkhaz custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

Character	Code	Character	Code
ѐ	0450	ӡ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
ѕ	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	ӷ	04F7
ј	0458	ӹ	04F9
#	0525		

Afrikaans character set

For Afrikaans custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
á	00E1	ï	00EF
è	00E8	ó	00F3
é	00E9	ô	00F4

Character	Code	Character	Code
ê	00EA	ö	00F6
ë	00EB	ú	00FA
í	00ED	û	00FB
î	00EE	ü	00FC

Arabic character set

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

Character	Code	Character	Code
ء	0621	س	0633
آ	0622	ش	0634
أ	0623	ص	0635
ؤ	0624	ض	0636
إ	0625	ط	0637
ئ	0626	ظ	0638
ا	0627	ع	0639
ب	0628	غ	063A
ة	0629	ف	0641
ت	062A	ق	0642
ث	062B	ك	0643
ج	062C	ل	0644

Character	Code	Character	Code
ح	062D	م	0645
خ	062E	ن	0646
د	062F	ه	0647
ذ	0630	و	0648
ر	0631	ى	0649
ز	0632	ي	064A

Asturian character set

For Asturian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Azerbaijani character set

For Azerbaijani custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the Phrase field:

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

Armenian character set

For Armenian custom vocabularies, you can use the following characters in the Phrase field:

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

You can also use the following Unicode characters in the Phrase field:

Character	Code	Character	Code
ա	0561	մ	0574
բ	0562	յ	0575
գ	0563	ն	0576
դ	0564	շ	0577

Character	Code	Character	Code
ե	0565	ո	0578
զ	0566	չ	0579
է	0567	պ	057A
ը	0568	ջ	057B
թ	0569	ռ	057C
ժ	056A	ս	057D
ի	056B	վ	057E
լ	056C	տ	057F
խ	056D	ր	0580
ծ	056E	ց	0581
կ	056F	ւ	0582
հ	0570	փ	0583
ձ	0571	ք	0584
ղ	0572	օ	0585
ճ	0573	ֆ	0586

Bashkir character set

For Bashkir custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
а	0430	љ	0459
б	0431	њ	045A
в	0432	ћ	045B
г	0433	ќ	045C
д	0434	ѝ	045D
е	0435	ў	045E
ж	0436	џ	045F
з	0437	ґ	0491
и	0438	ғ	0493
й	0439	җ	0497
к	043A	ҙ	0499
л	043B	қ	049B
м	043C	ҟ	049F
н	043D	ҡ	04A1
о	043E	ң	04A3
п	043F	ҥ	04A5
р	0440	ҩ	04A9
с	0441	ҫ	04AB
т	0442	ҭ	04AD

Character	Code	Character	Code
у	0443	ү	04AF
ф	0444	ұ	04B1
х	0445	ҳ	04B3
ц	0446	ҵ	04B5
ч	0447	ҷ	04B7
ш	0448	һ	04BB
щ	0449	ҽ	04BD
ъ	044A	ҿ	04BF
ы	044B	ӊ	04CA
ь	044C	ӑ	04D1
э	044D	ӓ	04D3
ю	044E	ӗ	04D7
я	044F	ә	04D9
ѐ	0450	ӡ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
ѕ	0455	ӱ	04F1
і	0456	ӳ	04F3

Character	Code	Character	Code
ї	0457	ӷ	04F7
ј	0458	ӹ	04F9

Basque character set

For Basque custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Belarusian character set

For Belarusian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
а	0430	с	0441
б	0431	т	0442
в	0432	у	0443
г	0433	ф	0444
д	0434	х	0445
е	0435	ц	0446
ж	0436	ч	0447
з	0437	ш	0448
й	0439	ы	044B
к	043A	ь	044C
л	043B	э	044D
м	043C	ю	044E
н	043D	я	044F
о	043E	ё	0451
п	043F	і	0456
р	0440	ў	045E

Bengali character set

For Bengali custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z
- - (hyphen)

- . (period)

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
ঁ	0981	দ	09A6
ং	0982	ধ	09A7
ঃ	0983	ন	09A8
অ	0985	প	09AA
আ	0986	ফ	09AB
ই	0987	ব	09AC
ঈ	0988	ভ	09AD
উ	0989	ম	09AE
ঊ	098A	য	09AF
ঋ	098B	র	09B0
এ	098F	ল	09B2
ঐ	0990	শ	09B6
ও	0993	ষ	09B7
ঔ	0994	স	09B8
ক	0995	হ	09B9
খ	0996	়	09BC
গ	0997	#	09BD
ঘ	0998	া	09BE

Character	Code	Character	Code
ঙ	0999	ি	09BF
চ	099A	ী	09C0
ছ	099B	ু	09C1
জ	099C	ূ	09C2
ঝ	099D	ৃ	09C3
ঞ	099E	ৄ	09C4
ট	099F	ে	09C7
ঠ	09A0	ৈ	09C8
ড	09A1	ো	09CB
ঢ	09A2	ৌ	09CC
ণ	09A3	্	09CD
ত	09A4	#	09CE
থ	09A5	ৗ	09D7

Bosnian character set

For Bosnian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Bulgarian character set

For Bulgarian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
а	0430	п	043F
б	0431	р	0440
в	0432	с	0441
г	0433	т	0442
д	0434	у	0443
е	0435	ф	0444
ж	0436	х	0445
з	0437	ц	0446
и	0438	ч	0447

Character	Code	Character	Code
й	0439	ш	0448
к	043A	щ	0449
л	043B	ъ	044A
м	043C	ь	044C
н	043D	ю	044E
о	043E	я	044F

Catalan character set

For Catalan custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Central Kurdish character set

For Central Kurdish custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
ئ	0626	م	0645
ا	0627	ن	0646
ب	0628	و	0648
ت	062A	پ	067E
ج	062C	چ	0686
ح	062D	ڕ	0695
خ	062E	ژ	0698
د	062F	ڤ	06A4
ر	0631	ک	06A9
ز	0632	گ	06AF
س	0633	ڵ	06B5
ش	0634	ھ	06BE
ع	0639	ۆ	06C6
غ	063A	ۇ	06C7

Character	Code	Character	Code
ف	0641	ی	06CC
ق	0642	ێ	06CE
ل	0644	ە	06D5

Chinese, Mandarin (Simplified) character set

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

- zh-cn-character-set

The `SoundsLike` field can contain the pinyin syllables listed in the following file:

- pinyin-character-set

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

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

Tone	Tone mark	Tone number
Tone 1	mā	ma1
Tone 2	má	ma2
Tone 3	mǎ	ma3
Tone 4	mà	ma4

Note

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

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

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

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

Chinese, Mandarin (Traditional) character set

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

- zh-tw-character-set

The `SoundsLike` field can contain the zhuyin syllables listed in the following file:

- zhuyin-character-set

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

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

Tone	Tone mark	
Tone 1	ㄇㄚ	

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

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

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

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

Chinese, Cantonese (Hong Kong), Traditional character set

For Chinese (Cantonese) Hong Kong custom vocabularies, the `Phrase` field can use any of the characters listed in the following file:

- zh-hk-character-set

Croatian character set

For Croatian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Czech character set

For Czech custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Danish character set

For Danish custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Dutch character set

For Dutch custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

English character set

For English custom vocabularies, you can use the following characters in the `Phrase` field:

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

Estonian character set

For Estonian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the Phrase field:

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

Farsi character set

For Farsi custom vocabularies, you can use the following characters in the Phrase field.

Character	Code	Character	Code
ء	0621	ظ	0638
آ	0622	ع	0639
أ	0623	غ	063A
ؤ	0624	ف	0641
ئ	0626	ق	0642
ا	0627	ل	0644
ب	0628	م	0645
ت	062A	ن	0646
ث	062B	ه	0647

Character	Code	Character	Code
ج	062C	و	0648
ح	062D	َ	064E
خ	062E	ُ	064F
د	062F	ِ	0650
ذ	0630	ّ	0651
ر	0631	پ	067E
ز	0632	چ	0686
س	0633	ژ	0698
ش	0634	ک	06A9
ص	0635	گ	06AF
ض	0636	ی	06CC
ط	0637		

Finnish character set

For Finnish custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
ä	00E4	ö	00F6

Character	Code	Character	Code
å	00E5	š	0161
ž	017E		

French character set

For French custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
À	00C0	à	00E0
Â	00C2	â	00E2
Ç	00C7	ç	00E7
È	00C8	è	00E8
É	00C9	é	00E9
Ê	00CA	ê	00EA
Ë	00CB	ë	00EB
Î	00CE	î	00EE
Ï	00CF	ï	00EF

Character	Code	Character	Code
Ô	00D4	ô	00F4
Ö	00D6	ö	00F6
Ù	00D9	ù	00F9
Û	00DB	û	00FB
Ü	00DC	ü	00FC

Galician character set

For Galician custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Georgian character set

For Georgian custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z

- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` field:

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

German character set

For German custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Greek character set

For Greek custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
ά	03AC	ν	03BD

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

Gujarati character set

For Gujarati custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z

- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` field:

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

Character	Code	Character	Code
ખ	0A96	઼	0ABC
ગ	0A97	ા	0ABE
ઘ	0A98	િ	0ABF
ઙ	0A99	ી	0AC0
ચ	0A9A	ુ	0AC1
છ	0A9B	ૂ	0AC2
જ	0A9C	ૃ	0AC3
ઝ	0A9D	ૅ	0AC5
ઞ	0A9E	ે	0AC7
ટ	0A9F	ૈ	0AC8
ઠ	0AA0	ૉ	0AC9
ડ	0AA1	ો	0ACB
ઢ	0AA2	ૌ	0ACC
ણ	0AA3	્	0ACD
ત	0AA4	ૐ	0AD0
થ	0AA5	ૠ	0AE0

Hausa character set

For Hausa custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z
- - (hyphen)

- . (period)

You can also use the following Unicode characters in the `Phrase` field:

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

Hebrew character set

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

Character	Code	Character	Code
-	002D	ם	05DD
א	05D0	מ	05DE
ב	05D1	ן	05DF
ג	05D2	נ	05E0
ד	05D3	ס	05E1
ה	05D4	ע	05E2
ו	05D5	ף	05E3
ז	05D6	פ	05E4
ח	05D7	ץ	05E5
ט	05D8	צ	05E6
י	05D9	ק	05E7

Character	Code	Character	Code
ך	05DA	ר	05E8
כ	05DB	ש	05E9
ל	05DC	ת	05EA

Hindi character set

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

Character	Code	Character	Code
-	002D	थ	0925
.	002E	द	0926
ँ	0901	ध	0927
ं	0902	न	0928
ः	0903	प	092A
अ	0905	फ	092B
आ	0906	ब	092C
इ	0907	भ	092D
ई	0908	म	092E
उ	0909	य	092F
ऊ	090A	र	0930
ऋ	090B	ल	0932
ए	090F	व	0935

Character	Code	Character	Code
ऐ	0910	श	0936
ऑ	0911	ष	0937
ओ	0913	स	0938
औ	0914	ह	0939
क	0915	ा	093E
ख	0916	ि	093F
ग	0917	ी	0940
घ	0918	ु	0941
ङ	0919	ू	0942
च	091A	ृ	0943
छ	091B	ॅ	0945
ज	091C	े	0947
झ	091D	ै	0948
ञ	091E	ॉ	0949
ट	091F	ो	094B
ठ	0920	ौ	094C
ड	0921	्	094D
ढ	0922	ज़	095B
ण	0923	ड़	095C
त	0924	ढ़	095D

Amazon Transcribe maps the following characters:

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

Hungarian character set

For Hungarian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Icelandic character set

For Icelandic custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Indonesian character set

For Indonesian custom vocabularies, you can use the following characters in the `Phrase` field:

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

Italian character set

For Italian custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z

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

You can also use the following Unicode characters in the `Phrase` field:

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

Japanese character set

For Japanese custom vocabularies, the `DisplayAs` field supports all hiragana, katakana, and kanji characters, and fullwidth romaji capital letters.

The `Phrase` field supports the characters listed in the following file:

- ja-jp-character-set

Kabyle character set

For Kabyle custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Kannada character set

For Kannada custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
ಂ	0C82	ಧ	0CA7

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

Character	Code	Character	Code
ಛ	0C9B	ು	0CC1
ಜ	0C9C	ೂ	0CC2
ಝ	0C9D	ೃ	0CC3
ಞ	0C9E	ೆ	0CC6
ಟ	0C9F	ೇ	0CC7
ಠ	0CA0	ೈ	0CC8
ಡ	0CA1	ೊ	0CCA
ಢ	0CA2	ೋ	0CCB
ಣ	0CA3	ೌ	0CCC
ತ	0CA4	್	0CCD
ಥ	0CA5	ೕ	0CD5
ದ	0CA6	ೖ	0CD6
ೠ	0CE0		

Kazakh character set

For Kazakh custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Character	Code	Character	Code
ъ	044A	ұ	04B1

Kinyarwanda character set

For Kinyarwanda custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Korean character set

For Korean custom vocabularies, you can use any of the Hangul syllables in the `Phrase` field. For more information, see Hangul Syllables on Wikipedia.

Kyrgyz character set

For Kyrgyz custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

Character	Code	Character	Code
ѐ	0450	ӡ	04E1
ё	0451	ӣ	04E3
ђ	0452	ӧ	04E7
ѓ	0453	ө	04E9
є	0454	ӯ	04EF
ѕ	0455	ӱ	04F1
і	0456	ӳ	04F3
ї	0457	ӷ	04F7
ј	0458	ӹ	04F9

Latvian character set

For Latvian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Character	Code	Character	Code
ī	012B	ū	016B
ž	017E		

Lithuanian character set

For Lithuanian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Luganda character set

For Luganda custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the Phrase field:

Character	Code	Character	Code
ÿ	00FF	ŋ	014B

Macedonian character set

For Macedonian custom vocabularies, you can use the following characters in the Phrase field:

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

You can also use the following Unicode characters in the Phrase field:

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

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

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

Malay character set

For Malay custom vocabularies, you can use the following characters in the `Phrase` field:

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

Malayalam character set

For Malayalam custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z
- - (hyphen)

- . (period)

You can also use the following Unicode characters in the `Phrase` field:

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

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

Maltese character set

For Maltese custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z
- - (hyphen)

- . (period)

You can also use the following Unicode characters in the `Phrase` field:

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

Marathi character set

For Marathi custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

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

Meadow Mari character set

For Meadow Mari custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

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

Mongolian character set

For Mongolian custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z
- - (hyphen)

- . (period)

You can also use the following Unicode characters in the `Phrase` field:

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

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

Character	Code	Character	Code
і	0456	ӳ	04F3
ї	0457	ӷ	04F7
ј	0458	ӹ	04F9

Norwegian Bokmål character set

For Norwegian Bokmål custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
å	00E5	æ	00E6
ø	00F8		

Odia/Oriya character set

For Odia/Oriya custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

Pashto character set

For Pashto custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
آ	0622	و	0648

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

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

Polish character set

For Polish custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Portuguese character set

For Portuguese custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

Punjabi character set

For Punjabi custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

Character	Code	Character	Code
ੳ	0A73		

Romanian character set

For Romanian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
ă	0103	ș	0219
â	00E2	ț	021B
î	00EE	ş	015F
ţ	0163		

Russian character set

For Russian custom vocabularies, you can use the following characters in the `Phrase` field:

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

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

Serbian character set

For Serbian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

Character	Code	Character	Code
ѓ	0453	ӱ	04F1
є	0454	ӳ	04F3
ѕ	0455	ӷ	04F7
ӹ	04F9		

Sinhala character set

For Sinhala custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

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

Slovak character set

For Slovak custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Character	Code	Character	Code
í	00ED	ť	0165
ĺ	013A	ú	00FA
ľ	013E	ý	00FD
ž	017E		

Slovenian character set

For Slovenian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Somali character set

For Somali custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Spanish character set

For Spanish custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Sundanese character set

For Sundanese custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Swahili character set

For Swahili custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
s	0073	d	0064
t	0074	a	0061
a	0061	r	0072

Character	Code	Character	Code
n	006E	d	0064

Swedish character set

For Swedish custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code	Character	Code
Ä	00C4	ä	00E4
Å	00C5	å	00E5
Ö	00D6	ö	00F6

Tagalog/Filipino character set

For Tagalog/Filipino custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

Character	Code		
ñ	00F1		

Tamil character set

For Tamil custom vocabularies, you can use the following characters in the `Phrase` field:

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

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

Tatar character set

For Tatar custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

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

Telugu character set

For Telugu custom vocabularies, you can use the following characters in the `Phrase` field:

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

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

Thai character set

For Thai custom vocabularies, you can use the following characters in the `Phrase` field:

- - (hyphen)
- . (period)

You can also use the following Unicode characters in the `Phrase` field:

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

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

Turkish character set

For Turkish custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z
- A - Z

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

You can also use the following Unicode characters in the `Phrase` field:

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

Ukrainian character set

For Ukrainian custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Uyghur character set

For Uyghur custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

Character	Code	Character	Code
ه	0647	ړ	0693
ڕ	0695		

Uzbek character set

For Uzbek custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

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

Vietnamese character set

Amazon Transcribe represents the six tones in Vietnamese using numbers. The following table shows how tone marks are mapped for the word 'ma'.

Tone name	Tone mark	Tone number
ngang	ma	ma1
sắc	má	ma2
huyền	mà	ma3
hỏi	mả	ma4
ngã	mã	ma5
nặng	mạ	ma6

For Vietnamese custom vocabularies, you can use the following characters in the `Phrase` field:

- a - z
- A - Z

- ' (apostrophe)
- - (hyphen)
- . (period)
- & (ampersand)
- ; (semicolon)
- _ (low line)

You can also use the following Unicode characters in the `Phrase` field:

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

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

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

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

Welsh character set

For Welsh custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Wolof character set

For Wolof custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

Zulu character set

For Zulu custom vocabularies, you can use the following characters in the `Phrase` field:

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

You can also use the following Unicode characters in the `Phrase` field:

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

How Amazon Transcribe works

Amazon Transcribe uses machine learning models to convert speech to text.

In addition to the transcribed text, transcripts contains data about the transcribed content, including confidence scores and timestamps for each word or punctuation mark. To see an output example, refer to the Data input and output section. For a complete list of features that you can apply to your transcription, refer to the feature summary.

Transcription methods can be separated into two main categories:

- **Batch transcriptions**: Transcribe media files that have been uploaded into an Amazon S3 bucket. You can use the Amazon CLI, Amazon Web Services Management Console, and various Amazon SDKs for batch transcriptions.
- **Streaming transcriptions**: Transcribe media streams in real time. You can use the Amazon Web Services Management Console, HTTP/2, WebSockets, and various Amazon SDKs for streaming transcriptions.

Note that feature and language support differs for batch and streaming transcriptions. For more information, refer to ??? and Supported languages.

Topics

API operations to get you started

Batch: `StartTranscriptionJob`
Streaming: `StartStreamTranscription`, StartStreamTranscriptionWebSocket

Data input and output

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

If you're transcribing media files stored in an Amazon S3 bucket, you're performing **batch transcriptions**. If you're transcribing media streams, you're performing **streaming transcriptions**. These two processes have different rules and requirements.

With batch transcriptions, you can use Job queueing if you don't need to process all of your transcription jobs concurrently. This allows Amazon Transcribe to keep track of your transcription jobs and process them when slots are available.

Note

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

Topics

Media formats

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

	Batch	Streaming
Supported formats	• AMR • FLAC • M4A • MP3 • MP4 • Ogg	• FLAC • Ogg Opus • PCM encoding

	Batch	Streaming
	• WebM • WAV	
Recommended formats	• FLAC • WAV with PCM 16-bit encoding	• FLAC • PCM signed 16-bit little-endian audio (note that this does **not** include WAV)

For best results, use a lossless format, such as FLAC or WAV with PCM 16-bit encoding.

Note

Streaming transcriptions are not supported with all languages. Refer to the 'Data input' column in the supported languages table for details.

Audio channels

Amazon Transcribe supports single-channel and dual-channel media. Media with more than two channels is not currently supported.

If your audio contains multiple speakers on one channel and you want to partition and label each speaker in your transcription output, you can use Speaker partitioning (diarization).

If your audio contains speech on two separate channels, you can use Channel identification to transcribe each channel separately within your transcript.

Both of these options produce one transcript file.

Note

If you don't enable Speaker partitioning or Channel identification, your transcript text is provided as one continuous section.

Sample rates

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

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

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

Output

Transcription output is in JSON format. The first part of your transcript contains the transcript itself in paragraph form, followed by additional data for every word and punctuation mark. The data provided depends on the features you include in your request. At a minimum, your transcript contains the start time, end time, and confidence score for every word. The following section shows example output from a basic transcription request that didn't include any additional options or features.

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

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

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

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

Streaming transcripts are returned via the same method you're using for your stream.

Tip

If you want to convert your JSON output into a turn-by-turn transcript in Word format, see this GitHub example (for Python3). This script works with post-call analytics transcripts and standard batch transcripts with diarization enabled.

Example output

Transcripts provide a complete transcription in paragraph form, followed by a word-for-word breakdown, which provides data for every word and punctuation mark. This includes start time, end time, a confidence score, and a type (`pronunciation` or `punctuation`).

The following example is from a simple batch transcription job that didn't include any additional features. With each additional feature that you apply to your transcription request, you get additional data in your transcript output file.

Basic batch transcripts contain two main sections:

1. `transcripts`: Contains the entire transcript in one text block.
2. `items`: Contains information on each word and punctuation mark from the `transcripts` section.
3. `audio_segments`: An audio segment is a specific portion of an audio recording that contains uninterrupted spoken language, with minimal pauses or breaks. This segment captures a natural flow of speech and is captured in `audio_segments` with a start time and end time. The `items` element within an audio segment is a sequence of identifiers that correspond to each item within the segment.

Each additional feature you include in your transcription request produces additional information in your transcript.

```
{
    "jobName": "my-first-transcription-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "Welcome to Amazon Transcribe."
            }
        ],
        "items": [
            {
                "id": 0,
                "start_time": "0.64",
                "end_time": "1.09",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Welcome"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 1,
                "start_time": "1.09",
                "end_time": "1.21",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "to"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 2,
                "start_time": "1.21",
                "end_time": "1.74",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Amazon"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 3,
                "start_time": "1.74",
                "end_time": "2.56",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Transcribe"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 4,
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ],
        "audio_segments": [
            {
                "id": 0,
                "transcript": "Welcome to Amazon Transcribe.",
                "start_time": "0.64",
                "end_time": "2.56",
                "items": [
                    0,
                    1,
                    2,
                    3,
                    4
                ]
            }
        ]
    },
    "status": "COMPLETED"
}
```

Transcribing numbers and punctuation

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

For most languages, numbers are transcribed into their word forms. However, for languages with support for transcribing numbers, Amazon Transcribe treats numbers differently depending on the context in which they're used.

For example, if a speaker says "*Meet me at eight-thirty AM on June first at one-hundred Main Street with three-dollars-and-fifty-cents and one-point-five chocolate bars*," this is transcribed as:

- Languages with number transcription support: Meet me at 8:30 a.m. on June 1st at 100 Main Street with $3.50 and 1.5 chocolate bars
- All other languages: Meet me at eight thirty a m on June first at one hundred Main Street with three dollars and fifty cents and one point five chocolate bars

To view the languages with support for transcribing numbers, refer to Supported languages and language-specific features.

Getting started with Amazon Transcribe

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

- Sign up for an Amazon Web Services account
- Install the Amazon CLI and SDKs (if you're using the Amazon Web Services Management Console for your transcriptions, you can skip this step)
- Configure IAM credentials
- Set up an Amazon S3 bucket
- Create an IAM policy

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

- Amazon CLI
- Amazon Web Services Management Console
- Amazon SDK
- HTTP
- WebSockets

Tip

If you're new to Amazon Transcribe or would like to explore our features, we recommend using the Amazon Web Services Management Console. This is also the easiest option if you'd like to start a stream using your computer microphone.

Because streaming using HTTP/2 and WebSockets is more complicated than the other transcription methods, we recommend reviewing the Setting up a streaming transcription section before getting started with these methods. **Note that we strongly recommend using an SDK for streaming transcriptions.**

Signing up for an Amazon Web Services account

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

Tip

When setting up your account, make note of your Amazon Web Services account ID because you need it to create IAM entities.

Installing the Amazon CLI and SDKs

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

Once you have the Amazon CLI installed, you must configure it for your security credentials and Amazon Web Services Region.

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

- .NET
- C++
- Go
- Java V2
- JavaScript
- PHP V3
- Amazon SDK for Python (Boto3) (batch transcriptions)
- Python (streaming transcriptions)
- Ruby V3
- Rust (batch transcriptions)

- Rust (streaming transcriptions)

Configure IAM credentials

When you create an Amazon Web Services account, you begin with one sign-in identity that has complete access to all Amazon services and resources in your account. This identity is called the Amazon Web Services 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. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform.

As a best practice, require users—including those that require administrator access—to use federation with an identity provider to access Amazon services by using temporary credentials.

A federated identity is any user who accesses Amazon services by using credentials provided through an identity source. When federated identities access Amazon Web Services accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use Amazon IAM Identity Center. You can create users and groups in IAM Identity Center. Or you can connect and synchronize to a set of users and groups in your own identity source for use across all your Amazon Web Services accounts and applications. For more information, see Identity and Access Management for Amazon Transcribe.

To learn more about IAM best practices, refer to Security best practices in IAM.

Creating an Amazon S3 bucket

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

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

Learn how to Create your first S3 bucket and Upload an object to your bucket.

Creating an IAM policy

To manage access in Amazon, you must create policies and attach them to IAM identities (users, groups, or roles) or Amazon resources. A policy defines the permissions of the entity it is attached to. For example, a role can only access a media file located in your Amazon S3 bucket if you've attached a policy to that role which grants it access. If you want to further restrict that role, you can instead limit its access to a specific file within an Amazon S3 bucket.

To learn more about using Amazon policies see:

- Policies and permissions in IAM
- Creating IAM policies
- How Amazon Transcribe works with IAM

For example policies you can use with Amazon Transcribe, see Amazon Transcribe identity-based policy examples. If you want to generate custom policies, consider using the Amazon Policy Generator.

You can add a policy using the Amazon Web Services Management Console, Amazon CLI, or Amazon SDK. For instructions, see Adding and removing IAM identity permissions.

Policies have the format:

Amazon Resource Names (ARNs) uniquely identify all Amazon resources, such as an Amazon S3 bucket. You can use ARNs in your policy to grant permissions for specific actions to use specific resources. For example, if you want to grant read access to an Amazon S3 bucket and its sub-folders, you can add the following code to your trust policy's `Statement` section:

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

Here's an example policy that grants Amazon Transcribe read (`GetObject`, `ListBucket`) and write (`PutObject`) permissions to an Amazon S3 bucket, `amzn-s3-demo-bucket`, and its sub-folders:

JSON

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

Transcribing with the Amazon Web Services Management Console

You can use the Amazon console for batch and streaming transcriptions. If you're transcribing a media file located in an Amazon S3 bucket, you're performing a batch transcription. If you're transcribing a real-time stream of audio data, you're performing a streaming transcription.

Before starting a batch transcription, you must first upload your media file to an Amazon S3 bucket. For streaming transcriptions using the Amazon Web Services Management Console, you must use your computer microphone.

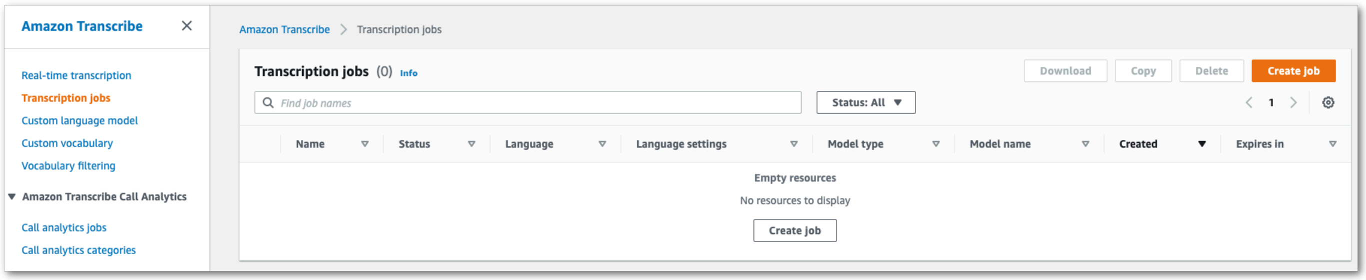
To view supported media formats and other media requirements and constraints, see Data input and output.

Expand the following sections for short walkthroughs of each transcription method.

Batch transcriptions

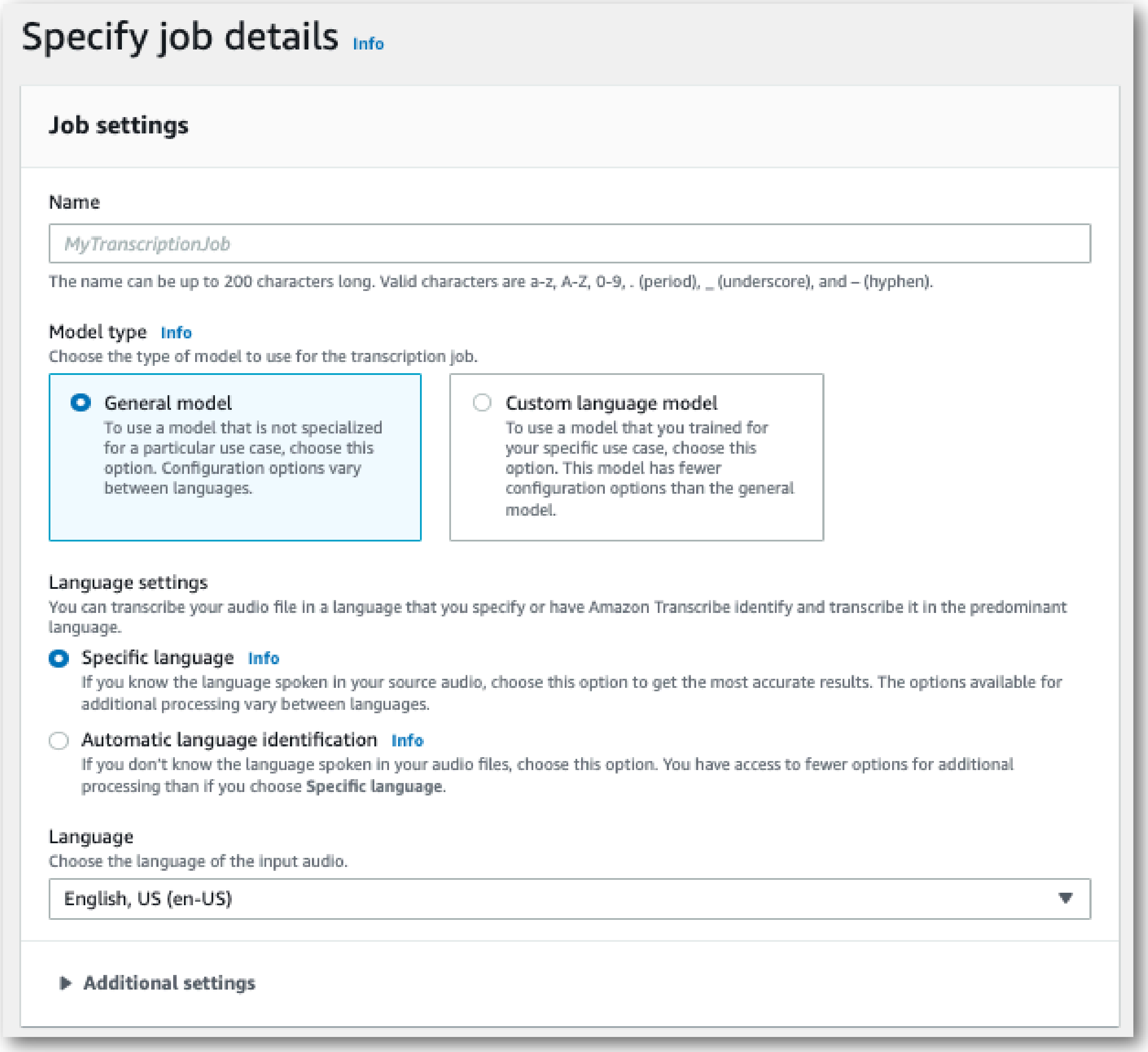
First make sure that you've uploaded the media file you want to transcribe into an Amazon S3 bucket. If you're unsure how to do this, refer to the *Amazon S3 User Guide*: Upload an object to your bucket.

1. From the Amazon Web Services Management Console, select **Transcription jobs** in the left navigation pane. This takes you to a list of your transcription jobs.



   Select **Create job**.

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


Specify job details Info

Job settings

Name

MyTranscriptionJob

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

Model type Info

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

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

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

Language settings

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

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

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

Language

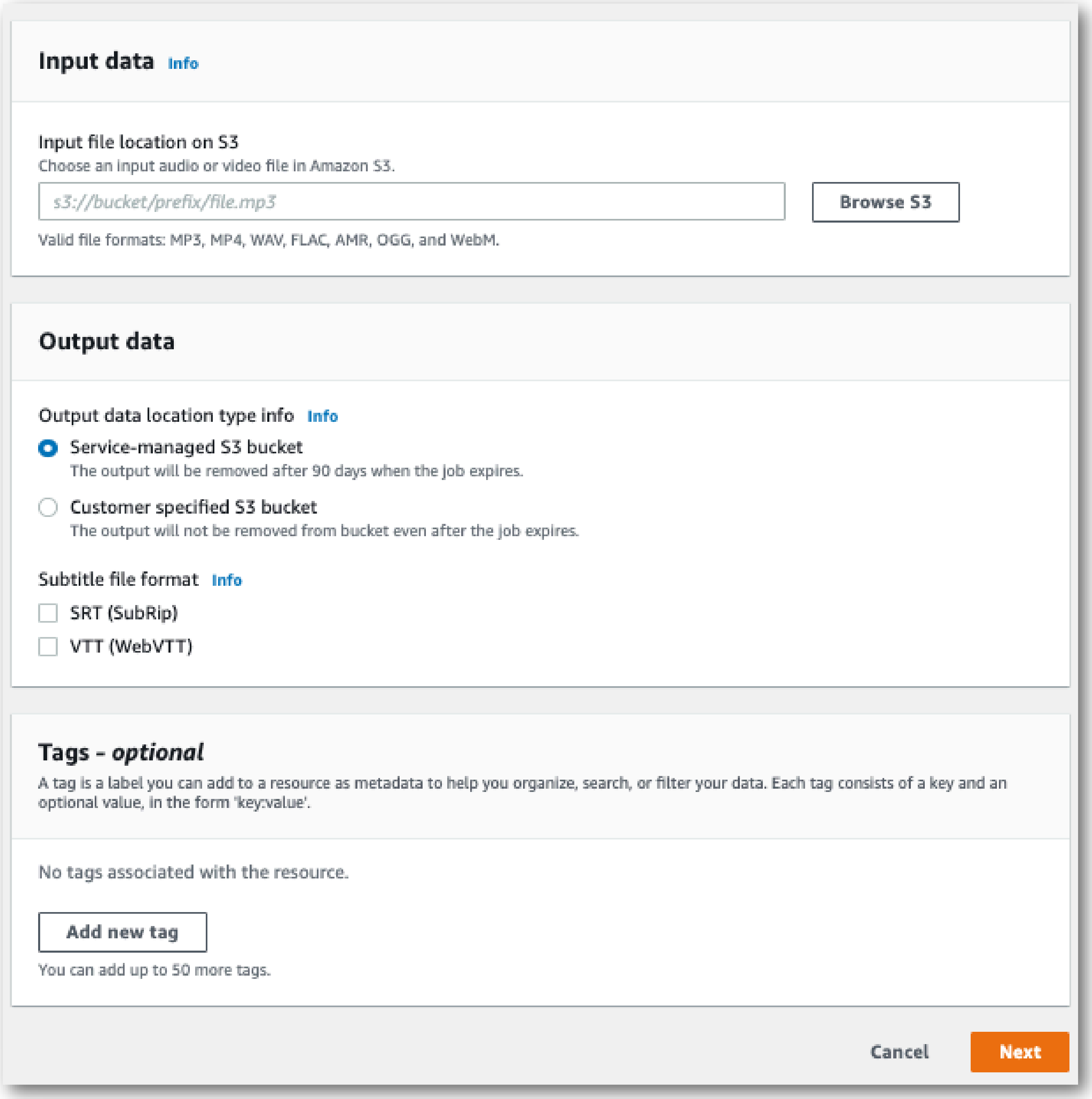
Choose the language of the input audio.

English, US (en-US)

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

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

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



Input data Info

Input file location on S3

Choose an input audio or video file in Amazon S3.

s3://bucket/prefix/file.mp3

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

Output data

Output data location type info Info

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

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

Subtitle file format Info

- SRT (SubRip)
- VTT (WebVTT)

Tags - *optional*

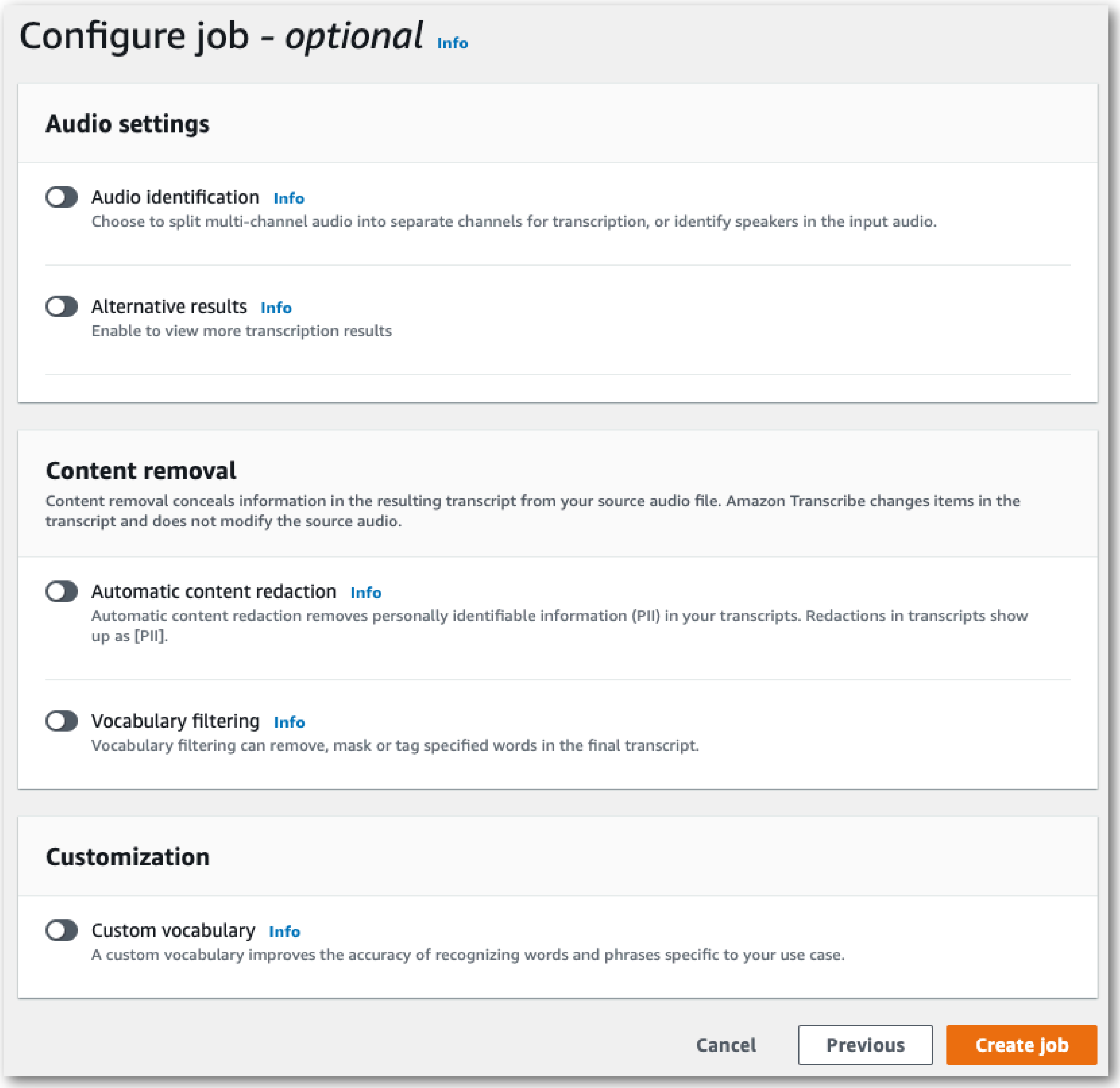
A tag is a label you can add to a resource as metadata to help you organize, search, or filter your data. Each tag consists of a key and an optional value, in the form 'key:value'.

No tags associated with the resource.

You can add up to 50 more tags.

Select **Next**.

3. Select any desired options on the **Configure job** page. If you want to use Custom vocabularies with your transcription, you must create these before starting your transcription job.



Configure job - *optional* Info

Audio settings

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

Alternative results Info
Enable to view more transcription results

Content removal
Content removal conceals information in the resulting transcript from your source audio file. Amazon Transcribe changes items in the transcript and does not modify the source audio.

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

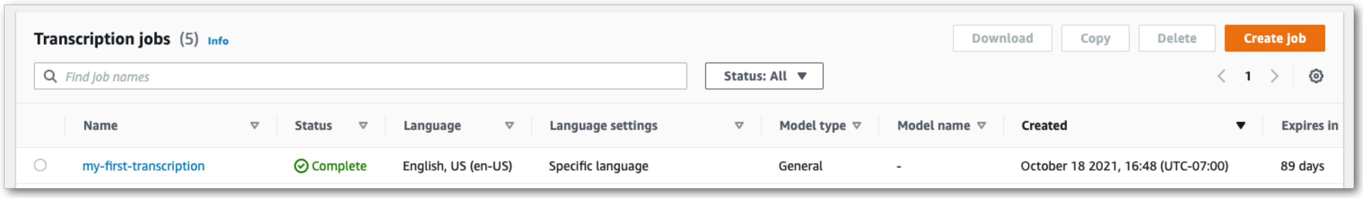
Vocabulary filtering Info
Vocabulary filtering can remove, mask or tag specified words in the final transcript.

Customization

Custom vocabulary Info
A custom vocabulary improves the accuracy of recognizing words and phrases specific to your use case.

Select **Create job**.

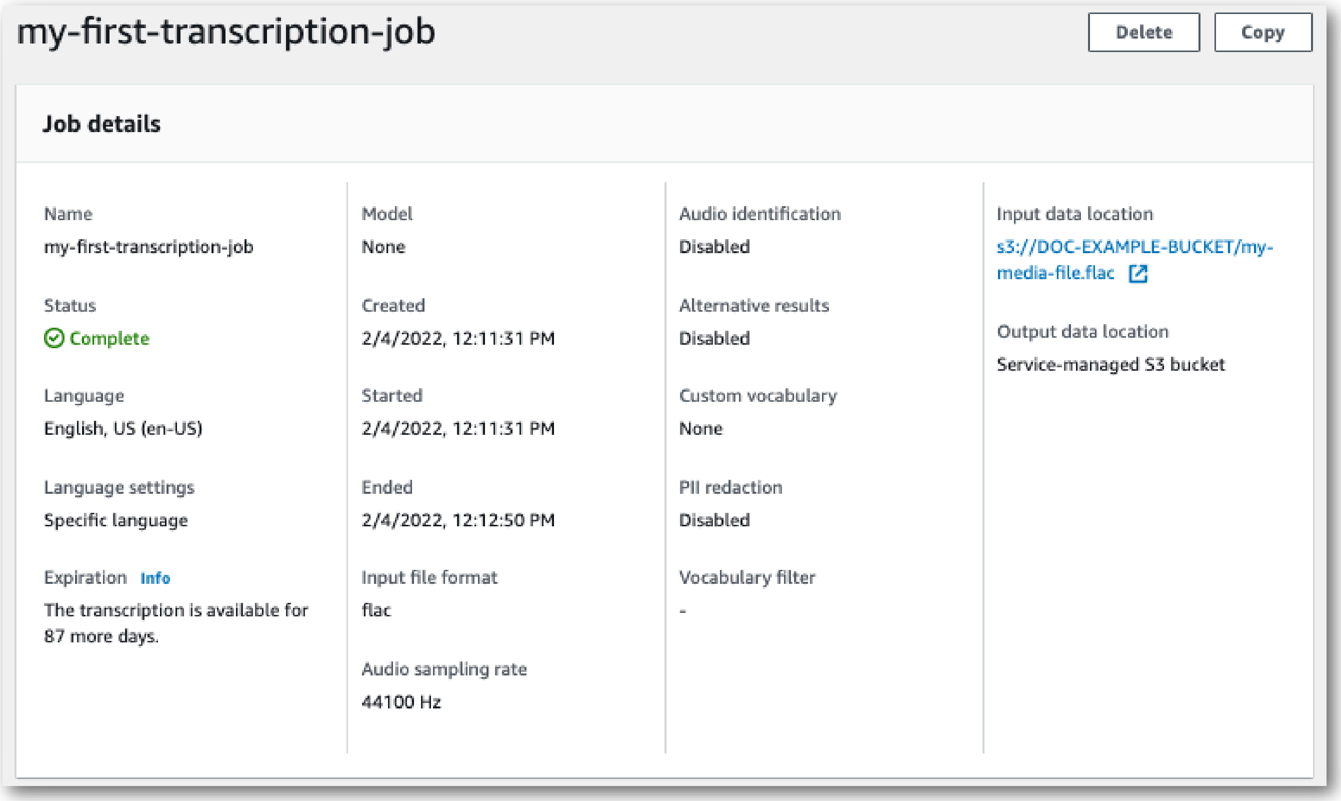
4. You're now on the **Transcription jobs** page. Here you can see the status of the transcription job. Once complete, select your transcription.



Name	Status	Language	Language settings	Model type	Model name	Created	Expires in
my-first-transcription	Complete	English, US (en-US)	Specific language	General	-	October 18 2021, 16:48 (UTC-07:00)	89 days

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

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



Job details

Name
my-first-transcription-job

Status
Complete

Language
English, US (en-US)

Language settings
Specific language

Expiration Info
The transcription is available for 87 more days.

Model
None

Created
2/4/2022, 12:11:31 PM

Started
2/4/2022, 12:11:31 PM

Ended
2/4/2022, 12:12:50 PM

Input file format
flac

Audio sampling rate
44100 Hz

Audio identification
Disabled

Alternative results
Disabled

Custom vocabulary
None

PII redaction
Disabled

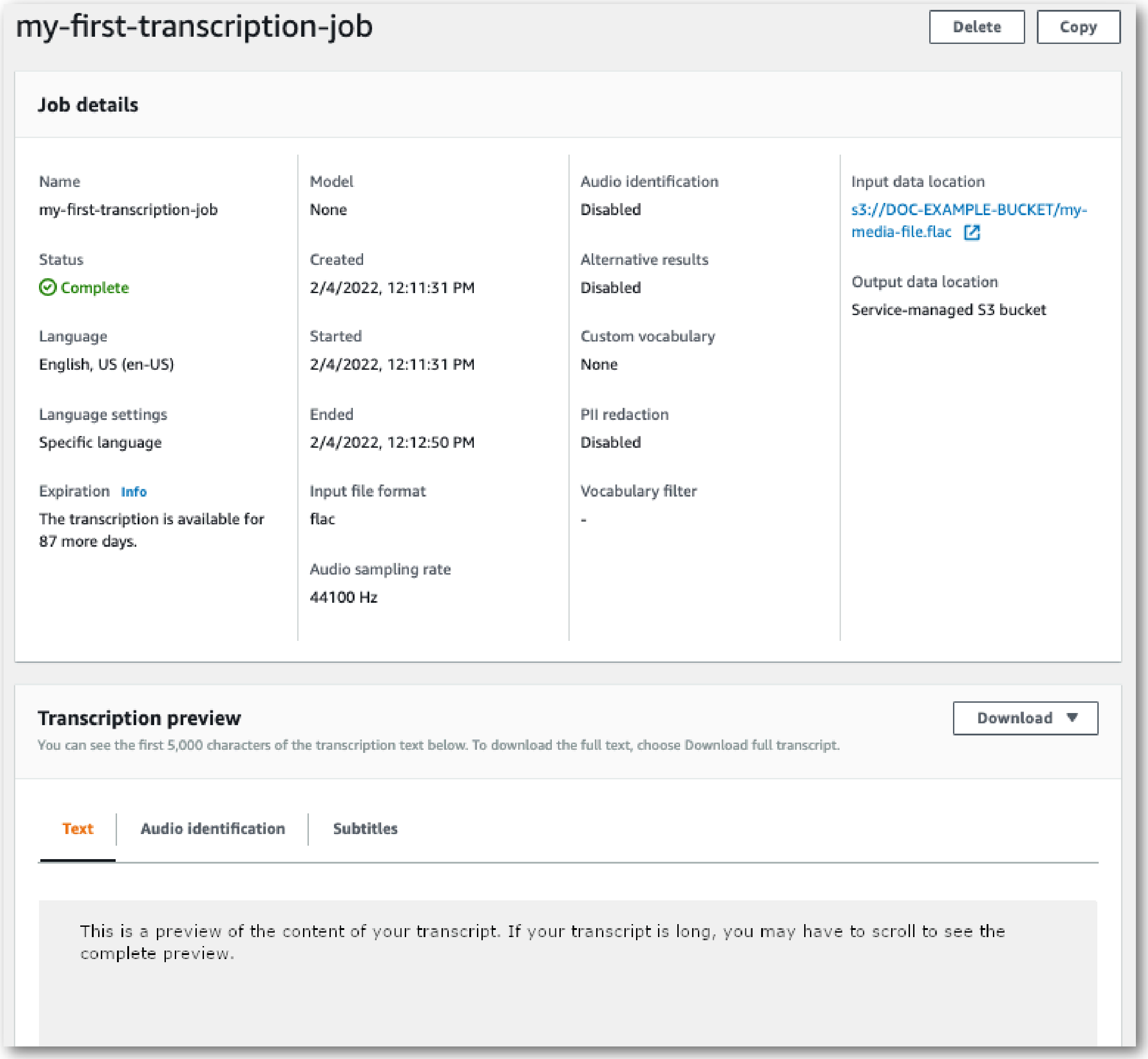
Vocabulary filter
-

Input data location
s3://DOC-EXAMPLE-BUCKET/my-media-file.flac

Output data location
Service-managed S3 bucket

6. How you download your transcript depends on whether you chose a service-managed Amazon S3 bucket or your own Amazon S3 bucket.

   a. If you chose a service-managed bucket, you can see a **Transcription preview** pane on your transcription job's information page, along with a **Download** button.



Job details

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

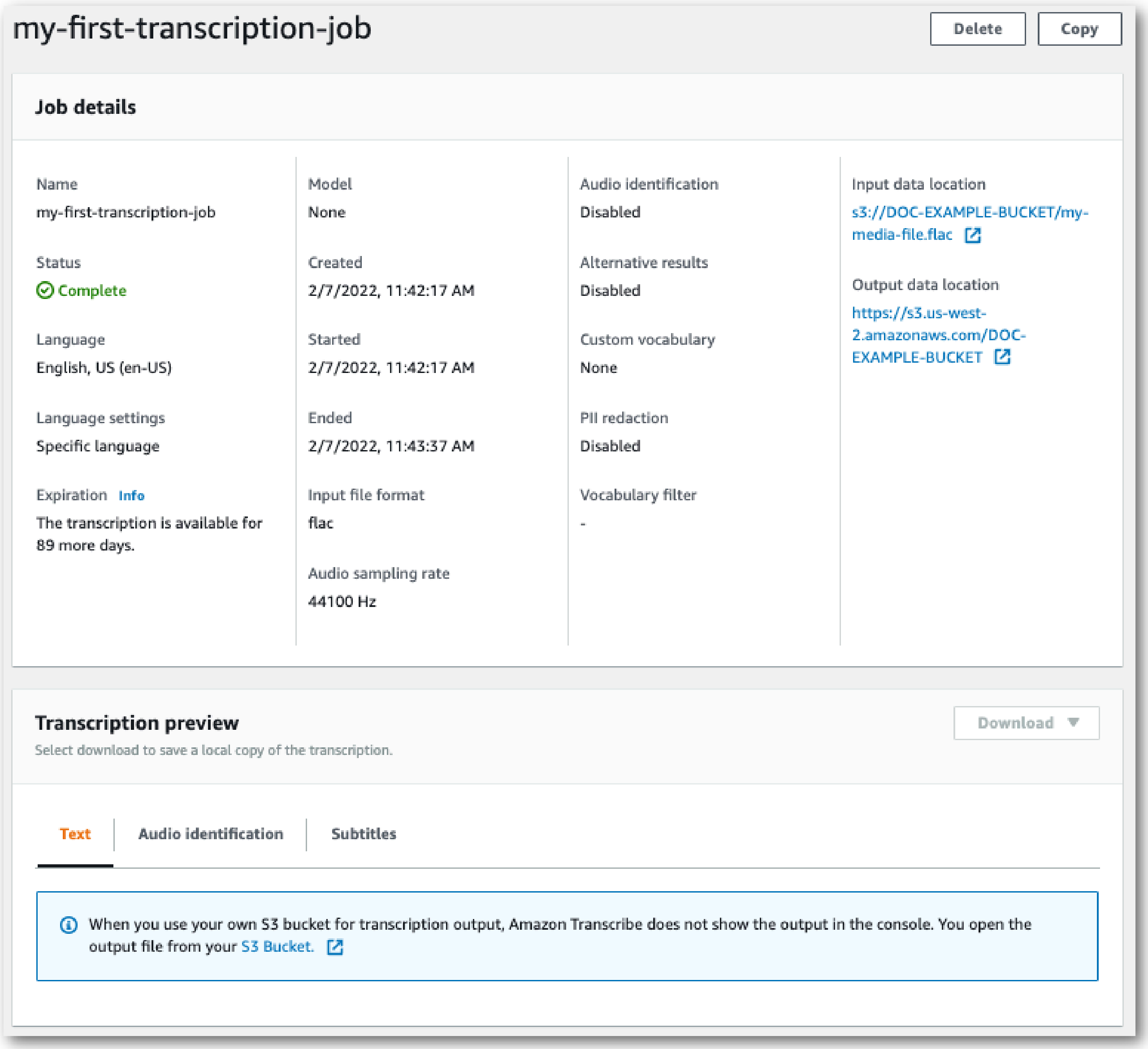
Transcription preview

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

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

Select **Download** and choose **Download transcript**.

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



my-first-transcription-job

Job details

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

Transcription preview

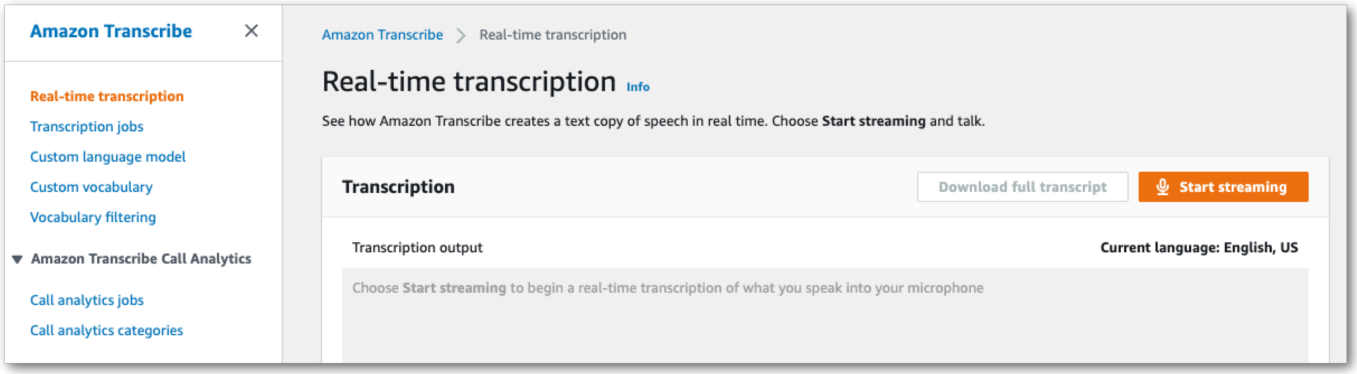
Select download to save a local copy of the transcription.

When you use your own S3 bucket for transcription output, Amazon Transcribe does not show the output in the console. You open the output file from your S3 Bucket.

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

Streaming transcriptions

1. From the Amazon Web Services Management Console, select **Real-time transcription** in the left navigation pane. This takes you to the main streaming page where you can select options before starting your stream.



Real-time transcription

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

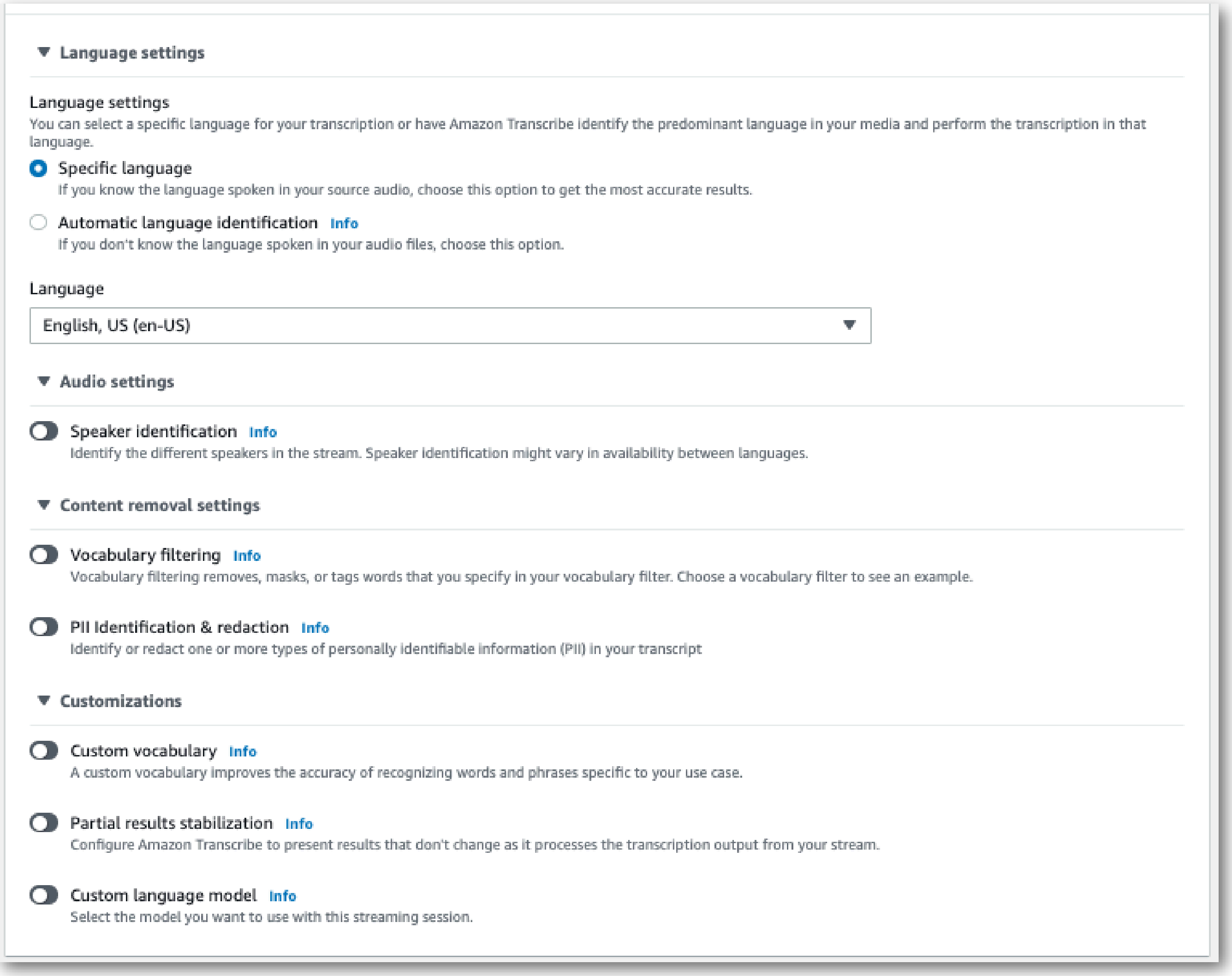
Transcription

Transcription output

Current language: English, US

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

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



Language settings

Language settings

You can select a specific language for your transcription or have Amazon Transcribe identify the predominant language in your media and perform the transcription in that language.

[✓ Specific language]
If you know the language spoken in your source audio, choose this option to get the most accurate results.

Automatic language identification Info
If you don't know the language spoken in your audio files, choose this option.

Language: English, US (en-US)

Audio settings

Speaker identification Info
Identify the different speakers in the stream. Speaker identification might vary in availability between languages.

Content removal settings

Vocabulary filtering Info
Vocabulary filtering removes, masks, or tags words that you specify in your vocabulary filter. Choose a vocabulary filter to see an example.

PII Identification & redaction Info
Identify or redact one or more types of personally identifiable information (PII) in your transcript

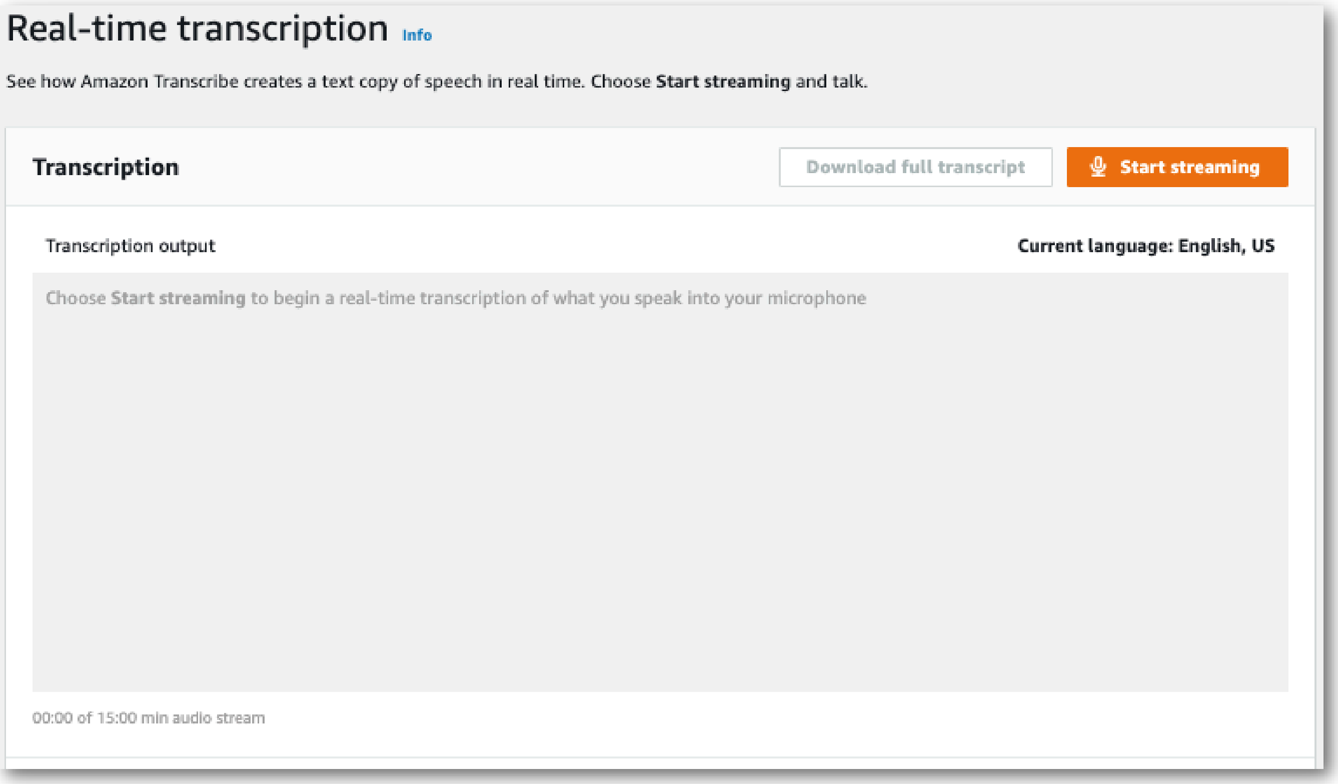
Customizations

Custom vocabulary Info
A custom vocabulary improves the accuracy of recognizing words and phrases specific to your use case.

Partial results stabilization Info
Configure Amazon Transcribe to present results that don't change as it processes the transcription output from your stream.

Custom language model Info
Select the model you want to use with this streaming session.

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



Real-time transcription Info

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

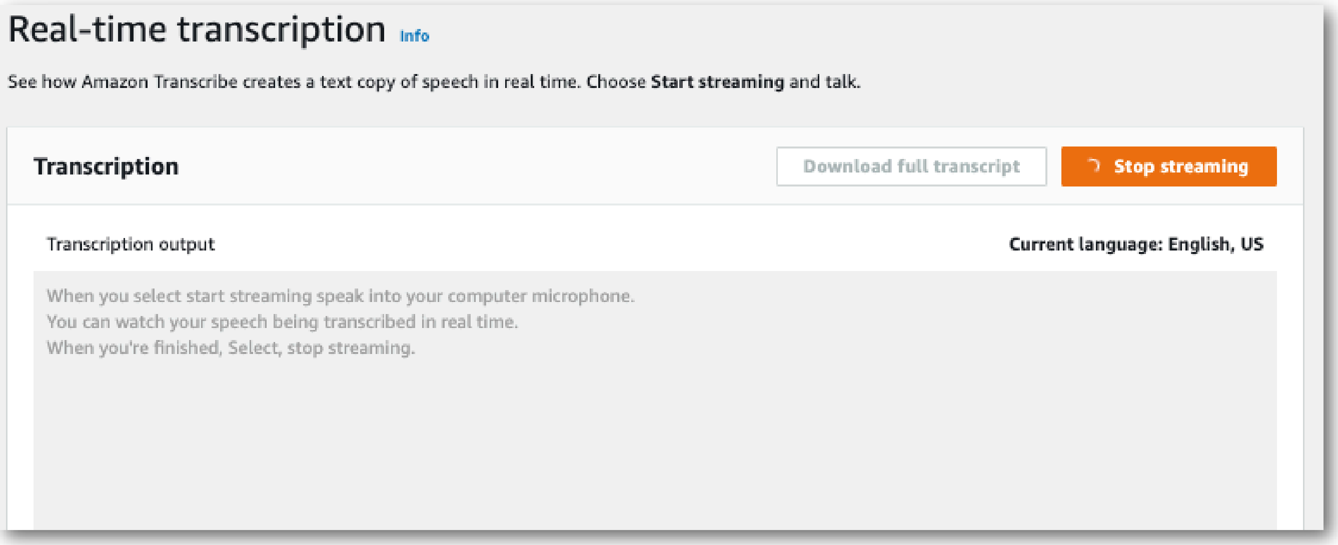
Transcription output

Current language: English, US

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

00:00 of 15:00 min audio stream

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



Real-time transcription Info

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

Transcription output

Current language: English, US

When you select start streaming speak into your computer microphone.
You can watch your speech being transcribed in real time.
When you're finished, Select, stop streaming.

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

Transcribing with the Amazon CLI

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

The Amazon CLI does not support streaming transcriptions.

Before you continue, make sure you've:

- Uploaded your media file into an Amazon S3 bucket. If you're unsure how to create an Amazon S3 bucket or upload your file, refer to Create your first Amazon S3 bucket and Upload an object to your bucket.
- Installed the Amazon CLI.

You can find all Amazon CLI commands for Amazon Transcribe in the Amazon CLI Command Reference.

Starting a new transcription job

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

1. In a terminal window, type the following:

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

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

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

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

   If you want to specify an output location, include `output-bucket-name` in your request; if you want to specify a sub-folder of the specified output bucket, also include `output-key`.

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

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

If appending all parameters, this request looks like:

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

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

Amazon Transcribe responds with:

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

Your transcription job is successful if `TranscriptionJobStatus` changes from `IN_PROGRESS` to `COMPLETED`. To see the updated `TranscriptionJobStatus`, use the `get-transcription-job` or `list-transcription-job` command, as shown in the following section.

Getting the status of a transcription job

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

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

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

Amazon Transcribe responds with:

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

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

Note

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

Listing your transcription jobs

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

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

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

Amazon Transcribe responds with:

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

Deleting your transcription job

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

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

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

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

Transcribing with the Amazon SDKs

You can use SDKs for both batch and streaming transcriptions. If you're transcribing a media file located in an Amazon S3 bucket, you're performing a batch transcription. If you're transcribing a real-time stream of audio data, you're performing a streaming transcription.

For a list of the programming languages you can use with Amazon Transcribe, see Supported programming languages. Note that streaming transcriptions are not supported with all Amazon SDKs. To view supported media formats and other media requirements and constraints, see Data input and output.

For more information on all available Amazon SDKs and builder tools, refer to Tools to Build on Amazon.

Tip

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.
You can also find SDK code samples in these GitHub repositories:

- Amazon Code Examples
- Amazon Transcribe Examples

Batch transcriptions

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

Java

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

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

    public static void main(String args[]) {

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

        String transcriptionJobName = "my-first-transcription-job";
        String mediaType = "flac"; // can be other types
        Media myMedia = Media.builder()
                .mediaFileUri("s3://amzn-s3-demo-bucket/my-input-files/my-media-
file.flac")
                .build();

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

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

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

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

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

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

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

}
```

JavaScript

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

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

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

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

startTranscriptionRequest();
```

Python

```
import time
import boto3

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

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


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


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

Streaming transcriptions

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

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

Java

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

To run this example, note the following:

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

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

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

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

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

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

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

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16,000 Hz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

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

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

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

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

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

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

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

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;
        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

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

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

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

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

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

            demand.getAndAdd(n);

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

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

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

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

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

            return audioBuffer;
        }

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

JavaScript

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

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

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

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

Python

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

To run this example, note the following:

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

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

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

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


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

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

    async def write_chunks():
        # NOTE: For pre-recorded files longer than 5 minutes, the sent audio
        # chunks should be rate limited to match the real-time bitrate of the
        # audio stream to avoid signing issues.
        async with aiofile.AIOFile('filepath/my-media-file.flac', 'rb') as afp:
            reader = aiofile.Reader(afp, chunk_size = 1024 * 16)
            async for chunk in reader:
                await stream.input_stream.send_audio_event(audio_chunk = chunk)
        await stream.input_stream.end_stream()

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

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

C++

Refer to the Code examples chapter for the streaming C++ SDK example .

Using this service with an Amazon SDK

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

SDK documentation
Amazon CLI
Amazon SDK for Java
Amazon SDK for JavaScript
Amazon SDK for .NET
Amazon SDK for PHP
Amazon Tools for PowerShell
Amazon SDK for Python (Boto3)
Amazon SDK for Ruby
Amazon SDK for SAP ABAP

For examples specific to this service, see Code examples for Amazon Transcribe using Amazon SDKs.

Transcribing with HTTP or WebSockets

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

If you're transcribing a media file located in an Amazon S3 bucket, you're performing a batch transcription. If you're transcribing a real-time stream of audio data, you're performing a streaming transcription.

Both HTTP and WebSockets require you to authenticate your request using Amazon Signature Version 4 headers. Refer to Signing Amazon API requests for more information.

Batch transcriptions

You can make a batch HTTP request using the following headers:

- host

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

Here's an example of a `StartTranscriptionJob` request:

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

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

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

Streaming transcriptions

Streaming transcriptions using HTTP/2 and WebSockets are more involved than using SDKs. We recommend reviewing the Setting up a streaming transcription section before setting up your first stream.

For more information on these methods, refer to Setting up an HTTP/2 stream or Setting up a WebSocket stream.

Note

We strongly recommend using an SDK for streaming transcriptions. For a list of supported SDKs, refer to Supported programming languages.

Transcribing streaming audio

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

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

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

Amazon Transcribe offers the following options for streaming:

- SDKs (preferred)
- HTTP/2
- WebSockets
- Amazon Web Services Management Console

To transcribe streaming audio in the Amazon Web Services Management Console, speak into your computer microphone.

Tip

For SDK code examples, refer to the Amazon Samples repository on GitHub.

Audio formats supported for streaming transcriptions are:

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

Lossless formats (FLAC or PCM) are recommended.

Note

Streaming transcriptions are not supported with all languages. Refer to the 'Data input' column in the supported languages table for details.

To view the Amazon Transcribe Region availability for streaming transcriptions, see: Amazon Transcribe Endpoints and Quotas.

Best practices

The following recommendations improve streaming transcription efficiency:

- If possible, use PCM-encoded audio.
- Ensure that your stream is as close to real-time as possible.
- Latency depends on the size of your audio chunks. If you're able to specify chunk size with your audio type (such as with PCM), set each chunk to between 50 ms and 200 ms. You can calculate the audio chunk size by the following formula:

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

- Use a uniform chunk size.
- Make sure you correctly specify the number of audio channels.
- With single-channel PCM audio, each sample consists of two bytes, so each chunk should consist of an even number of bytes.
- With dual-channel PCM audio, each sample consists of four bytes, so each chunk should be a multiple of 4 bytes.
- When your audio stream contains no speech, encode and send the same amount of silence. For example, silence for PCM is a stream of zero bytes.
- Make sure you specify the correct sampling rate for your audio. If possible, record at a sampling rate of 16,000 Hz; this provides the best compromise between quality and data volume sent over the network. Note that most high-end microphones record at 44,100 Hz or 48,000 Hz.

Streaming and partial results

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

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

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

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

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

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

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

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


"IsPartial": false (complete segment)
```

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

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

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

Tip

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

If you're running an application that requires low latency, you may want to use partial-result stabilization.

Partial-result stabilization

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

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

This process gives you two options for each speech segment:

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

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

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

"Transcript":	**"EndTime":**	**"IsPartial":**
Partial-result stabilization not enabled		
`The`	`0.545`	`true`
`The`	`1.045`	`true`
`The Amazon.`	`1.545`	`true`
`The Amazon is`	`2.045`	`true`
`The Amazon is the law.`	`2.545`	`true`
`The Amazon is the largest`	`3.045`	`true`
`The Amazon is the largest ray`	`3.545`	`true`
`The Amazon is the largest rain for`	`4.045`	`true`
`The Amazon is the largest rainforest.`	`4.545`	`true`
`The Amazon is the largest rainforest on the`	`5.045`	`true`
`The Amazon is the largest rainforest on the planet.`	`5.545`	`true`
`The Amazon is the largest rainforest on the planet.`	`6.025`	`true`
`The Amazon is the largest rainforest on the planet.`	`6.025`	`false`
Partial-result stabilization enabled (high stability)		
`The`	`0.515`	`true`

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

When you activate partial-result stabilization, Amazon Transcribe uses a `Stable` field to indicate whether an item is stable, where 'item' refers to a transcribed word or punctuation mark. Values for `Stable` are `true` or `false`. Items flagged as `false` (not stable) are more likely to change as your segment is transcribed. Conversely, items flagged as `true` (stable) won't change.

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

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

Partial-result stabilization example output

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

```
"Transcript": {
    "Results": [
        {
            "Alternatives": [
                {
                    "Items": [
                        {
                            "Content": "Welcome",
                            "EndTime": 2.4225,
                            "Stable": true,
                            "StartTime": 1.65,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": "to",
                            "EndTime": 2.8325,
                            "Stable": false,
                            "StartTime": 2.4225,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": "Amazon",
                            "EndTime": 3.635,
                            "Stable": false,
                            "StartTime": 2.8325,
                            "Type": "pronunciation",
                            "VocabularyFilterMatch": false
                        },
                        {
                            "Content": ".",
                            "EndTime": 3.635,
                            "Stable": false,
                            "StartTime": 3.635,
                            "Type": "punctuation",
                            "VocabularyFilterMatch": false
                        }
                    ],
                    "Transcript": "Welcome to Amazon."
                }
            ],
            "EndTime": 4.165,
            "IsPartial": true,
            "ResultId": "12345a67-8bc9-0de1-2f34-a5b678c90d12",
            "StartTime": 1.65
        }
    ]
}
```

Setting up a streaming transcription

This section expands on the main streaming section. It's intended to provide information for users who want to set up their stream with HTTP/2 or WebSockets directly, rather than with an Amazon SDK. The information in this section can also be used to build your own SDK.

Important

We strongly recommend using SDKs rather than using HTTP/2 and WebSockets directly. SDKs are the simplest and most reliable method for transcribing data streams. To start streaming using an Amazon SDK, see Transcribing with the Amazon SDKs.

Setting up an HTTP/2 stream

The key components for an HTTP/2 protocol for streaming transcription requests with Amazon Transcribe are:

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

Note

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

1. Attach the following policy to the IAM role that makes the request. See Adding IAM policies for more information.
2. To start the session, send an HTTP/2 request to Amazon Transcribe.

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

Additional operations and parameters are listed in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

Amazon Transcribe sends the following response:

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

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

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

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

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

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

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

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

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

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

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

b. Construct a string to sign, as outlined in Create a string to sign for Signature Version 4. Your string follows this format:

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

- **DateTime**: The date and time the signature is created. The format is YYYYMMDDTHHMMSSZ, where YYYY=year, MM=month, DD=day, HH=hour, MM=minute, SS=seconds, and 'T' and 'Z' are fixed characters. For more information, refer to Handling Dates in Signature Version 4.
- **Keypath**: The signature scope in the format `date/region/service/aws4_request`. For example, `20220127/us-west-2/transcribe/aws4_request`.
- **Hex**: A function that encodes input into a hexadecimal representation.
- **priorSignature**: The signature for the previous frame. For the first data frame, use the signature of the header frame.
- **HexHash**: A function that first creates a SHA-256 hash of its input and then uses the Hex function to encode the hash.
- **nonSignatureHeaders**: The DateTime header encoded as a string.
- **payload**: The byte buffer containing the audio event data.

c. Derive a signing key from your Amazon secret access key and use it to sign the `stringToSign`. For a greater degree of protection, the derived key is specific to the date, service, and Amazon Web Services Region. For more information, see Calculate the signature for AmazonSignature Version 4.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Handling HTTP/2 streaming errors

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

The response contains the standard prelude and the following headers:

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	16	application/json

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
11	:event-type	7	19	BadReques tException
13	:message-type	7	9	exception

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

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

Exception message
```

Setting up a WebSocket stream

The key components for a WebSocket protocol for streaming transcription requests with Amazon Transcribe are:

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

Note

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

1. Attach the following policy to the IAM role that makes the request. See Adding IAM policies for more information.
2. To start the session, create a presigned URL in the following format. Line breaks have been added for readability.

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

Note

The maximum value for `X-Amz-Expires` is 300 (5 minutes).

Additional operations and parameters are listed in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

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

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

```
# HTTP verb
method = "GET"
# Service name
service = "transcribe"
# Region
region = "us-west-2"
# Amazon Transcribe streaming endpoint
endpoint = "wss://transcribestreaming.us-west-2.amazonaws.com:8443"
# Host
host = "transcribestreaming.us-west-2.amazonaws.com:8443"
# Date and time of request
```

```
amz-date = YYYYMMDDTHHMMSSZ
# Date without time for credential scope
datestamp = YYYYMMDD
```

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

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

c. Create the canonical headers and signed headers. Note the trailing `\n` in the canonical headers.

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

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

d. Match the algorithm to the hashing algorithm. Use `SHA-256`.

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

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

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

f. Create the canonical query string. Query string values must be URI-encoded and sorted by name.

- Sort the parameter names by character code point in ascending order. Parameters with duplicate names should be sorted by value. For example, a parameter name that begins with the uppercase letter F precedes a parameter name that begins with the lowercase letter b.

- Do not URI-encode any of the unreserved characters that RFC 3986 defines: A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).
- Percent-encode all other characters with %XY, where X and Y are hexadecimal characters (0-9 and uppercase A-F). For example, the space character must be encoded as %20 (don't include '+', as some encoding schemes do); extended UTF-8 characters must be in the form %XY%ZA%BC.
- Double-encode any equals (=) characters in parameter values.

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

g. Create a hash of the payload. For a GET request, the payload is an empty string.

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

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

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

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

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

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

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

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

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

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

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

   After you calculate the signature, add it to the query string. For more information, see Add the Signature to the Request.

   First, add the authentication information to the query string.

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

   Second, create the URL for the request.

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

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

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

   ```
   Host: transcribestreaming.us-west-2.amazonaws.com:8443
   Connection: Upgrade
   Upgrade: websocket
   Origin: URI-of-WebSocket-client
   ```

```
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: randomly-generated-string
```

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

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

```
HTTP/1.1 101 WebSocket Protocol Handshake

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

8. Make your WebSocket streaming request.

   After the WebSocket connection is established, the client can start sending a sequence of audio frames, each encoded using event stream encoding.

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

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

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

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

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

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

Handling WebSocket streaming errors

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

Header name byte length	Header name (string)	Header value type	Value string byte length	Value string (UTF-8)
13	:content-type	7	16	application/json
15	:exception-type	7	varies	varies, see below
13	:message-type	7	9	exception

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

- **`BadRequestException`**: There was a client error when the stream was created, or an error occurred while streaming data. Make sure 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. Reduce the number of streams that you're transcribing.
- **`UnrecognizedClientException`**: The WebSocket upgrade request was signed with an incorrect access key or secret key. Make sure you're correctly creating the access key and try your request again.

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

Event stream encoding

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

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

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

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

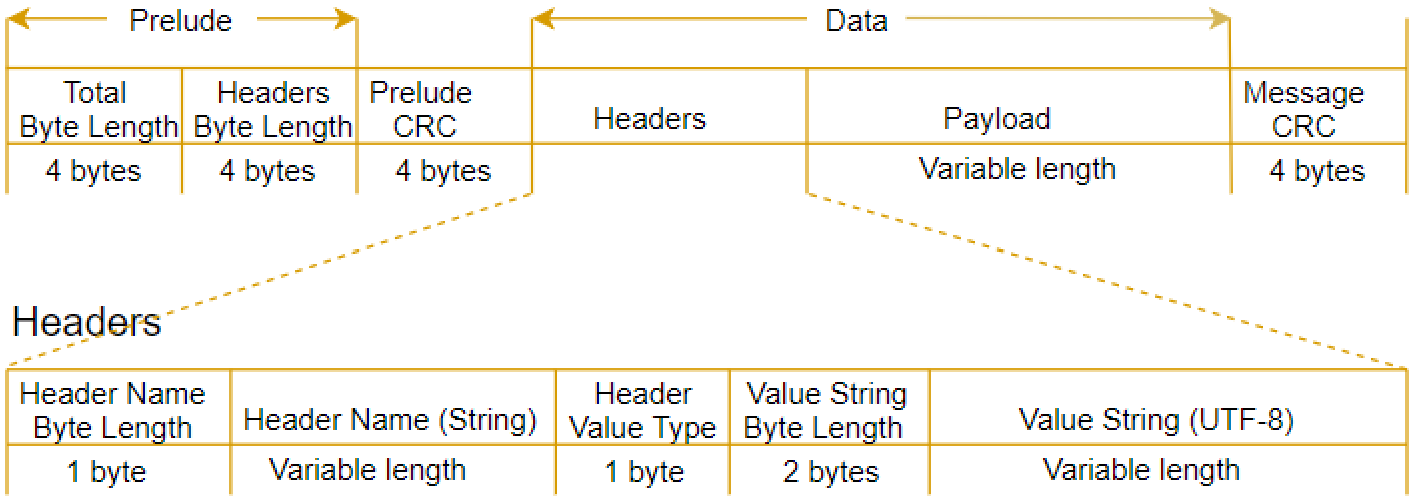
The data section consists of:

1. Headers
2. Payload

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

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

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



Each message contains the following components:

- **Prelude**: Consists of two, 4-byte fields, for a fixed total of 8 bytes.
  - *First 4 bytes*: The big-endian integer byte-length of the entire message, inclusive of this 4-byte length field.
  - *Second 4 bytes*: The big-endian integer byte-length of the 'headers' portion of the message, excluding the 'headers' length field itself.
- **Prelude CRC**: The 4-byte CRC checksum for the prelude portion of the message, excluding the CRC itself. The prelude has a separate CRC from the message CRC. That ensures that Amazon Transcribe can detect corrupted byte-length information immediately without causing errors, such as buffer overruns.

- **Headers**: Metadata annotating the message; for example, message type and content type. Messages have multiple headers, which are key:value pairs, where the key is a UTF-8 string. Headers can appear in any order in the 'headers' portion of the message, and each header can appear only once.
- **Payload**: The audio content to be transcribed.
- **Message CRC**: The 4-byte CRC checksum from the start of the message to the start of the checksum. That is, everything in the message except the CRC itself.

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

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

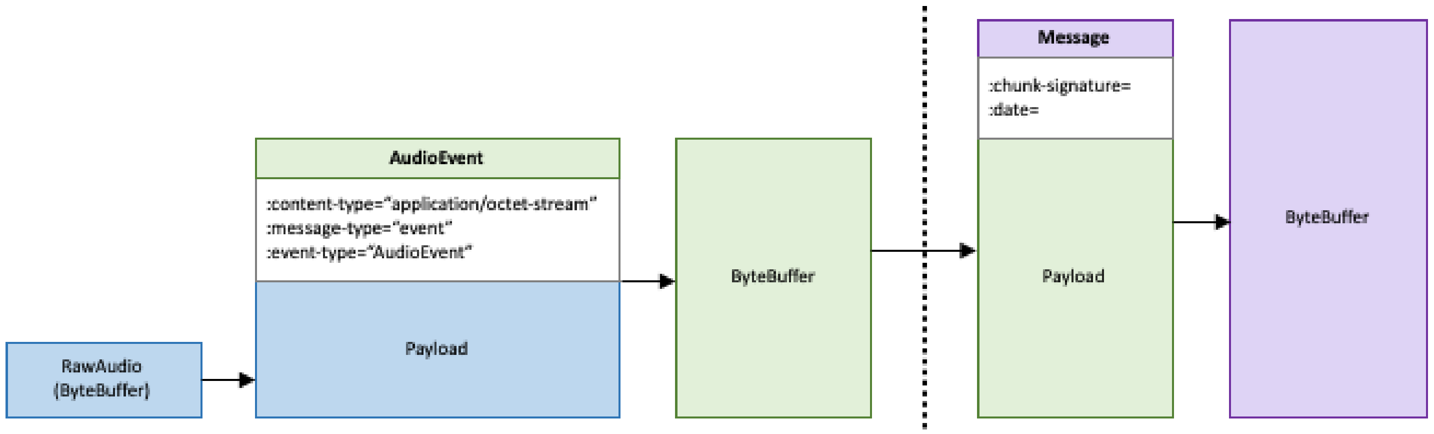
- **Header name byte-length**: The byte-length of the header name.
- **Header name**: The name of the header that indicates the header type. For valid values, see the following frame descriptions.
- **Header value type**: A number indicating the header value. The following list shows the possible values for the header and what they indicate.
  - `0` – TRUE
  - 1 – FALSE
  - 2 – BYTE
  - 3 – SHORT
  - 4 – INTEGER
  - 5 – LONG
  - 6 – BYTE ARRAY
  - 7 – STRING
  - 8 – TIMESTAMP
  - 9 – UUID
- **Value string byte length**: The byte length of the header value string.
- **Header value**: The value of the header string. Valid values for this field depend on the type of header. See Setting up an HTTP/2 stream or Setting up a WebSocket stream for more information.

Data frames

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

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

The following diagram shows how this works.



Job queueing

Using job queueing, you can submit more transcription job requests than can be concurrently processed. Without job queueing, once you reach the quota of allowed concurrent requests, you must wait until one or more requests are completed before submitting a new request.

Job queueing is optional for both transcription job and post-call analytics job requests.

If you enable job queueing, Amazon Transcribe creates a queue that contains all requests that exceed your limit. As soon as a request is completed, a new request is pulled from your queue and processed. Queued requests are processed in a FIFO (first in, first out) order.

You can add up to 10,000 jobs to your queue. If you exceed this limit, you get a `LimitExceededConcurrentJobException` error. To maintain optimal performance, Amazon Transcribe only uses up to 90 percent of your quota (a bandwidth ratio of 0.9) to process queued jobs. Note that these are default values that can be increased upon request.

Tip

You can find a list of default limits and quotas for Amazon Transcribe resources in the Amazon General Reference. Some of these defaults can be increased upon request.

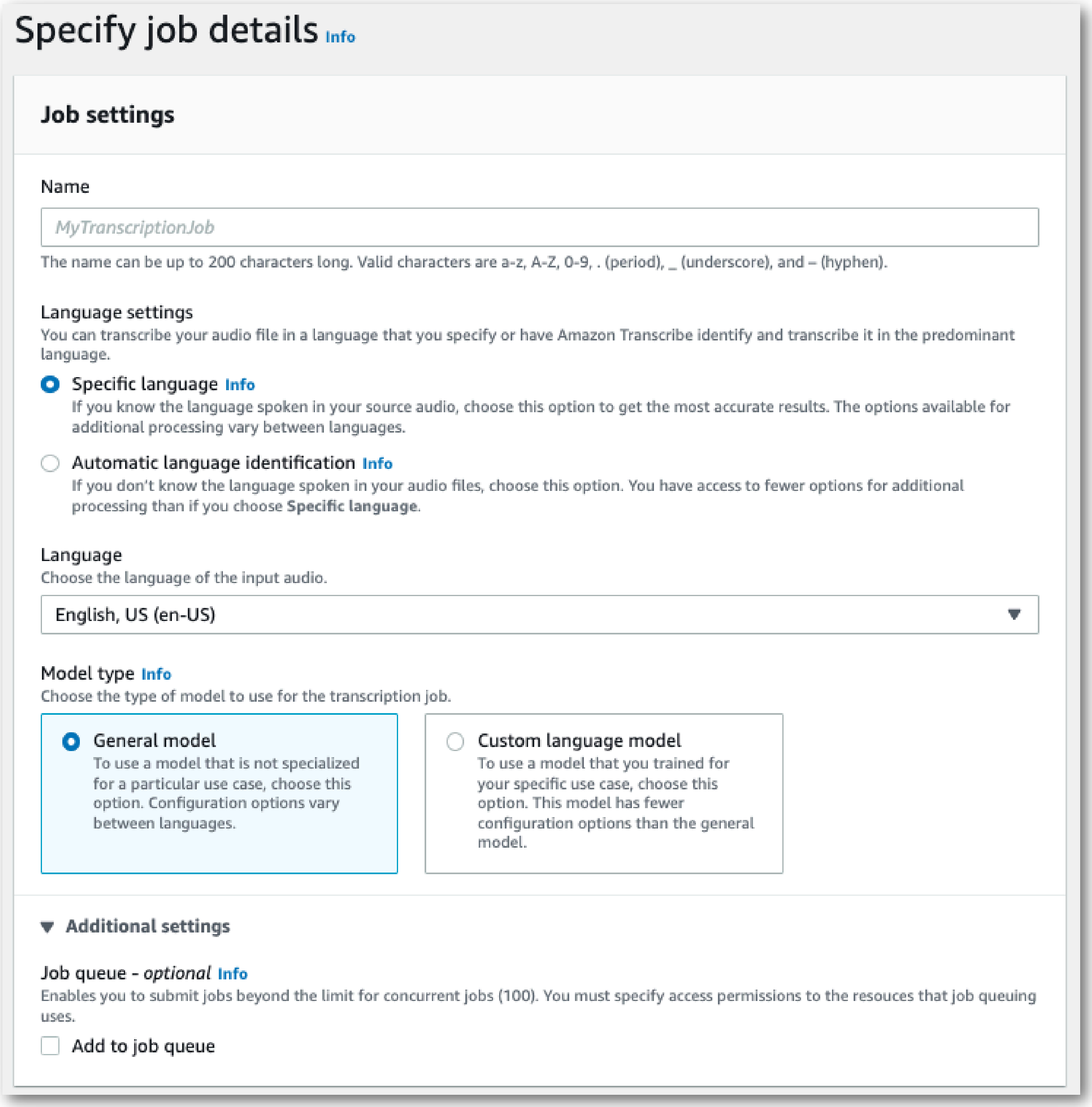
If you enable job queueing but don't exceed the quota for concurrent requests, all requests are processed concurrently.

Enabling job queueing

You can enable job queueing using the **Amazon Web Services Management Console**, **Amazon CLI**, or **Amazon SDKs**; see the following for examples; see the following for examples:

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Transcription jobs**, then select **Create job** (top right). This opens the **Specify job details** page.
3. In the **Job Settings** box, there is an **Additional settings** panel. If you expand this panel, you can select the **Add to job queue** box to enable job queueing.



Specify job details Info

Job settings

Name

MyTranscriptionJob

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

Language settings

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

[✓ Specific language] Info

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

Automatic language identification Info

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

Language

Choose the language of the input audio.

English, US (en-US)

Model type Info

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

[✓ General model]

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

Custom language model

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

Additional settings

Job queue - *optional* Info

Enables you to submit jobs beyond the limit for concurrent jobs (100). You must specify access permissions to the resouces that job queuing uses.

Add to job queue

4. Fill in any other fields you want to include on the **Specify job details** page, then select **Next**. This takes you to the **Configure job - *optional* page**.
5. Select **Create job** to run your transcription job.

Amazon CLI

This example uses the start-transcription-job command and `job-execution-settings` parameter with the `AllowDeferredExecution` sub-parameter. Note that when you include `AllowDeferredExecution` in your request, you must also include `DataAccessRoleArn`.

For more information, see `StartTranscriptionJob` and `JobExecutionSettings`.

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

Here's another example using the start-transcription-job command, and a request body that enables queueing.

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

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

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

Amazon SDK for Python (Boto3)

This example uses the Amazon SDK for Python (Boto3) to enable job queueing using the `AllowDeferredExecution` argument for the start_transcription_job method. Note that when you include `AllowDeferredExecution` in your request, you must also include `DataAccessRoleArn`. For more information, see `StartTranscriptionJob` and `JobExecutionSettings`.

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

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

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

You can view the progress of a queued job via the Amazon Web Services Management Console or by submitting a `GetTranscriptionJob` request. When a job is queued, the `Status` is QUEUED.

The status changes to `IN_PROGRESS` once your job starts processing, then changes to `COMPLETED` or `FAILED` when processing is finished.

Tagging resources

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

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

Tip

Amazon Billing and Cost Management can use tags to separate your bills into dynamic categories. For example, if you add tags to represent different departments within your company, such as `Department:Sales` or `Department:Legal`, Amazon can provide you with your cost distribution per department.

In Amazon Transcribe, you can tag the following resources:

- Transcription jobs
- Medical transcription jobs
- Call analytics post call transcription jobs
- Custom vocabularies
- Custom medical vocabularies
- Custom vocabulary filters
- Call analytics categories
- Custom language models

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

API operations specific to resource tagging

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

To learn more about tagging, including best practices, see Tagging Amazon resources.

Tag-based access control

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

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

For an example tag-based access control policy, see Viewing transcription jobs based on tags.

For more detailed information on tag-based access control, see Controlling access to Amazon resources using tags.

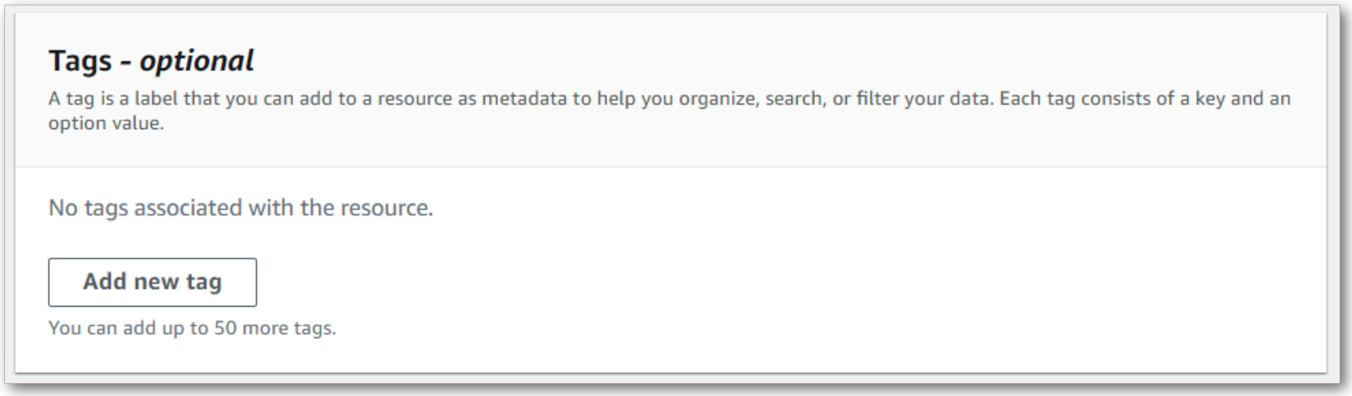
Adding tags to your Amazon Transcribe resources

You can add tags before or after you run your Amazon Transcribe job. Using the existing **Create*** and **Start*** APIs, you can add add tags with your transcription request.

You can add, modify or delete tags using the **Amazon Web Services Management Console**, **Amazon CLI**, or **Amazon SDKs**; see the following for examples:

Amazon Web Services Management Console

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



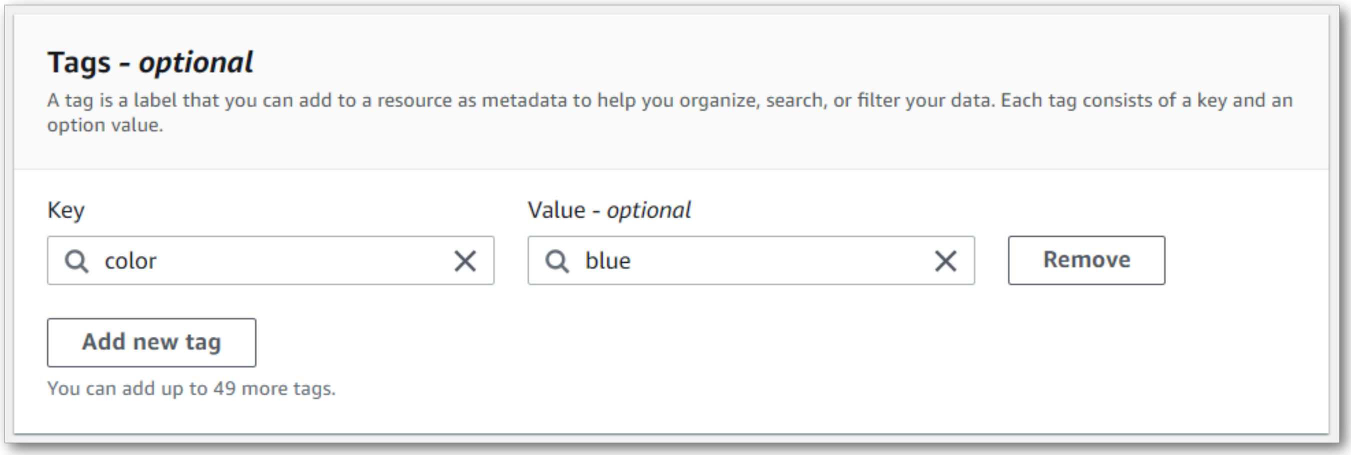
Tags - *optional*

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

No tags associated with the resource.

You can add up to 50 more tags.

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



Tags - *optional*

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

Key: color

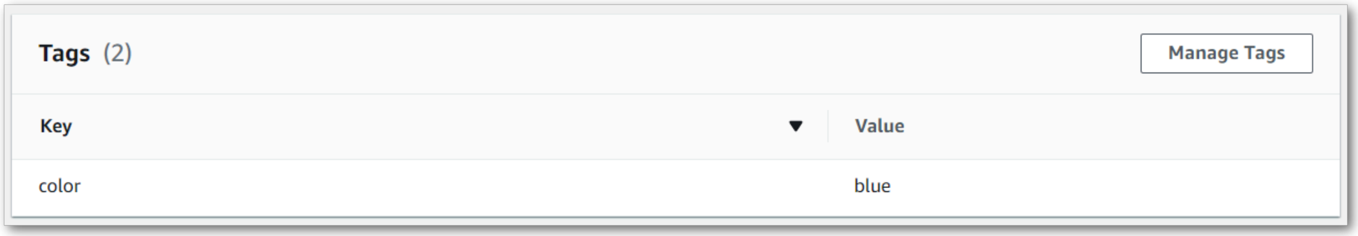
Value - *optional*: blue

You can add up to 49 more tags.

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

   Select **Create job** to run your transcription job.

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



Key	Value
color	blue

Amazon CLI

This example uses the start-transcription-job command and Tags parameter. For more information, see StartTranscriptionJob and Tag.

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

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

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

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

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

Amazon SDK for Python (Boto3)

The following example uses the Amazon SDK for Python (Boto3) to add a tag by using the `Tags` argument for the start_transcription_job method. For more information, see `StartTranscriptionJob` and `Tag`.

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

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

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

Partitioning speakers (diarization)

With speaker diarization, you can distinguish between different speakers in your transcription output. Amazon Transcribe can differentiate between a maximum of 30 unique speakers and labels the text from each unique speaker with a unique value (`spk_0` through `spk_9`).

In addition to the standard transcript sections (`transcripts` and `items`), requests with speaker partitioning enabled include a `speaker_labels` section. This section is grouped by speaker and contains information on each utterance, including speaker label and timestamps.

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

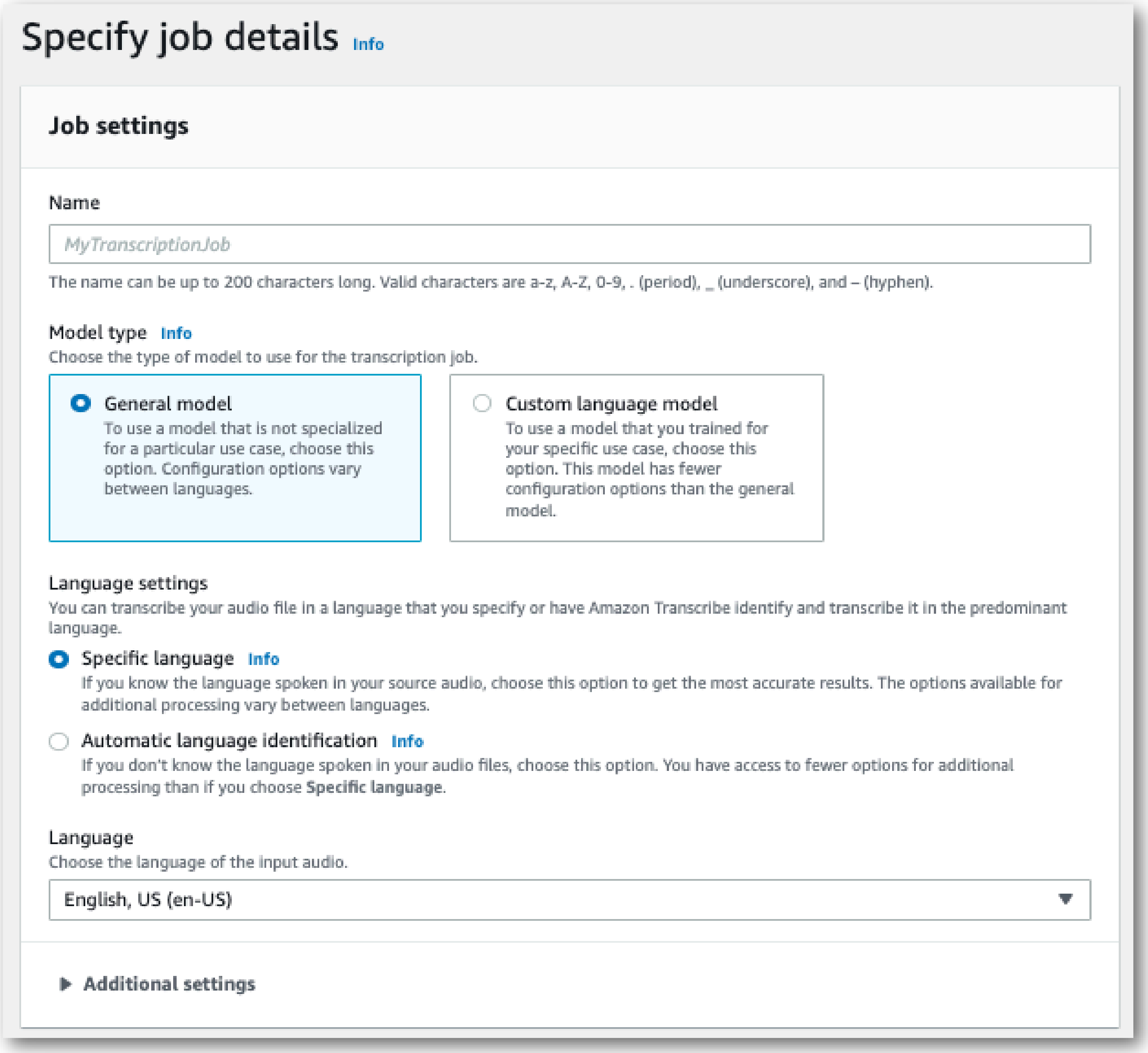
To view a complete example transcript with speaker partitioning (for two speakers), see Example diarization output (batch).

Partitioning speakers in a batch transcription

To partition speakers in a batch transcription, see the following examples:

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Transcription jobs**, then select **Create job** (top right). This opens the **Specify job details** page.



Specify job details Info

Job settings

Name

MyTranscriptionJob

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

Model type Info

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

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

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

Language settings

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

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

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

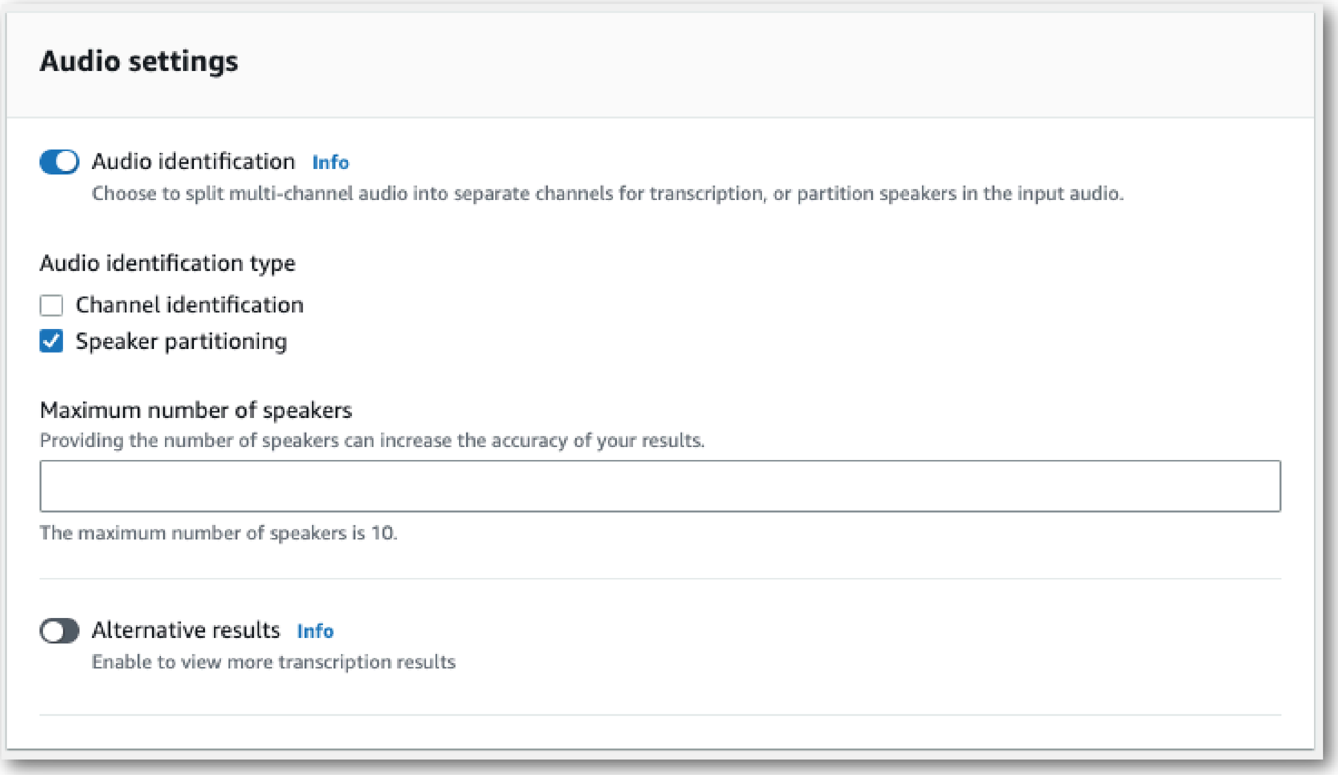
Language

Choose the language of the input audio.

English, US (en-US)

3. Fill in any fields you want to include on the **Specify job details** page, then select **Next**. This takes you to the **Configure job - *optional*** page.

   To enable speaker partitioning, in **Audio settings**, choose **Audio identification**. Then choose **Speaker partitioning** and specify the number of speakers.



Audio settings

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

Audio identification type
☐ Channel identification
☑ Speaker partitioning

Maximum number of speakers
Providing the number of speakers can increase the accuracy of your results.

The maximum number of speakers is 10.

Alternative results Info
Enable to view more transcription results

4. Select **Create job** to run your transcription job.

Amazon CLI

This example uses the start-transcription-job. For more information, see `StartTranscriptionJob`.

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
--language-code en-US \
--settings ShowSpeakerLabels=true,MaxSpeakerLabels=3
```

Here's another example using the start-transcription-job command, and a request body that enables speaker partitioning with that job.

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

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

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

Amazon SDK for Python (Boto3)

This example uses the Amazon SDK for Python (Boto3) to identify channels using the start_transcription_job method. For more information, see StartTranscriptionJob.

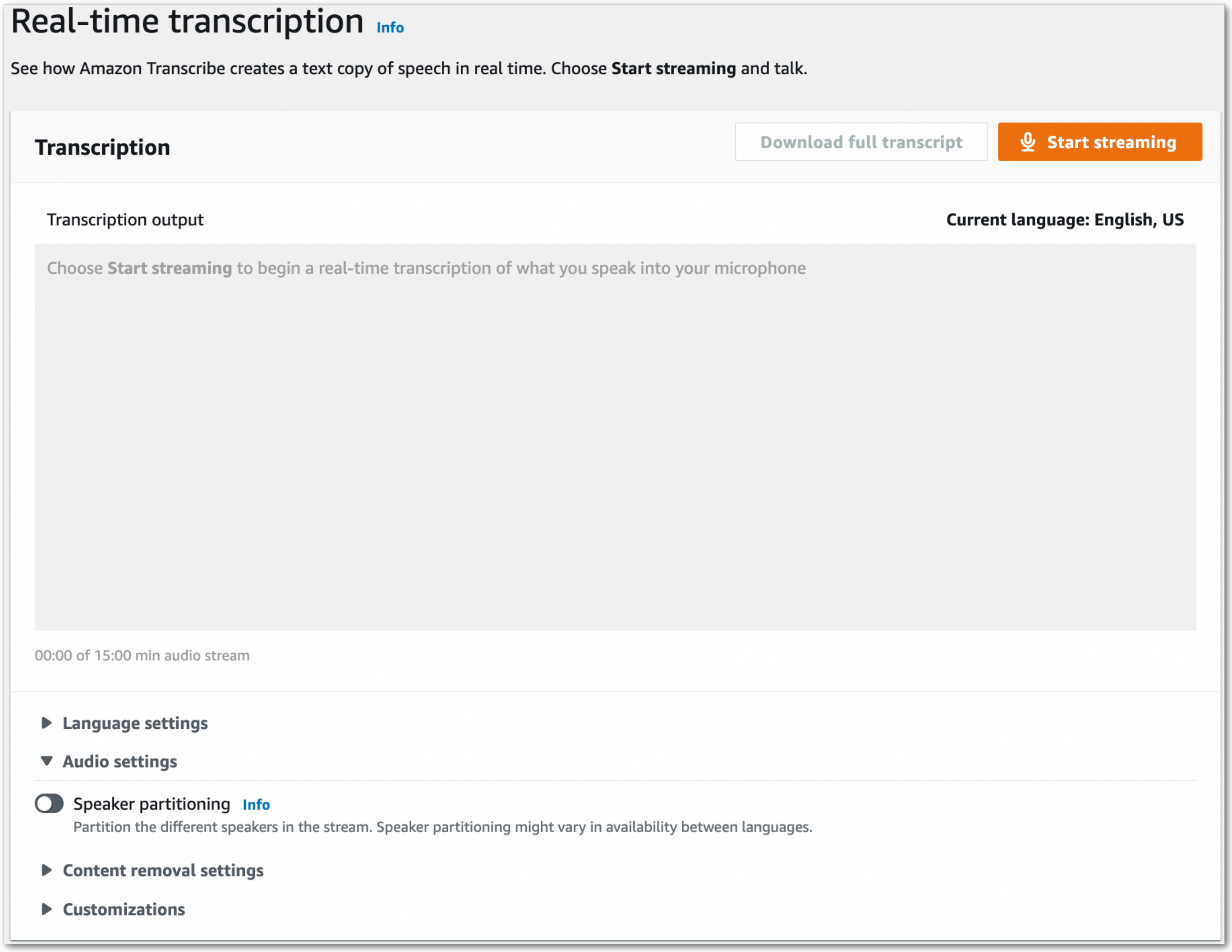
```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,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

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

Partitioning speakers in a streaming transcription

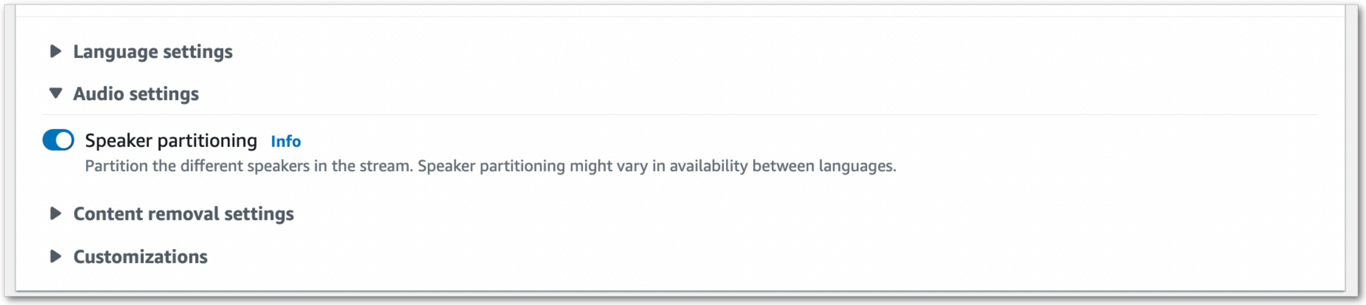
To partition speakers in a streaming transcription, see the following examples:

Streaming transcriptions

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Real-time transcription**. Scroll down to **Audio settings** and expand this field if it is minimized.


Real-time transcription Info
See how Amazon Transcribe creates a text copy of speech in real time. Choose **Start streaming** and talk.
Transcription
Download full transcript
Start streaming
Transcription output
Current language: English, US
Choose **Start streaming** to begin a real-time transcription of what you speak into your microphone
00:00 of 15:00 min audio stream
Language settings
Audio settings
Speaker partitioning Info
Partition the different speakers in the stream. Speaker partitioning might vary in availability between languages.
Content removal settings
Customizations

3. Toggle on **Speaker partitioning**.


Language settings
Audio settings
Speaker partitioning Info
Partition the different speakers in the stream. Speaker partitioning might vary in availability between languages.
Content removal settings
Customizations

4. You're now ready to transcribe your stream. Select **Start streaming** and begin speaking. To end your dictation, select **Stop streaming**.

HTTP/2 stream

This example creates an HTTP/2 request that partitions speakers in your transcription output. For more information on using HTTP/2 streaming with Amazon Transcribe, see Setting up an HTTP/2 stream. For more detail on parameters and headers specific to Amazon Transcribe, see StartStreamTranscription.

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

Parameter definitions can be found in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

WebSocket stream

This example creates a presigned URL that separates speakers in your transcription output. Line breaks have been added for readability. For more information on using WebSocket streams with Amazon Transcribe, see Setting up a WebSocket stream. For more detail on parameters, see `StartStreamTranscription`.

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
```

```
&language-code=en-US
&specialty=PRIMARYCARE
&type=DICTATION
&media-encoding=flac
&sample-rate=16000
&show-speaker-label=true
```

Parameter definitions can be found in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

Example diarization output (batch)

Here's an output example for a batch transcription with diarization enabled.

```
{
    "jobName": "my-first-transcription-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "I've been on hold for an hour. Sorry about that."
            }
        ],
        "speaker_labels": {
            "channel_label": "ch_0",
            "speakers": 2,
            "segments": [
                {
                    "start_time": "4.87",
                    "speaker_label": "spk_0",
                    "end_time": "6.88",
                    "items": [
                        {
                            "start_time": "4.87",
                            "speaker_label": "spk_0",
                            "end_time": "5.02"
                        },
                        {
                            "start_time": "5.02",
                            "speaker_label": "spk_0",
                            "end_time": "5.17"
                        },
                        {
                            "start_time": "5.17",
                            "speaker_label": "spk_0",
                            "end_time": "5.29"
                        },
                        {
                            "start_time": "5.29",
                            "speaker_label": "spk_0",
                            "end_time": "5.64"
                        },
                        {
                            "start_time": "5.64",
                            "speaker_label": "spk_0",
                            "end_time": "5.84"
                        },
                        {
                            "start_time": "6.11",
                            "speaker_label": "spk_0",
                            "end_time": "6.26"
                        },
                        {
                            "start_time": "6.26",
                            "speaker_label": "spk_0",
                            "end_time": "6.88"
                        }
                    ]
                },
                {
                    "start_time": "8.49",
                    "speaker_label": "spk_1",
                    "end_time": "9.24",
                    "items": [
                        {
                            "start_time": "8.49",
                            "speaker_label": "spk_1",
                            "end_time": "8.88"
                        },
                        {
                            "start_time": "8.88",
                            "speaker_label": "spk_1",
                            "end_time": "9.05"
                        },
                        {
                            "start_time": "9.05",
                            "speaker_label": "spk_1",
                            "end_time": "9.24"
                        }
                    ]
                }
            ]
        },
        "items": [
            {
                "id": 0,
                "start_time": "4.87",
                "speaker_label": "spk_0",
                "end_time": "5.02",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "I've"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 1,
                "start_time": "5.02",
                "speaker_label": "spk_0",
                "end_time": "5.17",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "been"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 2,
                "start_time": "5.17",
                "speaker_label": "spk_0",
                "end_time": "5.29",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "on"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 3,
                "start_time": "5.29",
                "speaker_label": "spk_0",
                "end_time": "5.64",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hold"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 4,
                "start_time": "5.64",
                "speaker_label": "spk_0",
                "end_time": "5.84",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "for"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 5,
                "start_time": "6.11",
                "speaker_label": "spk_0",
                "end_time": "6.26",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "an"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 6,
                "start_time": "6.26",
                "speaker_label": "spk_0",
                "end_time": "6.88",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hour"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 7,
                "speaker_label": "spk_0",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            },
            {
                "id": 8,
                "start_time": "8.49",
                "speaker_label": "spk_1",
                "end_time": "8.88",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Sorry"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 9,
                "start_time": "8.88",
                "speaker_label": "spk_1",
                "end_time": "9.05",
                "alternatives": [
                    {
                        "confidence": "0.902",
                        "content": "about"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 10,
                "start_time": "9.05",
                "speaker_label": "spk_1",
                "end_time": "9.24",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "that"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 11,
                "speaker_label": "spk_1",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ],
        "audio_segments": [
            {
                "id": 0,
                "transcript": "I've been on hold for an hour.",
                "start_time": "4.87",
                "end_time": "6.88",
                "speaker_label": "spk_0",
                "items": [
                    0,
                    1,
                    2,
                    3,
                    4,
                    5,
                    6,
                    7
                ]
            },
            {
                "id": 1,
                "transcript": "Sorry about that.",
                "start_time": "8.49",
                "end_time": "9.24",
                "speaker_label": "spk_1",
                "items": [
                    8,
                    9,
                    10,
                    11
                ]
            }
        ]
    },
    "status": "COMPLETED"
}
```

Transcribing multi-channel audio

If your audio has two channels, you can use channel identification to transcribe the speech from each channel separately. Amazon Transcribe doesn't currently support audio with more than two channels.

In your transcript, channels are assigned the labels `ch_0` and `ch_1`.

In addition to the standard transcript sections (`transcripts` and `items`), requests with channel identification enabled include a `channel_labels` section. This section contains each utterance or punctuation mark, grouped by channel, and its associated channel label, timestamps, and confidence score.

```
"channel_labels": {
    "channels": [
        {
            "channel_label": "ch_0",
            "items": [
                {
                    "channel_label": "ch_0",
                    "start_time": "4.86",
                    "end_time": "5.01",
                    "alternatives": [
                        {
                            "confidence": "1.0",
                            "content": "I've"
                        }
                    ],
                    "type": "pronunciation"
                },
                ...
            "channel_label": "ch_1",
            "items": [
                {
                    "channel_label": "ch_1",
                    "start_time": "8.5",
                    "end_time": "8.89",
                    "alternatives": [
                        {
                            "confidence": "1.0",
                            "content": "Sorry"
                        }
                    ],
                    "type": "pronunciation"
                },
                ...
            "number_of_channels": 2
        },
```

Note that if a person on one channel speaks at the same time as a person on a separate channel, timestamps for each channel overlap while the individuals are speaking over each other.

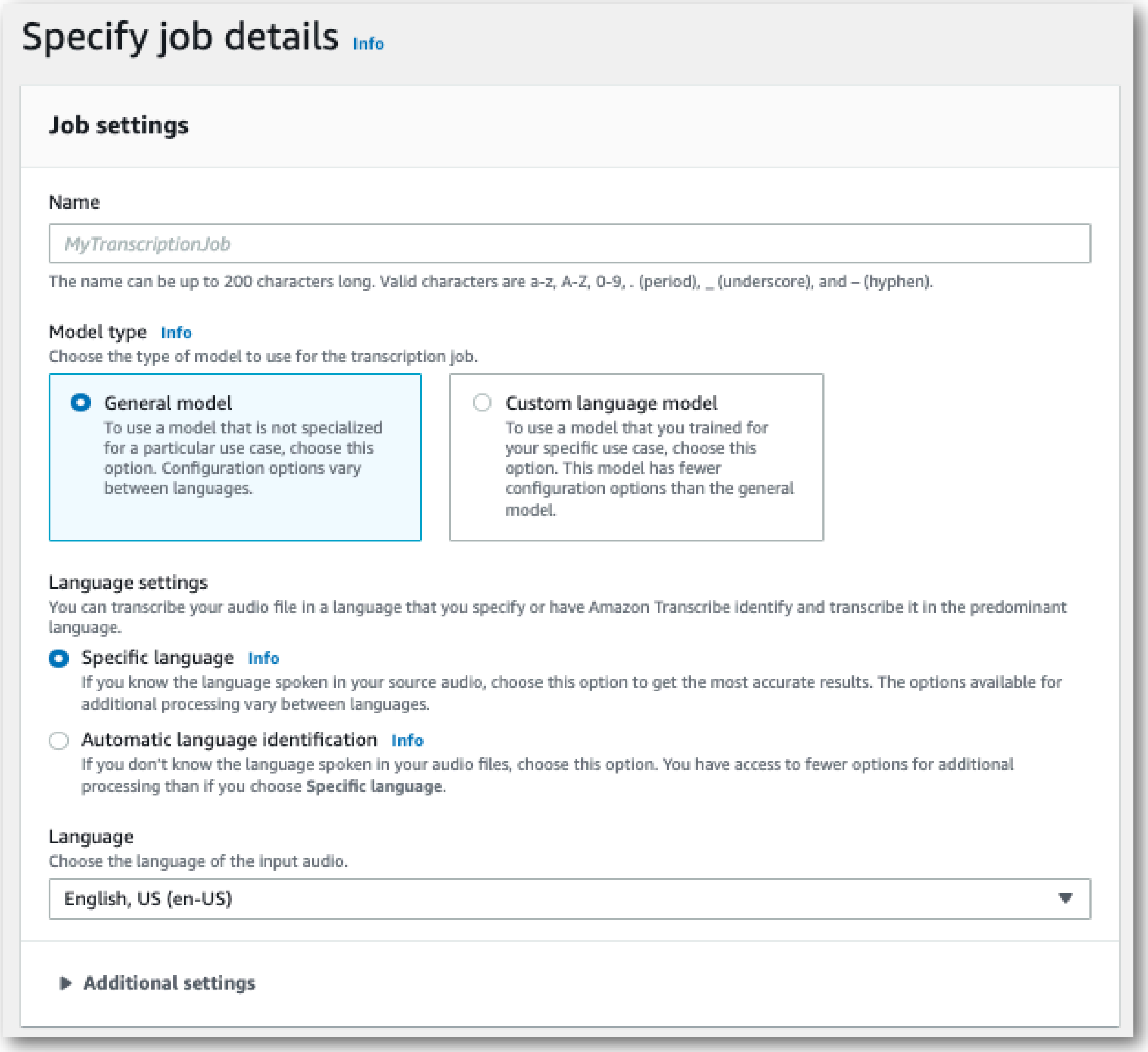
To view a complete example transcript with channel identification, see Example channel identification output (batch).

Using channel identification in a batch transcription

To identify channels in a batch transcription, you can use the **Amazon Web Services Management Console**, **Amazon CLI**, or **Amazon SDKs**; see the following for examples:

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Transcription jobs**, then select **Create job** (top right). This opens the **Specify job details** page.



Specify job details Info

Job settings

Name

MyTranscriptionJob

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

Model type Info

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

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

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

Language settings

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

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

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

Language

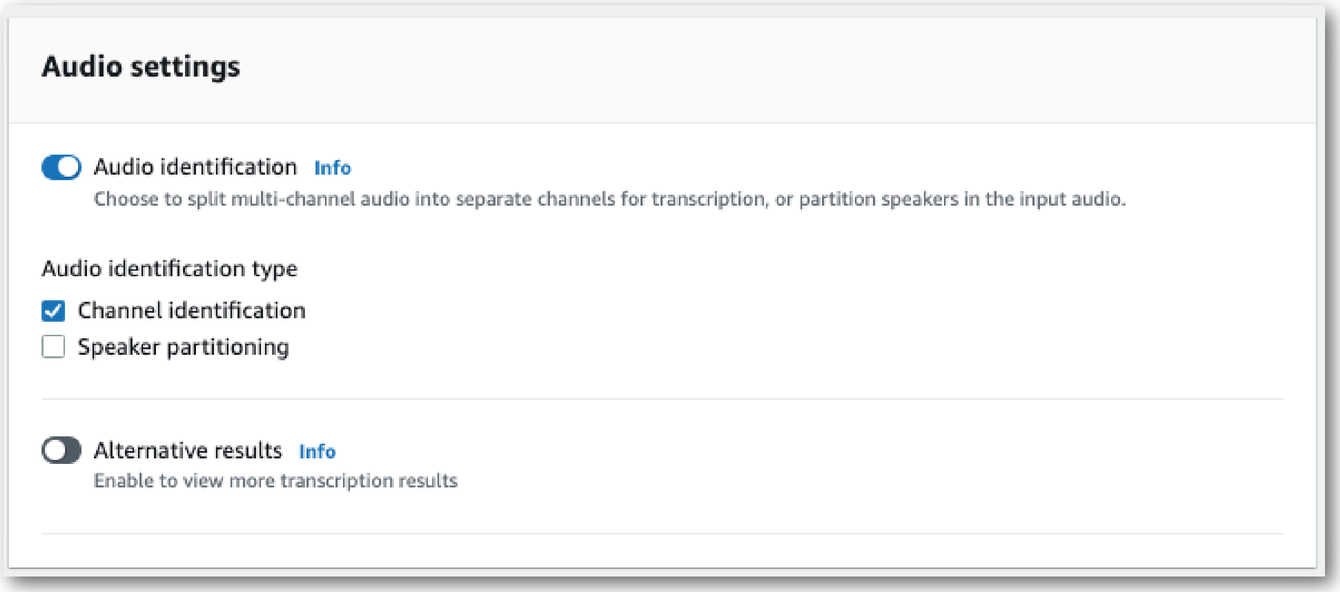
Choose the language of the input audio.

English, US (en-US)

▶ Additional settings

3. Fill in any fields you want to include on the **Specify job details** page, then select **Next**. This takes you to the **Configure job - *optional*** page.

   In the **Audio settings** panel, select **Channel identification** (under the 'Audio identification type' heading).



Audio settings

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

Audio identification type
[✓] Channel identification
[ ] Speaker partitioning

Alternative results Info
Enable to view more transcription results

4. Select **Create job** to run your transcription job.

Amazon CLI

This example uses the start-transcription-job. For more information, see StartTranscriptionJob.

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

Here's another example using the start-transcription-job command, and a request body that enables channel identification with that job.

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

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

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

Amazon SDK for Python (Boto3)

This example uses the Amazon SDK for Python (Boto3) to identify channels using the start_transcription_job method. For more information, see StartTranscriptionJob.

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

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

```
print(status)
```

Using channel identification in a streaming transcription

To identify channels in a streaming transcription, you can use **HTTP/2** or **WebSockets**; see the following for examples:

HTTP/2 stream

This example creates an HTTP/2 request that separates channels in your transcription output. For more information on using HTTP/2 streaming with Amazon Transcribe, see Setting up an HTTP/2 stream. For more detail on parameters and headers specific to Amazon Transcribe, see StartStreamTranscription.

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

Parameter definitions can be found in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

WebSocket stream

This example creates a presigned URL that separates channels in your transcription output. Line breaks have been added for readability. For more information on using WebSocket streams with Amazon Transcribe, see Setting up a WebSocket stream. For more detail on parameters, see `StartStreamTranscription`.

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

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

Parameter definitions can be found in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

Example channel identification output (batch)

Here's an output example for a batch transcription with channel identification enabled.

```
{
    "jobName": "my-first-transcription-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [
            {
                "transcript": "I've been on hold for an hour. Sorry about that."
            }
        ],
        "channel_labels": {
            "channels": [
                {
                    "channel_label": "ch_0",
                    "items": [
                        {
                            "channel_label": "ch_0",
                            "start_time": "4.86",
                            "end_time": "5.01",
                            "alternatives": [
                                {
                                    "confidence": "1.0",
                        "content": "I've"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.01",
                "end_time": "5.16",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "been"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.16",
                "end_time": "5.28",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "on"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.28",
                "end_time": "5.62",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hold"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "channel_label": "ch_0",
                "start_time": "5.62",
                            "end_time": "5.83",
                            "alternatives": [
                                {
                                    "confidence": "1.0",
                                    "content": "for"
                                }
                            ],
                            "type": "pronunciation"
                        },
                        {
                            "channel_label": "ch_0",
                            "start_time": "6.1",
                            "end_time": "6.25",
                            "alternatives": [
                                {
                                    "confidence": "1.0",
                                    "content": "an"
                                }
                            ],
                            "type": "pronunciation"
                        },
                        {
                            "channel_label": "ch_0",
                            "start_time": "6.25",
                            "end_time": "6.87",
                            "alternatives": [
                                {
                                    "confidence": "1.0",
                                    "content": "hour"
                                }
                            ],
                            "type": "pronunciation"
                        },
                        {
                            "channel_label": "ch_0",
                            "language_code": "en-US",
                            "alternatives": [
                                {
                                    "confidence": "0.0",
                                    "content": "."
                                }
                            ],
                            "type": "punctuation"
                        }
```

```
                    ]
                },
                {
                "channel_label": "ch_1",
                    "items": [
                        {
                            "channel_label": "ch_1",
                            "start_time": "8.5",
                            "end_time": "8.89",
                            "alternatives": [
                                {
                                    "confidence": "1.0",
                                    "content": "Sorry"
                                }
                            ],
                            "type": "pronunciation"
                        },
                        {
                            "channel_label": "ch_1",
                            "start_time": "8.89",
                            "end_time": "9.06",
                            "alternatives": [
                                {
                                    "confidence": "0.9176",
                                    "content": "about"
                                }
                            ],
                            "type": "pronunciation"
                        },
                        {
                            "channel_label": "ch_1",
                            "start_time": "9.06",
                            "end_time": "9.25",
                            "alternatives": [
                                {
                                    "confidence": "1.0",
                                    "content": "that"
                                }
                            ],
                            "type": "pronunciation"
                        },
                        {
                            "channel_label": "ch_1",
                            "alternatives": [
                                {
                                    "confidence": "0.0",
                                    "content": "."
                                }
                            ],
                            "type": "punctuation"
                        }
                    ]
                }
            ],
            "number_of_channels": 2
        },
        "items": [
            {
                "id": 0,
                "channel_label": "ch_0",
                "start_time": "4.86",
                "end_time": "5.01",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "I've"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 1,
                "channel_label": "ch_0",
                "start_time": "5.01",
                "end_time": "5.16",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "been"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 2,
                "channel_label": "ch_0",
                "start_time": "5.16",
                "end_time": "5.28",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "on"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 3,
                "channel_label": "ch_0",
                "start_time": "5.28",
                "end_time": "5.62",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hold"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 4,
                "channel_label": "ch_0",
                "start_time": "5.62",
                "end_time": "5.83",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "for"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 5,
                "channel_label": "ch_0",
                "start_time": "6.1",
                "end_time": "6.25",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "an"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 6,
                "channel_label": "ch_0",
                "start_time": "6.25",
                "end_time": "6.87",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "hour"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 7,
                "channel_label": "ch_0",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            },
            {
                "id": 8,
                "channel_label": "ch_1",
                "start_time": "8.5",
                "end_time": "8.89",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "Sorry"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 9,
                "channel_label": "ch_1",
                "start_time": "8.89",
                "end_time": "9.06",
                "alternatives": [
                    {
                        "confidence": "0.9176",
                        "content": "about"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 10,
                "channel_label": "ch_1",
                "start_time": "9.06",
                "end_time": "9.25",
                "alternatives": [
                    {
                        "confidence": "1.0",
                        "content": "that"
                    }
                ],
                "type": "pronunciation"
            },
            {
                "id": 11,
                "channel_label": "ch_1",
                "alternatives": [
                    {
                        "confidence": "0.0",
                        "content": "."
                    }
                ],
                "type": "punctuation"
            }
        ],
        "audio_segments": [
            {
                "id": 0,
                "transcript": "I've been on hold for an hour.",
                "start_time": "4.86",
                "end_time": "6.87",
                "channel_label": "ch_0",
                "items": [
                    0,
                    1,
                    2,
                    3,
                    4,
                    5,
                    6,
                    7
                ]
            },
            {
                "id": 1,
                "transcript": "Sorry about that.",
                "start_time": "8.5",
                "end_time": "9.25",
                "channel_label": "ch_1",
                "items": [
                    8,
                    9,
                    10,
                    11
                ]
            }
        ]
    },
    "status": "COMPLETED"
}
```

Alternative transcriptions

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

If you turn on alternative transcriptions, Amazon Transcribe returns other versions of your transcript that have lower confidence levels. You can choose to have up to 10 alternative transcriptions returned. If you specify a greater number of alternatives than what Amazon Transcribe identifies, only the actual number of alternatives is returned.

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

Alternative transcriptions are only available for batch transcriptions.

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

1. A complete final transcription for a given segment.

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

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

```
"items": [
    {
        "start_time": "1.15",
        "end_time": "1.35",
        "alternatives": [
            {
                "confidence": "1.0",
                "content": "The"
            }
        ],
        "type": "pronunciation"
    },
    {
        "start_time": "1.35",
        "end_time": "2.05",
        "alternatives": [
            {
                "confidence": "1.0",
                "content": "amazon"
            }
        ],
        "type": "pronunciation"
    },
```

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

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

4. A status at the end of your transcription output.

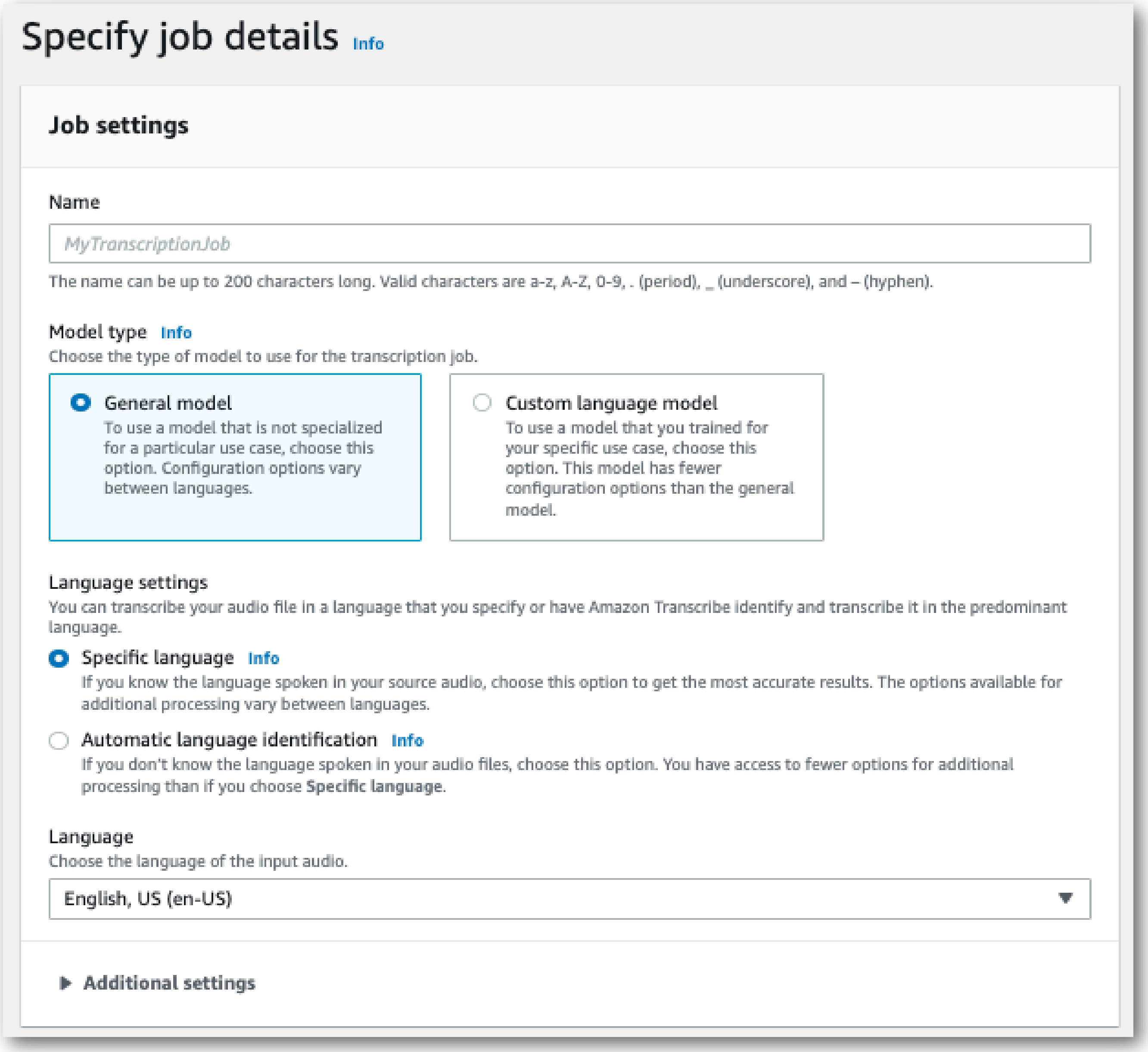
```
"status": "COMPLETED"
}
```

Requesting alternative transcriptions

You can request alternative transcriptions using the **Amazon Web Services Management Console**, **Amazon CLI**, or **Amazon SDKs**; see the following for examples:

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Transcription jobs**, then select **Create job** (top right). This opens the **Specify job details** page.



Specify job details Info

Job settings

Name

MyTranscriptionJob

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

Model type Info

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

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

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

Language settings

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

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

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

Language

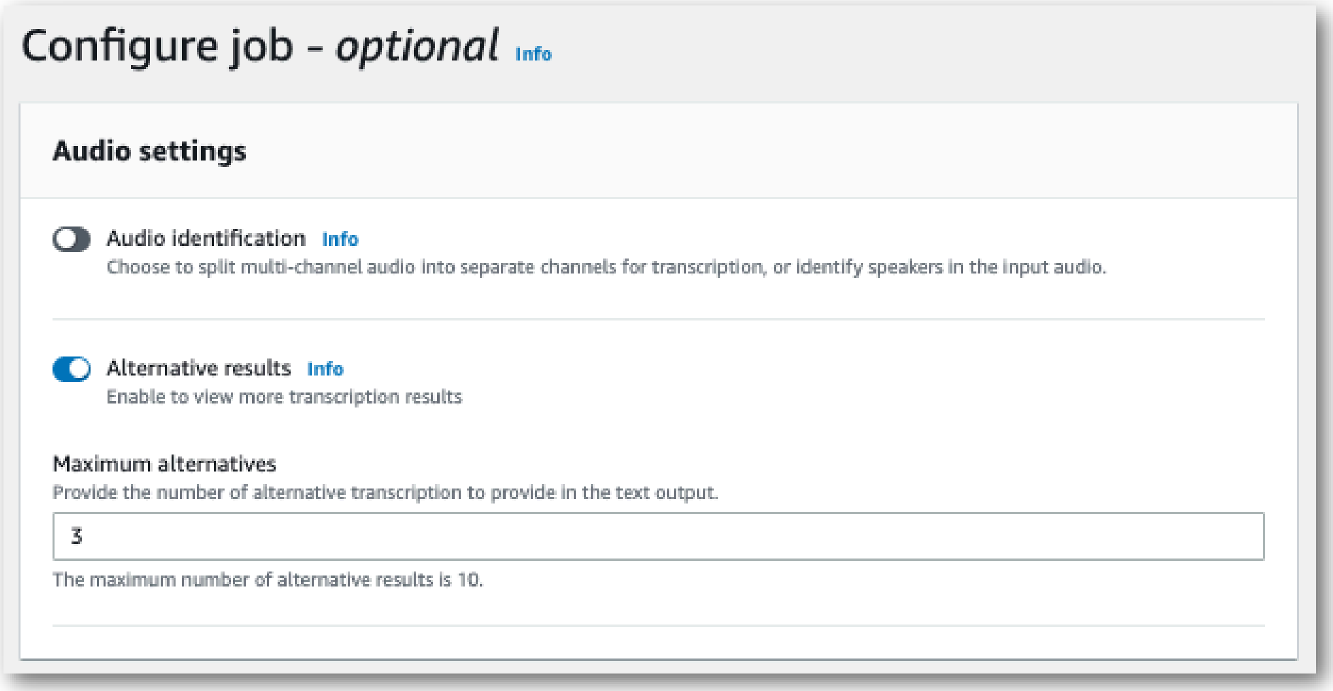
Choose the language of the input audio.

English, US (en-US)

▶ Additional settings

3. Fill in any fields you want to include on the **Specify job details** page, then select **Next**. This takes you to the **Configure job - *optional*** page.

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


Configure job - *optional* Info

Audio settings

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

Alternative results Info
Enable to view more transcription results

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

3

The maximum number of alternative results is 10.

4. Select **Create job** to run your transcription job.

Amazon CLI

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

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

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

Here's another example using the start-transcription-job command, and a request body that includes alternative transcriptions.

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

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

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

Amazon SDK for Python (Boto3)

The following example uses the Amazon SDK for Python (Boto3) to request alternative transcriptions by using the `ShowAlternatives` argument for the start_transcription_job method. For more information, see `StartTranscriptionJob` and `ShowAlternatives`.

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

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

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-bucket',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Settings = {
        'ShowAlternatives':True,
        'MaxAlternatives':4
    }
)

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

Improving transcription accuracy

If your media contains domain-specific or non-standard terms, such as brand names, acronyms, technical words, and jargon, Amazon Transcribe might not correctly capture these terms in your transcription output.

To correct transcription inaccuracies and customize your output for your specific use case, you can create Custom vocabularies.

- Custom vocabularies are designed to tune and boost both the recognition and the formatting of specific words in all contexts. This involves supplying Amazon Transcribe with words and, optionally, pronunciation and display forms.

  If Amazon Transcribe is not correctly rendering specific terms in your transcripts, you can create a custom vocabulary file that tells Amazon Transcribe how you want these terms displayed. This word-specific approach is most appropriate for correcting terms like brand names and acronyms.

Tip

To learn more, including how to create a custom vocabulary, refer to the Custom vocabularies section.

Custom vocabularies

Use custom vocabularies to improve transcription accuracy for one or more specific words. These are generally domain-specific terms, such as brand names and acronyms, proper nouns, and words that Amazon Transcribe isn't rendering correctly.

Custom vocabularies can be used with all supported languages. Note that only the characters listed in your language's character set can be used in a custom vocabulary.

Important

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

Considerations when creating a custom vocabulary:

- You can have up to 100 custom vocabulary files per Amazon Web Services account
- The size limit for each custom vocabulary file is 50 Kb
- If using the API to create your custom vocabulary, your vocabulary file must be in text (*.txt) format. If using the Amazon Web Services Management Console, your vocabulary file can be in text (*.txt) format or comma-separated value (*.csv) format.
- Each entry within a custom vocabulary cannot exceed 256 characters
- To use a custom vocabulary, it must have been created in the same Amazon Web Services Region as your transcription.

Tip

You can test your custom vocabulary using the Amazon Web Services Management Console. Once your custom vocabulary is ready to use, log in to the Amazon Web Services Management Console, select **Real-time transcription**, scroll to **Customizations**, toggle on **Custom vocabulary**, and select your custom vocabulary from the dropdown list. Then select **start streaming**. Speak some of the words in your custom vocabulary into your microphone to see if they render correctly.

Custom vocabulary tables versus lists

Important

Custom vocabularies in list format are being deprecated. If you're creating a new custom vocabulary, use the table format.

Tables give you more options for—and more control over—the input and output of words within your custom vocabulary. With tables, you must specify multiple categories (Phrase and DisplayAs), allowing you to fine-tune your output.

Lists don't have additional options, so you can only type in entries as you want them to appear in your transcript, replacing all spaces with hyphens.

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

For more information, see Creating a custom vocabulary using a table and Creating a custom vocabulary using a list.

API operations specific to custom vocabularies

`CreateVocabulary`, `DeleteVocabulary`, `GetVocabulary`, `ListVocabularies`, `UpdateVocabulary`

Creating a custom vocabulary using a table

Using a table format is the preferred way to create your custom vocabulary. Vocabulary tables must consist of four columns (Phrase, SoundsLike, IPA, and DisplayAs), which can be included in any order:

Phrase	SoundsLike	IPA	DisplayAs
Required. Every row in your table must contain an entry in this column. Do not use spaces in this column. If your entry contains multiple words, separate each word	`SoundsLike` is no longer supported for Custom Vocabulary. Please leave the column empty. Any values in this column will be ignored. We will remove the support for this column in the future.	`IPA` is no longer supported for Custom Vocabulary. Please leave the column empty. Any values in this column will be ignored. We will remove the support for this column in the future.	Optional. Rows in this column can be left empty. You can use spaces in this column. Defines the how you want your entry to look in your transcription output. For

Phrase	SoundsLike	IPA	DisplayAs
with a hyphen (-). For example, **`Andorra-la-Vella`** or **`Los-Angeles`** . For acronyms, any pronounced letters must be separated by a period. The trailing period also needs to be pronounced. If your acronym is plural, you must use a hyphen between the acronym and the 's'. For example, 'CLI' is **`C.L.I.`** (not **`C.L.I`**) and 'ABCs' is **`A.B.C.-s`** (not **`A.B.C-s`**). If your phrase consists of both a word and an acronym, these two components must be separated by a hyphen. For example, 'DynamoDB' is **`Dynamo-D.B.`** . Do not include digits in this column; numbers must be spelled out. For example, 'VX02Q' is			example, **`Andorra-la-Vella`** in the `Phrase` column is **`Andorra la Vella`** in the `DisplayAs` column. If a row in this column is empty, Amazon Transcribe uses the contents of the `Phrase` column to determine output. You can include digits (`0-9`) in this column.

Phrase	SoundsLike	IPA	DisplayAs
`V.X.-zero-two-Q.`.			

Things to note when creating your table:

- Your table must contain all four column headers (Phrase, SoundsLike, IPA, and DisplayAs). The `Phrase` column must contain an entry on each row. The ability to provide pronunciation inputs through `IPA` and `SoundsLike` is no longer supported and you may leave the column empty. Any values in these columns will be ignored.
- Each column must be TAB or comma (,) delineated; this applies to every row in your custom vocabulary file. If a row contains empty columns, you must still include a delineator (TAB or comma) for each column.
- Spaces are only allowed within the `IPA` and `DisplayAs` columns. Do not use spaces to separate columns.
- `IPA` and `SoundsLike` are no longer supported for Custom Vocabulary. Please leave the column empty. Any values in these column will be ignored. We will remove the support for this column in the future.
- The `DisplayAs` column supports symbols and special characters (for example, C++). All other columns support the characters that are listed on your language's character set page.
- If you want to include numbers in the `Phrase` column, you must spell them out. Digits (`0-9`) are only supported in the `DisplayAs` column.
- You must save your table as a plaintext (*.txt) file in `LF` format. If you use any other format, such as `CRLF`, your custom vocabulary can't be processed.
- You must upload your custom vocabulary file into an Amazon S3 bucket and process it using `CreateVocabulary` before you can include it in a transcription request. Refer to Creating custom vocabulary tables for instructions.

Note

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

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

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

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

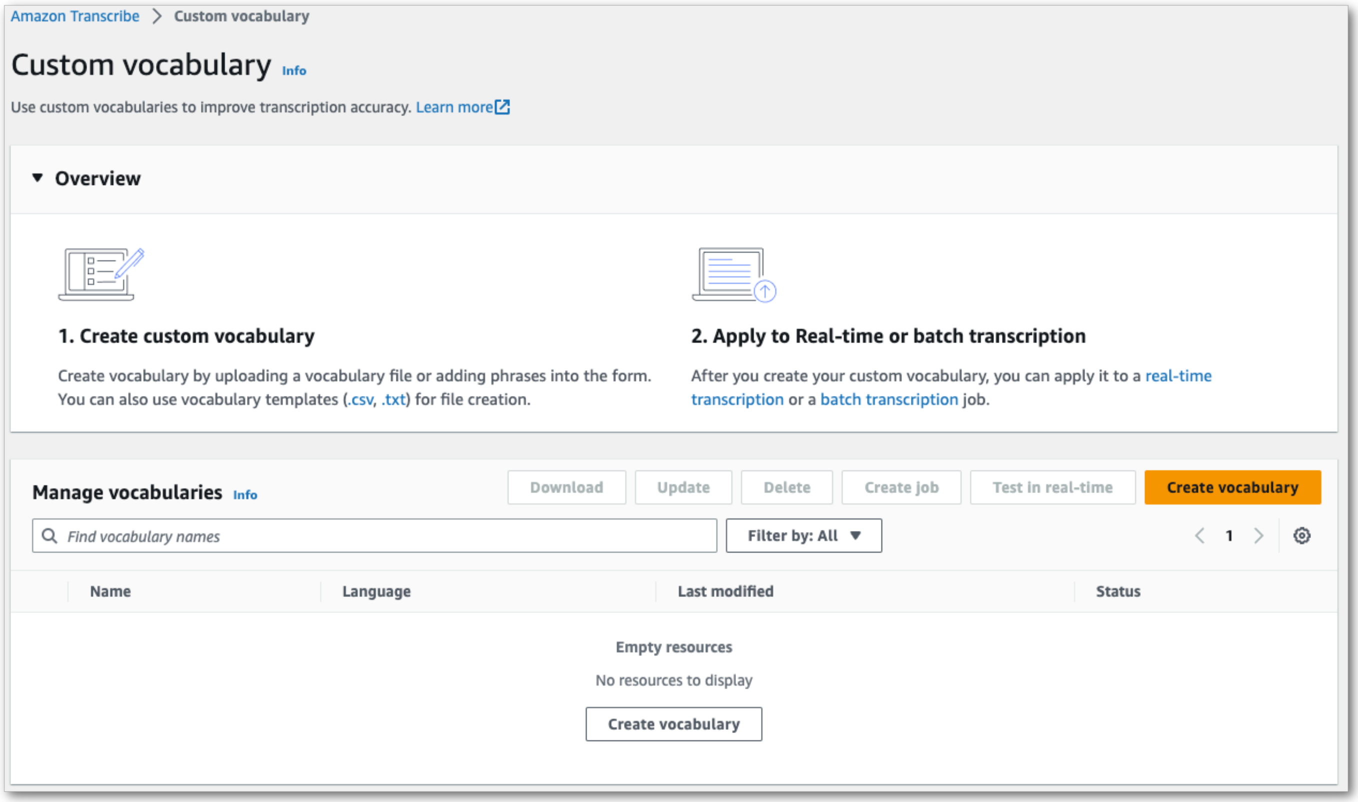
```
Phrase          [TAB]SoundsLike            [TAB]IPA                   [TAB]DisplayAs
Los-Angeles     [TAB]                      [TAB]                      [TAB]Los Angeles
Eva-Maria       [TAB]                      [TAB]                      [TAB]
A.B.C.-s        [TAB]                      [TAB]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

Creating custom vocabulary tables

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

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Custom vocabulary**. This opens the **Custom vocabulary** page where you can view existing vocabularies or create a new one.
3. Select **Create vocabulary**.



Amazon Transcribe > Custom vocabulary

Custom vocabulary Info

Use custom vocabularies to improve transcription accuracy. Learn more

▼ Overview

1. Create custom vocabulary

Create vocabulary by uploading a vocabulary file or adding phrases into the form. You can also use vocabulary templates (.csv, .txt) for file creation.

2. Apply to Real-time or batch transcription

After you create your custom vocabulary, you can apply it to a real-time transcription or a batch transcription job.

Manage vocabularies Info

Name	Language	Last modified	Status

Empty resources

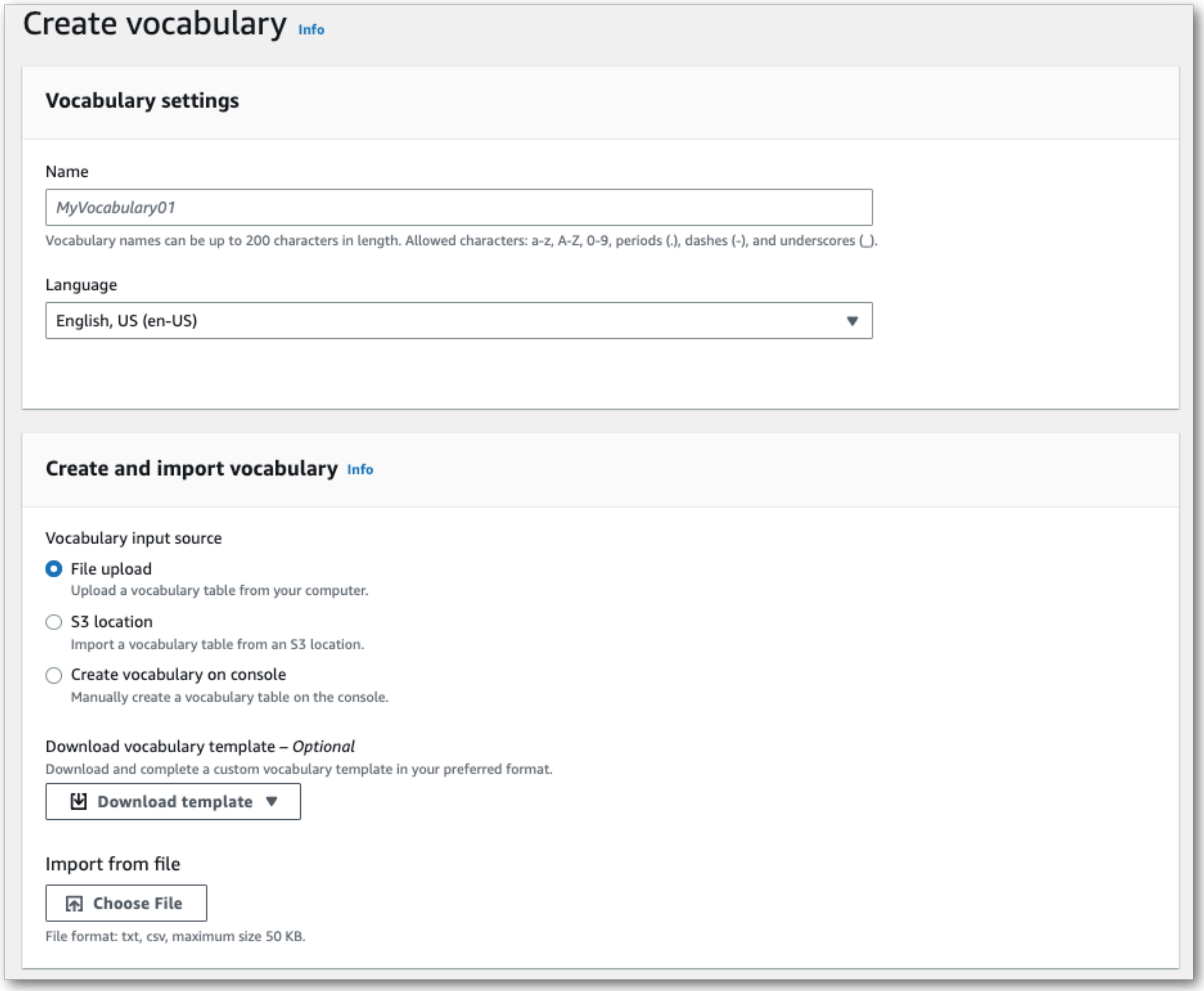
No resources to display

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

Here, you have three options:

a. Upload a txt or csv file from your computer.

   You can either create your custom vocabulary from scratch or download a template to help you get started. Your vocabulary is then auto-populated in the **View and edit vocabulary** pane.



Create vocabulary

Vocabulary settings

Name

MyVocabulary01

Vocabulary names can be up to 200 characters in length. Allowed characters: a-z, A-Z, 0-9, periods (.), dashes (-), and underscores (_).

Language

English, US (en-US)

Create and import vocabulary

Vocabulary input source

- [✓ File upload] Upload a vocabulary table from your computer.
- S3 location Import a vocabulary table from an S3 location.
- Create vocabulary on console Manually create a vocabulary table on the console.

Download vocabulary template – *Optional*

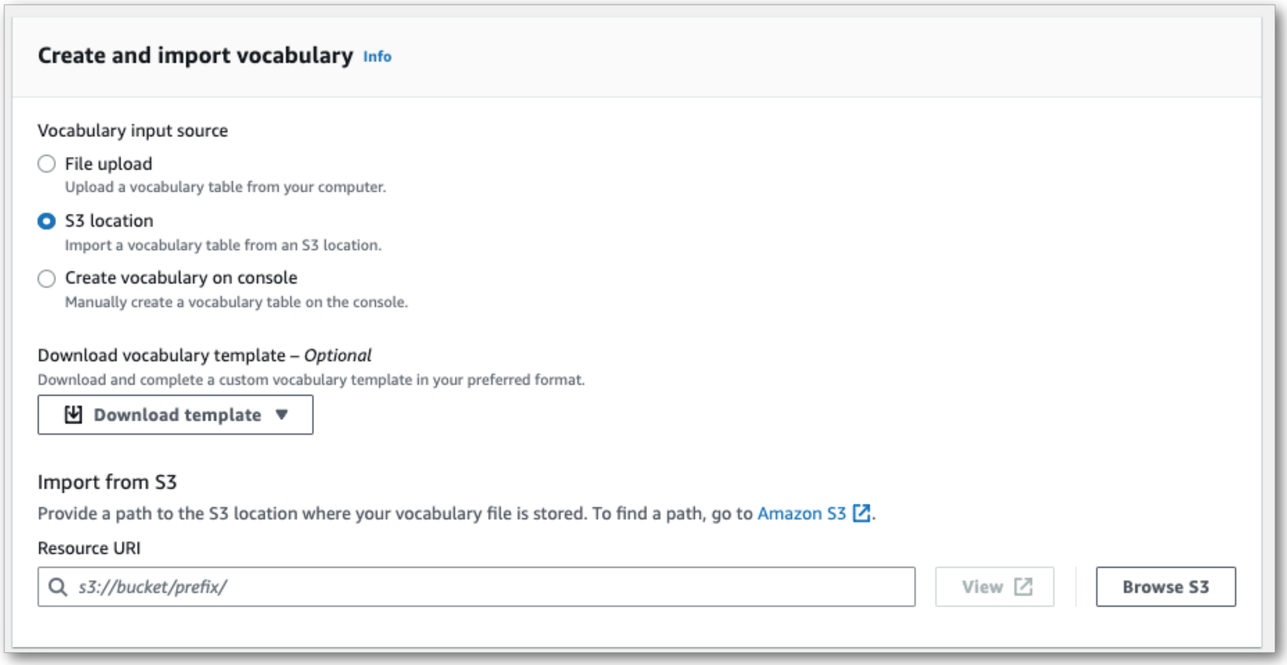
Download and complete a custom vocabulary template in your preferred format.

Import from file

File format: txt, csv, maximum size 50 KB.

b. Import a txt or csv file from an Amazon S3 location.

You can either create your custom vocabulary from scratch or download a template to help you get started. Upload your finished vocabulary file to an Amazon S3 bucket and specify its URI in your request. Your vocabulary is then auto-populated in the **View and edit vocabulary** pane.



Create and import vocabulary Info

Vocabulary input source

- File upload
  Upload a vocabulary table from your computer.
- [✓ S3 location]
  Import a vocabulary table from an S3 location.
- Create vocabulary on console
  Manually create a vocabulary table on the console.

Download vocabulary template – *Optional*
Download and complete a custom vocabulary template in your preferred format.

Download template

Import from S3
Provide a path to the S3 location where your vocabulary file is stored. To find a path, go to Amazon S3.

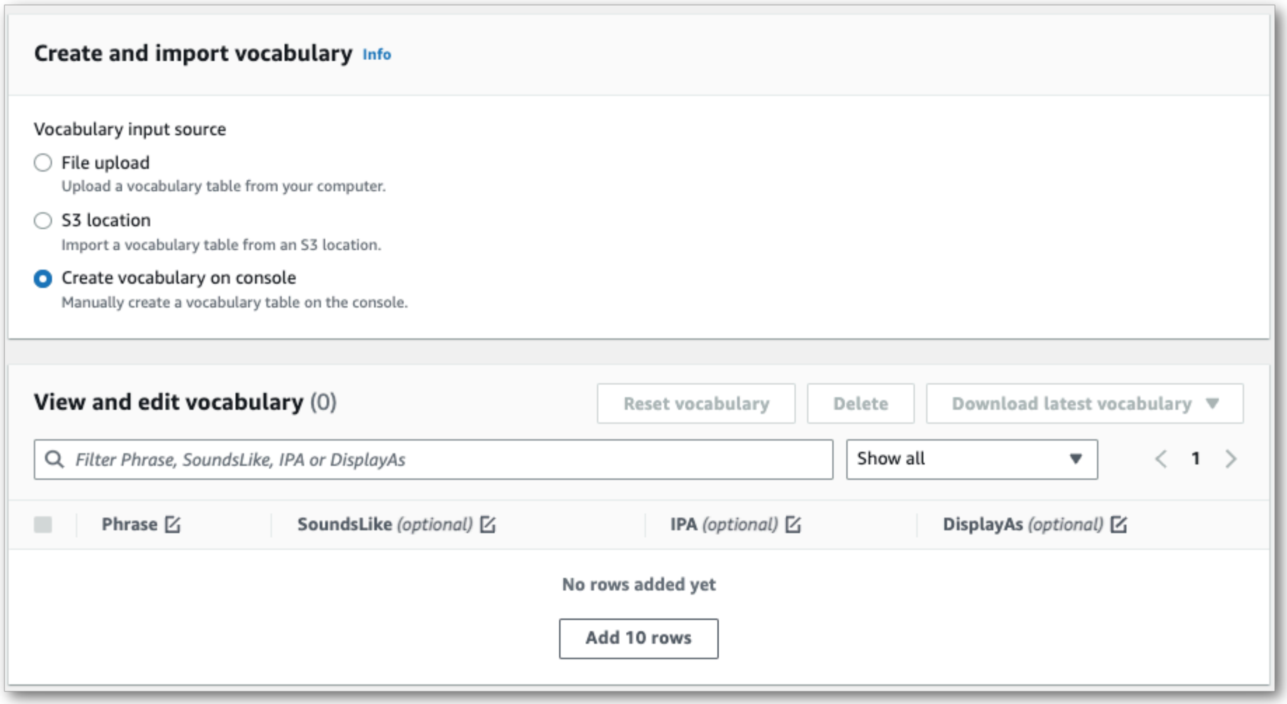
Resource URI

s3://bucket/prefix/

View | Browse S3

c. Manually create your vocabulary in the console.

Scroll to the **View and edit vocabulary** pane and select **Add 10 rows**. You can now manually enter terms.



Create and import vocabulary Info

Vocabulary input source

- File upload
  Upload a vocabulary table from your computer.
- S3 location
  Import a vocabulary table from an S3 location.
- [✓ Create vocabulary on console]
  Manually create a vocabulary table on the console.

View and edit vocabulary (0)

Reset vocabulary | Delete | Download latest vocabulary

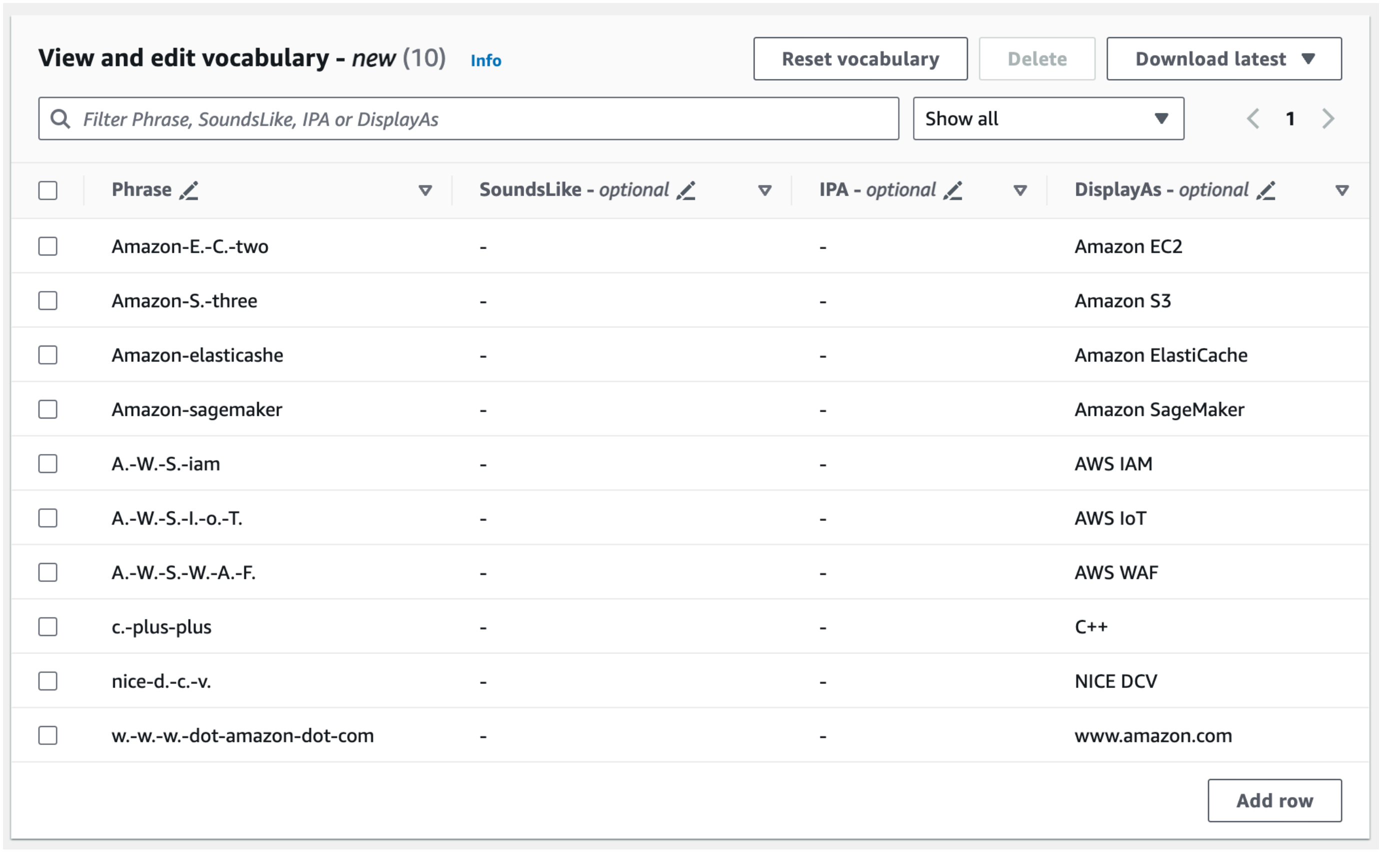
Filter Phrase, SoundsLike, IPA or DisplayAs | Show all | 1

	Phrase	SoundsLike *(optional)*	IPA *(optional)*	DisplayAs *(optional)*

No rows added yet

Add 10 rows

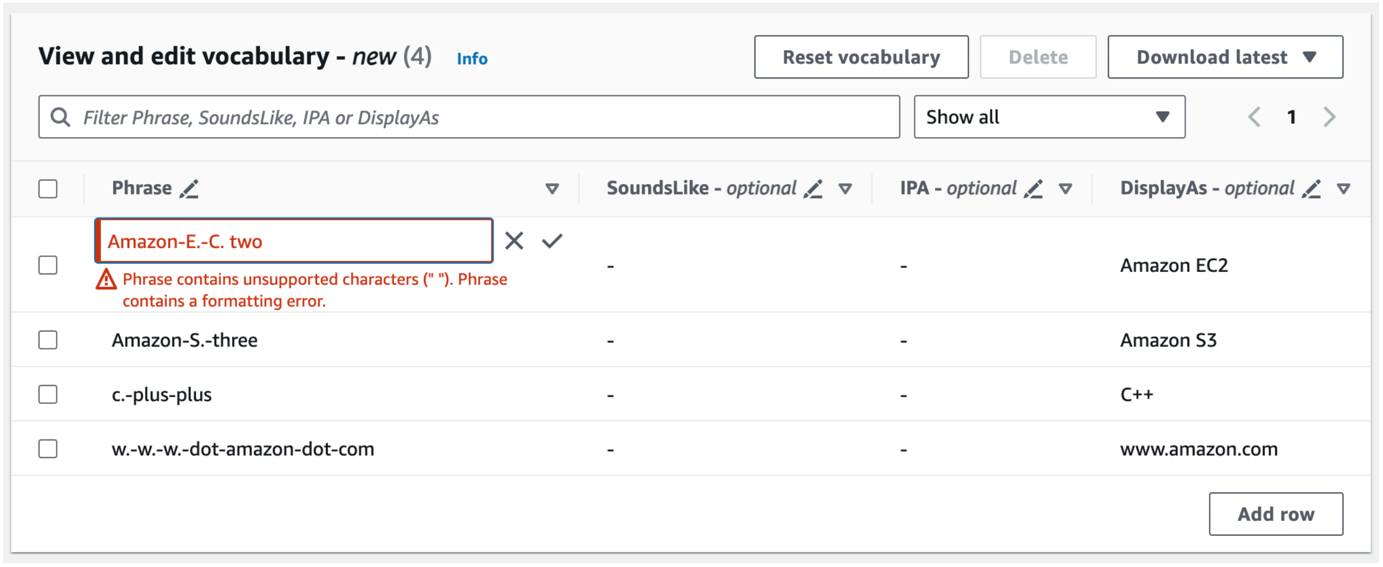
4. You can edit your vocabulary the **View and edit vocabulary** pane. To make changes, click on the entry you want to modify.



View and edit vocabulary - *new* (10) Info

Phrase	SoundsLike - *optional*	IPA - *optional*	DisplayAs - *optional*
Amazon-E.-C.-two	-	-	Amazon EC2
Amazon-S.-three	-	-	Amazon S3
Amazon-elasticashe	-	-	Amazon ElastiCache
Amazon-sagemaker	-	-	Amazon SageMaker
A.-W.-S.-iam	-	-	AWS IAM
A.-W.-S.-I.-o.-T.	-	-	AWS IoT
A.-W.-S.-W.-A.-F.	-	-	AWS WAF
c.-plus-plus	-	-	C++
nice-d.-c.-v.	-	-	NICE DCV
w.-w.-w.-dot-amazon-dot-com	-	-	www.amazon.com

If you make an error, you get a detailed error message so you can correct any issues prior to processing your vocabulary. Note that if you don't correct all errors before selecting **Create vocabulary**, your vocabulary request fails.

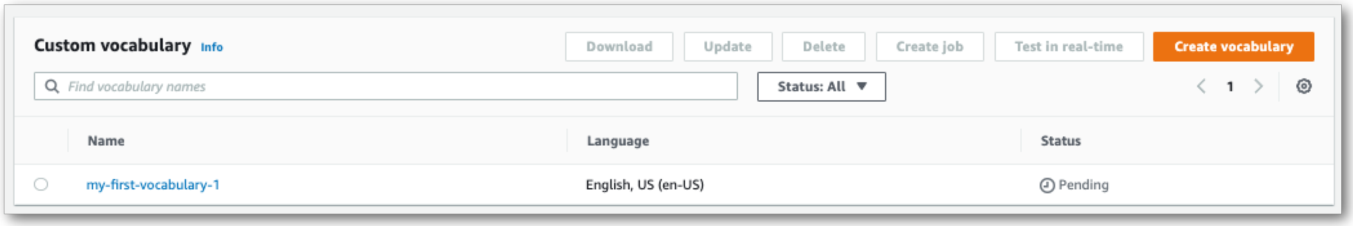


View and edit vocabulary - *new* (4) Info

Phrase	SoundsLike - *optional*	IPA - *optional*	DisplayAs - *optional*
Amazon-E.-C. two ✕ ✓ Phrase contains unsupported characters (" "). Phrase contains a formatting error.	-	-	Amazon EC2
Amazon-S.-three	-	-	Amazon S3
c.-plus-plus	-	-	C++
w.-w.-w.-dot-amazon-dot-com	-	-	www.amazon.com

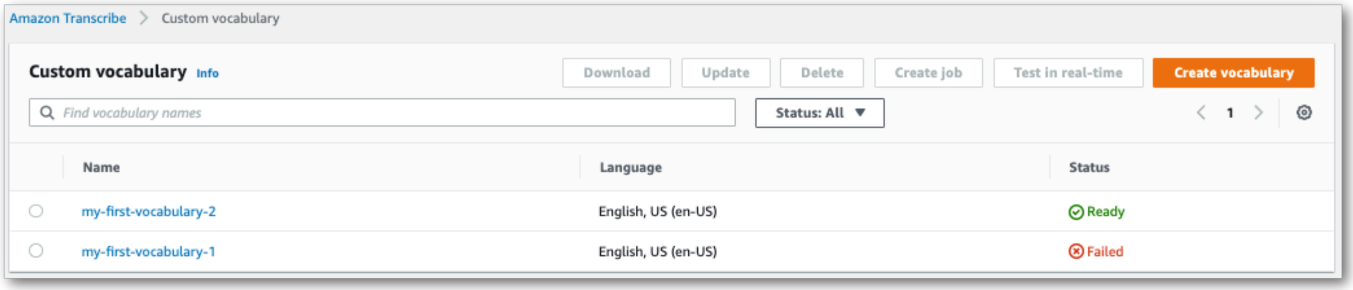
Select the check mark (✓) to save your changes or the 'X' to discard your changes.

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



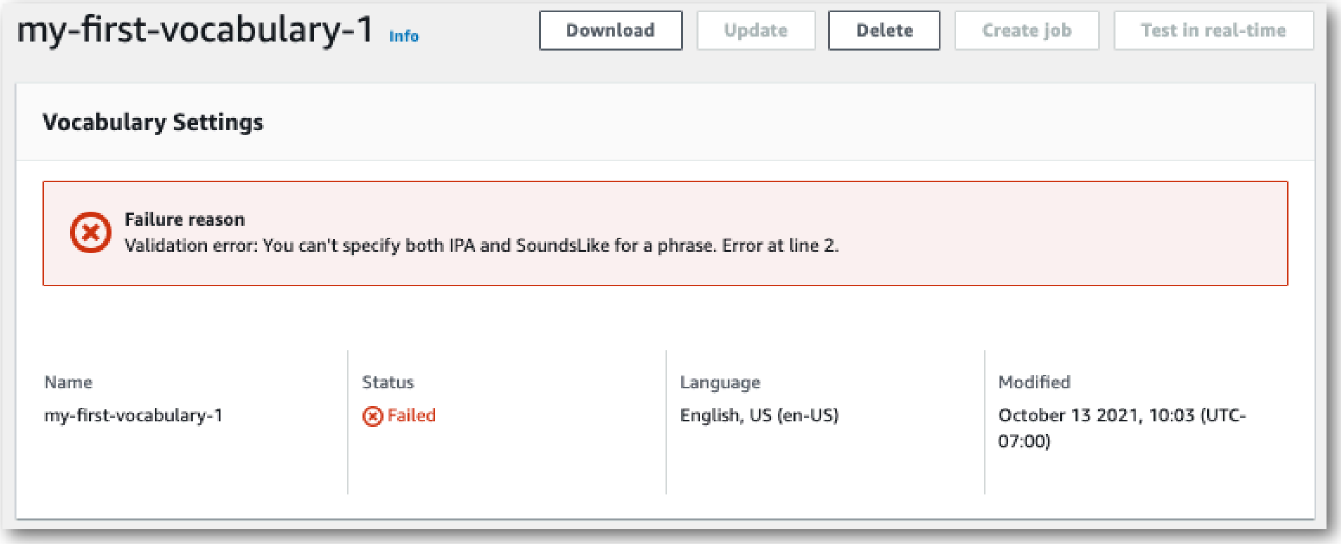
Name	Language	Status
my-first-vocabulary-1	English, US (en-US)	Pending

6. If the status changes to 'Failed', select the name of your custom vocabulary to go to its information page.



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

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



Failure reason
Validation error: You can't specify both IPA and SoundsLike for a phrase. Error at line 2.

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

Amazon CLI

This example uses the create-vocabulary command with a table-formatted vocabulary file. For more information, see `CreateVocabulary`.

To use an existing custom vocabulary in a transcription job, set the `VocabularyName` in the `Settings` field when you call the `StartTranscriptionJob` operation or, from the Amazon Web Services Management Console, choose the custom vocabulary from the dropdown list.

```
aws transcribe create-vocabulary \
--vocabulary-name my-first-vocabulary \
--vocabulary-file-uri s3://amzn-s3-demo-bucket/my-vocabularies/my-vocabulary-file.txt \
--language-code en-US
```

Here's another example using the create-vocabulary command, and a request body that creates your custom vocabulary.

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

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

```
{
  "VocabularyName": "my-first-vocabulary",
  "VocabularyFileUri": "s3://amzn-s3-demo-bucket/my-vocabularies/my-vocabulary-
table.txt",
  "LanguageCode": "en-US"
}
```

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

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

Amazon SDK for Python (Boto3)

This example uses the Amazon SDK for Python (Boto3) to create a custom vocabulary from a table using the create_vocabulary method. For more information, see `CreateVocabulary`.

To use an existing custom vocabulary in a transcription job, set the `VocabularyName` in the `Settings` field when you call the `StartTranscriptionJob` operation or, from the Amazon Web Services Management Console, choose the custom vocabulary from the dropdown list.

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

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

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

Note

If you create a new Amazon S3 bucket for your custom vocabulary files, make sure the IAM role making the `CreateVocabulary` request has permissions to access this bucket. If the role doesn't have the correct permissions, your request fails. You can optionally specify an IAM role within your request by including the `DataAccessRoleArn` parameter. For more information on IAM roles and policies in Amazon Transcribe, see Amazon Transcribe identity-based policy examples.

Creating a custom vocabulary using a list

Important

Custom vocabularies in list format are being deprecated, so if you're creating a new custom vocabulary, we strongly recommend using the table format.

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

- **Amazon Web Services Management Console**: You must create and upload a text file containing your custom vocabulary. You can use line-separated or comma-separated entries. Note that your list must be saved as a text (*.txt) file in LF format. If you use any other format, such as CRLF, your custom vocabulary is not accepted by Amazon Transcribe.
- **Amazon CLI** and **Amazon SDKs**: You must include your custom vocabulary as comma-separated entries within your API call using the `Phrases` flag.

If an entry contains multiple words, you must hyphenate each word. For example, you include 'Los Angeles' as **`Los-Angeles`** and 'Andorra la Vella' as **`Andorra-la-Vella`**.

Here are examples of the two valid list formats. Refer to Creating custom vocabulary lists for method-specific examples.

- Comma-separated entries:

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

- Line-separated entries:

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

Important

You can only use characters that are supported for your language. Refer to your language's character set for details.

Custom vocabulary lists are not supported with the `CreateMedicalVocabulary` operation. If creating a custom medical vocabulary, you must use a table format; refer to Creating a custom vocabulary using a table for instructions.

Creating custom vocabulary lists

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

Amazon CLI

This example uses the create-vocabulary command with a list-formatted custom vocabulary file. For more information, see `CreateVocabulary`.

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

Here's another example using the create-vocabulary command, and a request body that creates your custom vocabulary.

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

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

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

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

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

Amazon SDK for Python (Boto3)

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

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
vocab_name = "my-first-vocabulary"
response = transcribe.create_vocabulary(
    LanguageCode = 'en-US',
    VocabularyName = vocab_name,
    Phrases = [
        'CLI','Eva-Maria','ABCs'
    ]
)

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

Note

If you create a new Amazon S3 bucket for your custom vocabulary files, make sure the IAM role making the `CreateVocabulary` request has permissions to access this bucket. If the role doesn't have the correct permissions, your request fails. You can optionally specify an IAM role within your request by including the `DataAccessRoleArn` parameter. For more information on IAM roles and policies in Amazon Transcribe, see Amazon Transcribe identity-based policy examples.

Using a custom vocabulary

Once your custom vocabulary is created, you can include it in your transcription requests; refer to the following sections for examples.

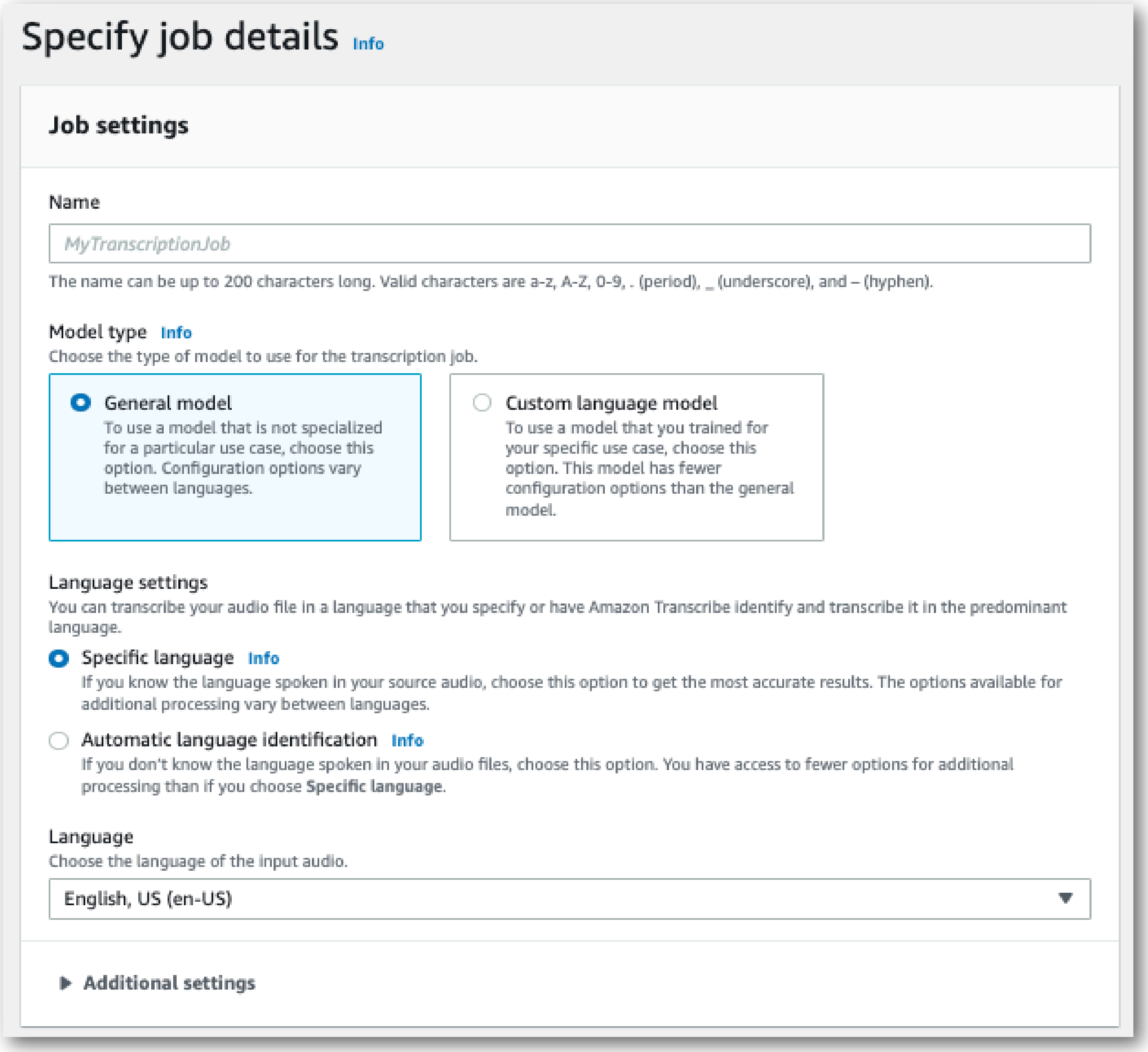
The language of the custom vocabulary you're including in your request must match the language code you specify for your media. If the languages don't match, your custom vocabulary is not applied to your transcription and there are no warnings or errors.

Using a custom vocabulary in a batch transcription

To use a custom vocabulary with a batch transcription, see the following for examples:

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Transcription jobs**, then select **Create job** (top right). This opens the **Specify job details** page.



Specify job details Info

Job settings

Name

MyTranscriptionJob

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

Model type Info

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

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

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

Language settings

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

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

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

Language

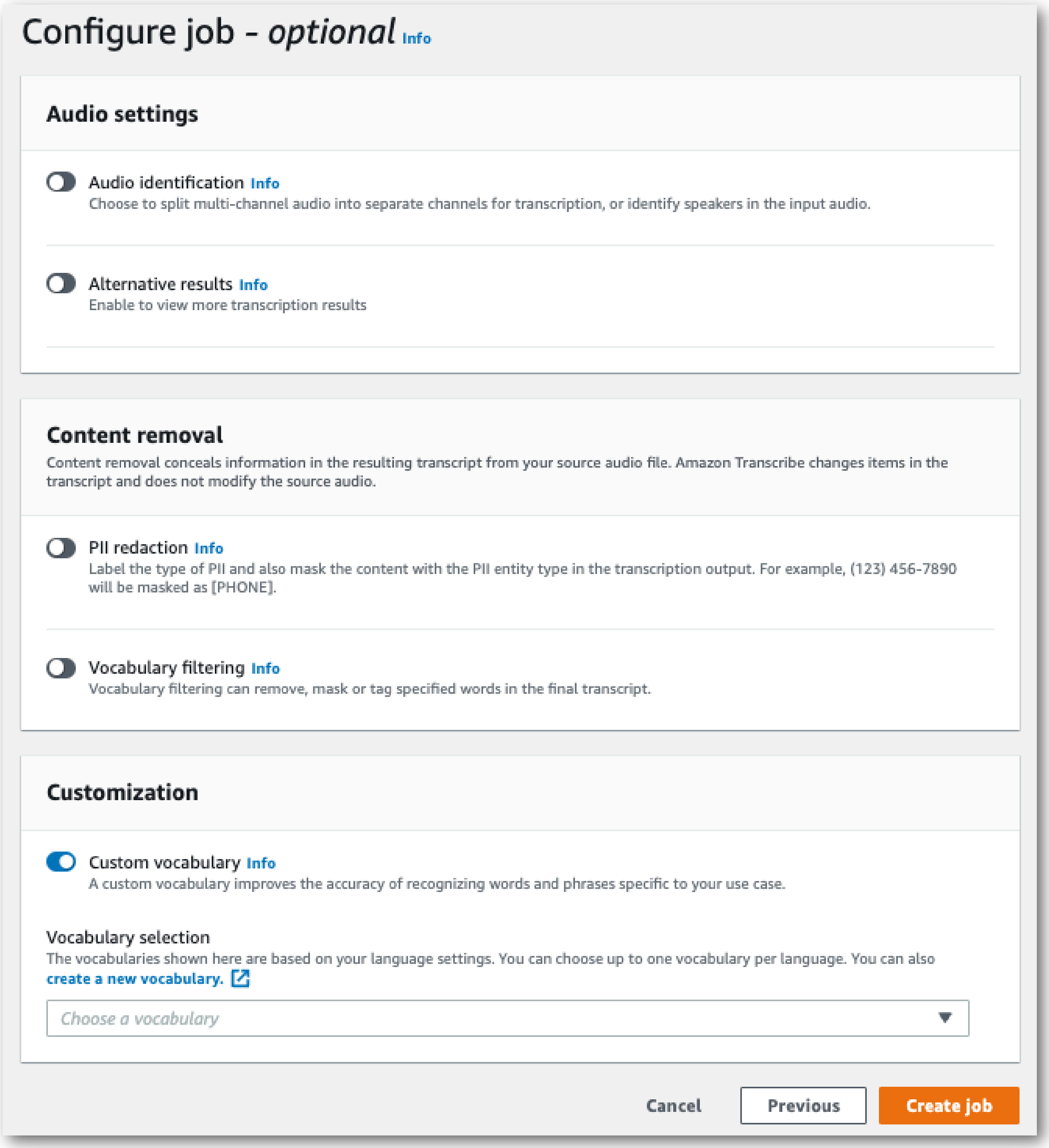
Choose the language of the input audio.

English, US (en-US)

▶ Additional settings

Name your job and specify your input media. Optionally include any other fields, then choose **Next**.

3. At the bottom of the **Configure job** page, in the **Customization** panel, toggle on **Custom vocabulary**.


Configure job - *optional* Info

Audio settings

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

Alternative results Info
Enable to view more transcription results

Content removal
Content removal conceals information in the resulting transcript from your source audio file. Amazon Transcribe changes items in the transcript and does not modify the source audio.

PII redaction Info
Label the type of PII and also mask the content with the PII entity type in the transcription output. For example, (123) 456-7890 will be masked as [PHONE].

Vocabulary filtering Info
Vocabulary filtering can remove, mask or tag specified words in the final transcript.

Customization

Custom vocabulary Info
A custom vocabulary improves the accuracy of recognizing words and phrases specific to your use case.

Vocabulary selection
The vocabularies shown here are based on your language settings. You can choose up to one vocabulary per language. You can also create a new vocabulary.

4. Select your custom vocabulary from the dropdown menu.

   Select **Create job** to run your transcription job.

Amazon CLI

This example uses the start-transcription-job command and `Settings` parameter with the `VocabularyName` sub-parameter. For more information, see `StartTranscriptionJob` and `Settings`.

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

Here's another example using the start-transcription-job command, and a request body that includes your custom vocabulary with that job.

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

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

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

Amazon SDK for Python (Boto3)

This example uses the Amazon SDK for Python (Boto3) to include a custom vocabulary using the `Settings` argument for the start_transcription_job method. For more information, see `StartTranscriptionJob` and `Settings`.

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

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

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

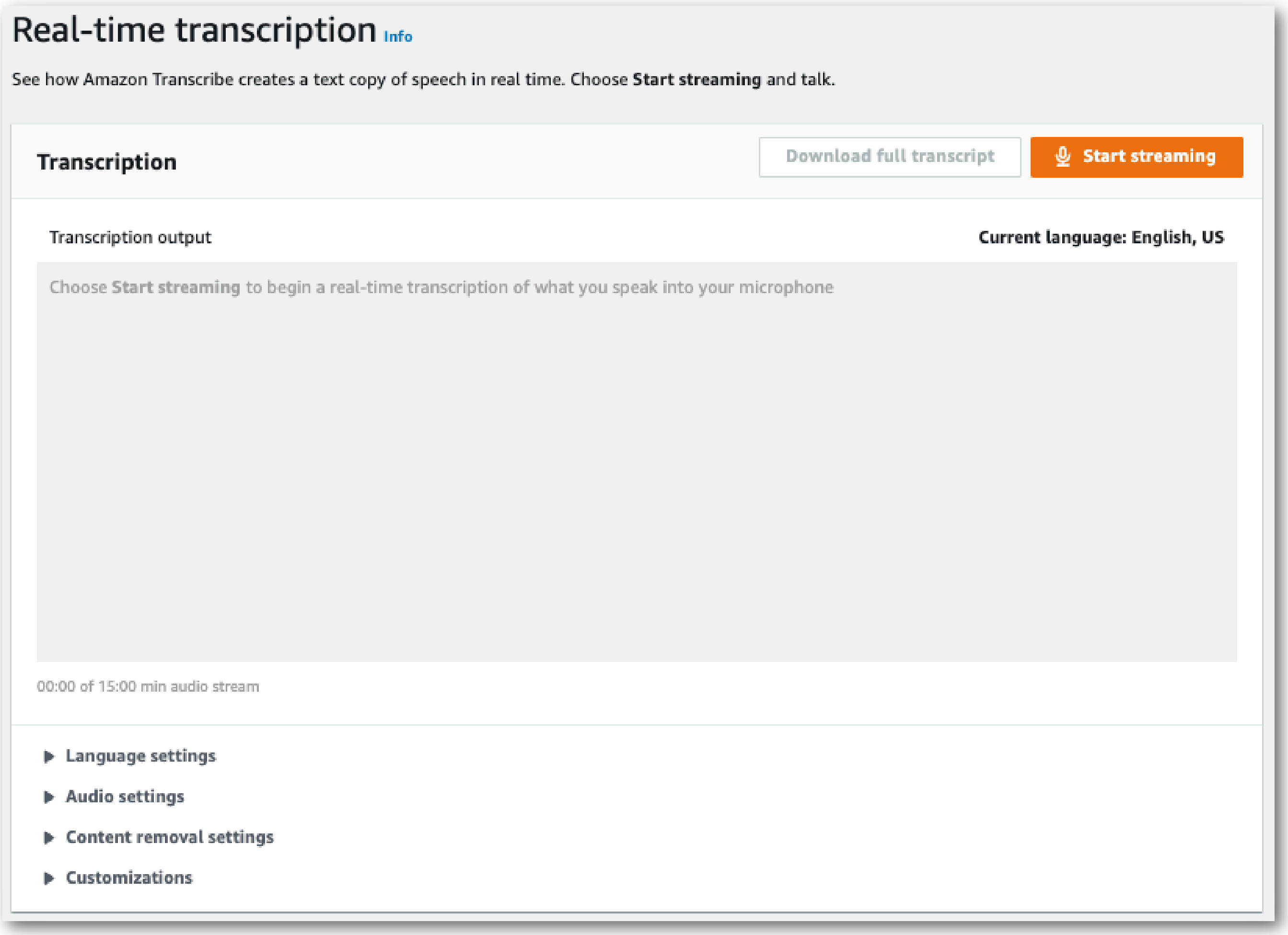
Using a custom vocabulary in a streaming transcription

To use a custom vocabulary with a streaming transcription, see the following for examples:

Amazon Web Services Management Console

1. Sign into the Amazon Web Services Management Console.

2. In the navigation pane, choose **Real-time transcription**. Scroll down to **Customizations** and expand this field if it is minimized.



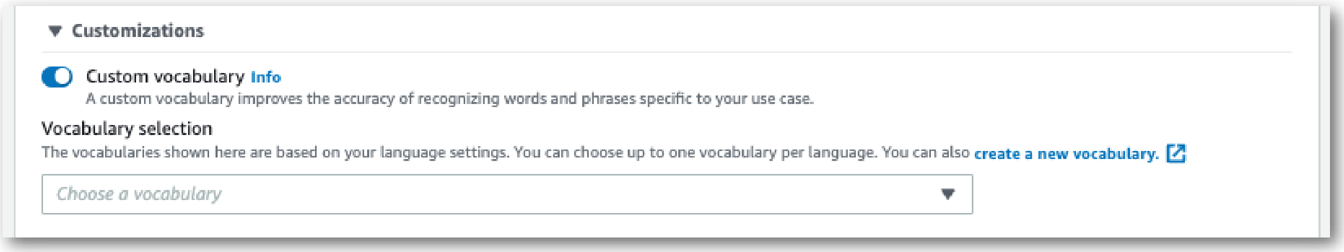
Real-time transcription Info

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

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

00:00 of 15:00 min audio stream

3. Toggle on **Custom vocabulary** and select a custom vocabulary from the dropdown menu.



A custom vocabulary improves the accuracy of recognizing words and phrases specific to your use case.

Vocabulary selection

The vocabularies shown here are based on your language settings. You can choose up to one vocabulary per language. You can also create a new vocabulary.

Include any other settings that you want to apply to your stream.

4. You're now ready to transcribe your stream. Select **Start streaming** and begin speaking. To end your dictation, select **Stop streaming**.

HTTP/2 stream

This example creates an HTTP/2 request that includes your custom vocabulary. For more information on using HTTP/2 streaming with Amazon Transcribe, see Setting up an HTTP/2 stream. For more detail on parameters and headers specific to Amazon Transcribe, see `StartStreamTranscription`.

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

Parameter definitions can be found in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

WebSocket stream

This example creates a presigned URL that applies your custom vocabulary to a WebSocket stream. Line breaks have been added for readability. For more information on using WebSocket streams with Amazon Transcribe, see Setting up a WebSocket stream. For more detail on parameters, see `StartStreamTranscription`.

```
GET wss://transcribestreaming.us-west-2.amazonaws.com:8443/stream-transcription-
websocket?
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220208%2Fus-
west-2%2Ftranscribe%2Faws4_request
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
```

```
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
&vocabulary-name=my-first-vocabulary
```

Parameter definitions can be found in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

Using custom vocabulary filters to delete, mask, or flag words

A custom vocabulary filter is a text file that contains a custom list of individual words that you want to modify in your transcription output.

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

Vocabulary filtering has three display methods: `mask`, `remove`, and `tag`. Refer to the following examples to see how each works.

- **Mask**: Replaces specified words with three asterisks (***).

```
"transcript": "You can specify a list of *** or *** words, and *** *** removes them
 from transcripts automatically."
```

- **Remove**: Deletes specified words, leaving nothing in their place.

```
"transcript": "You can specify a list of or words, and removes them from transcripts
 automatically."
```

- **Tag**: Adds a tag (`"vocabularyFilterMatch": true`) to each specified word, but doesn't alter the word itself. Tagging allows for rapid transcript substitutions and edits.

```
"transcript": "You can specify a list of profane or offensive words, and amazon
 transcribe removes them from transcripts automatically."
...
    "alternatives": [
        {
            "confidence": "1.0",
            "content": "profane"
        }
    ],
    "type": "pronunciation",
    "vocabularyFilterMatch": true
```

When you submit a transcription request, you can specify a custom vocabulary filter and the filtration method you want to apply. Amazon Transcribe then modifies exact word matches when they appear in your transcript, according to the filtration method you specify.

Custom vocabulary filters can be applied to batch and streaming transcription requests. To learn how to create a custom vocabulary filter, see Creating a vocabulary filter. To learn how to apply your custom vocabulary filter, see Using a custom vocabulary filter.

Note

Amazon Transcribe automatically masks racially sensitive terms, though you can opt out of this default filter by contacting Amazon Technical Support.

For a video walkthrough of vocabulary filtering, see Using vocabulary filters.

API operations specific to vocabulary filtering

`CreateVocabularyFilter`, `DeleteVocabularyFilter`, `GetVocabularyFilter`, `ListVocabularyFilters`, `UpdateVocabularyFilter`

Creating a vocabulary filter

There are two options for creating a custom vocabulary filter:

1. Save a list of line-separated words as a plain text file with UTF-8 encoding.
   - You can use this approach with the Amazon Web Services Management Console, Amazon CLI, or Amazon SDKs.
   - If using the Amazon Web Services Management Console, you can provide a local path or an Amazon S3 URI for your custom vocabulary file.
   - If using the Amazon CLI or Amazon SDKs, you must upload your custom vocabulary file to an Amazon S3 bucket and include the Amazon S3 URI in your request.
2. Include a list of comma-separated words directly in your API request.
   - You can use this approach with the Amazon CLI or Amazon SDKs using the `Words` parameter.

For examples of each method, refer to Creating custom vocabulary filters

Things to note when creating your custom vocabulary filter:

- Words aren't case sensitive. For example, "curse" and "CURSE" are treated the same.
- Only exact word matches are filtered. For example, if your filter includes "swear" but your media contains the word "swears" or "swearing", these are not filtered. Only instances of "swear" are filtered. You must therefore include all variations of the words you want filtered.
- Filters don't apply to words that are contained in other words. For example, if a custom vocabulary filter contains "marine" but not "submarine", "submarine" is not altered in the transcript.
- Each entry can only contain one word (no spaces).
- If you save your custom vocabulary filter as a text file, it must be in plain text format with UTF-8 encoding.
- You can have up to 100 custom vocabulary filters per Amazon Web Services account and each can be up to 50 Kb in size.
- You can only use characters that are supported for your language. Refer to your language's character set for details.

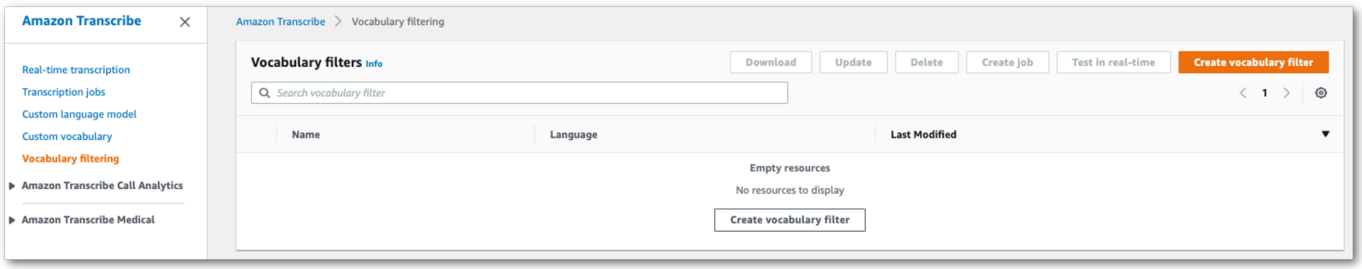
Creating custom vocabulary filters

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

Amazon Web Services Management Console

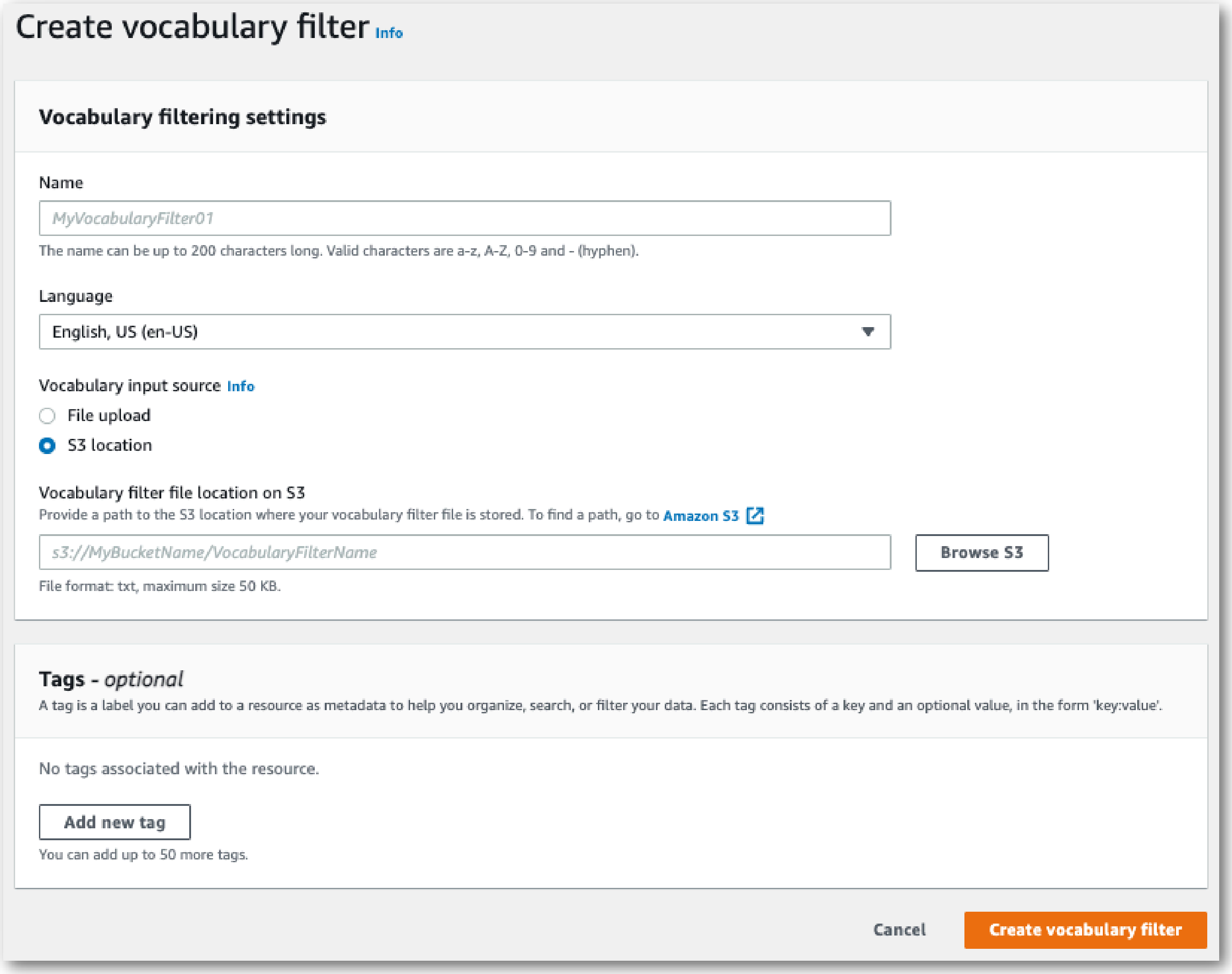
Before continuing, save your custom vocabulary filter as a text (*.txt) file. You can optionally upload your file to an Amazon S3 bucket.

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Vocabulary filtering**. This opens the **Vocabulary filters** page where you can view existing custom vocabulary filters or create a new one.
3. Select **Create vocabulary filter**.



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

Select the **File upload** or **S3 location** option under **Vocabulary input source**. Then specify the location of your custom vocabulary file.



Create vocabulary filter Info

Vocabulary filtering settings

Name

MyVocabularyFilter01

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

Language

English, US (en-US)

Vocabulary input source Info

File upload [✓ S3 location]

Vocabulary filter file location on S3

Provide a path to the S3 location where your vocabulary filter file is stored. To find a path, go to Amazon S3

s3://MyBucketName/VocabularyFilterName

File format: txt, maximum size 50 KB.

Tags - *optional*

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

No tags associated with the resource.

You can add up to 50 more tags.

4. Optionally, add tags to your custom vocabulary filter. Once you have all fields completed, select **Create vocabulary filter** at the bottom of the page. If there are no errors processing your file, this takes you back to the **Vocabulary filters** page.

   Your custom vocabulary filter is now ready to use.

Amazon CLI

This example uses the create-vocabulary-filter command to process a word list into a usable custom vocabulary filter. For more information, see `CreateVocabularyFilter`.

Option 1: You can include your list of words to your request using the `words` parameter.

```
aws transcribe create-vocabulary-filter \
--vocabulary-filter-name my-first-vocabulary-filter \
--language-code en-US \
--words profane,offensive,Amazon,Transcribe
```

Option 2: You can save your list of words as a text file and upload it to an Amazon S3 bucket, then include the file's URI in your request using the `vocabulary-filter-file-uri` parameter.

```
aws transcribe create-vocabulary-filter \
--vocabulary-filter-name my-first-vocabulary-filter \
--language-code en-US \
--vocabulary-filter-file-uri s3://amzn-s3-demo-bucket/my-vocabulary-filters/my-
vocabulary-filter.txt
```

Here's another example using the create-vocabulary-filter command, and a request body that creates your custom vocabulary filter.

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

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

Option 1: You can include your list of words to your request using the `Words` parameter.

```
{
  "VocabularyFilterName": "my-first-vocabulary-filter",
  "LanguageCode": "en-US",
  "Words": [
        "profane","offensive","Amazon","Transcribe"
  ]
}
```

Option 2: You can save your list of words as a text file and upload it to an Amazon S3 bucket, then include the file's URI in your request using the `VocabularyFilterFileUri` parameter.

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

Note

If you include `VocabularyFilterFileUri` in your request, you cannot use `Words`; you must choose one or the other.

Amazon SDK for Python (Boto3)

This example uses the Amazon SDK for Python (Boto3) to create a custom vocabulary filter using the create_vocabulary_filter method. For more information, see `CreateVocabularyFilter`.

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

Option 1: You can include your list of words to your request using the `Words` parameter.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
vocab_name = "my-first-vocabulary-filter"
response = transcribe.create_vocabulary_filter(
    LanguageCode = 'en-US',
    VocabularyFilterName = vocab_name,
    Words = [
        'profane','offensive','Amazon','Transcribe'
    ]
)
```

Option 2: You can save your list of words as a text file and upload it to an Amazon S3 bucket, then include the file's URI in your request using the `VocabularyFilterFileUri` parameter.

```
from __future__ import print_function
```

```
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
vocab_name = "my-first-vocabulary-filter"
response = transcribe.create_vocabulary_filter(
    LanguageCode = 'en-US',
    VocabularyFilterName = vocab_name,
    VocabularyFilterFileUri = 's3://amzn-s3-demo-bucket/my-vocabulary-filters/my-
vocabulary-filter.txt'
)
```

Note

If you include `VocabularyFilterFileUri` in your request, you cannot use `Words`; you must choose one or the other.

Note

If you create a new Amazon S3 bucket for your custom vocabulary filter files, make sure the IAM role making the `CreateVocabularyFilter` request has permissions to access this bucket. If the role doesn't have the correct permissions, your request fails. You can optionally specify an IAM role within your request by including the `DataAccessRoleArn` parameter. For more information on IAM roles and policies in Amazon Transcribe, see Amazon Transcribe identity-based policy examples.

Using a custom vocabulary filter

Once your custom vocabulary filter is created, you can include it in your transcription requests; refer to the following sections for examples.

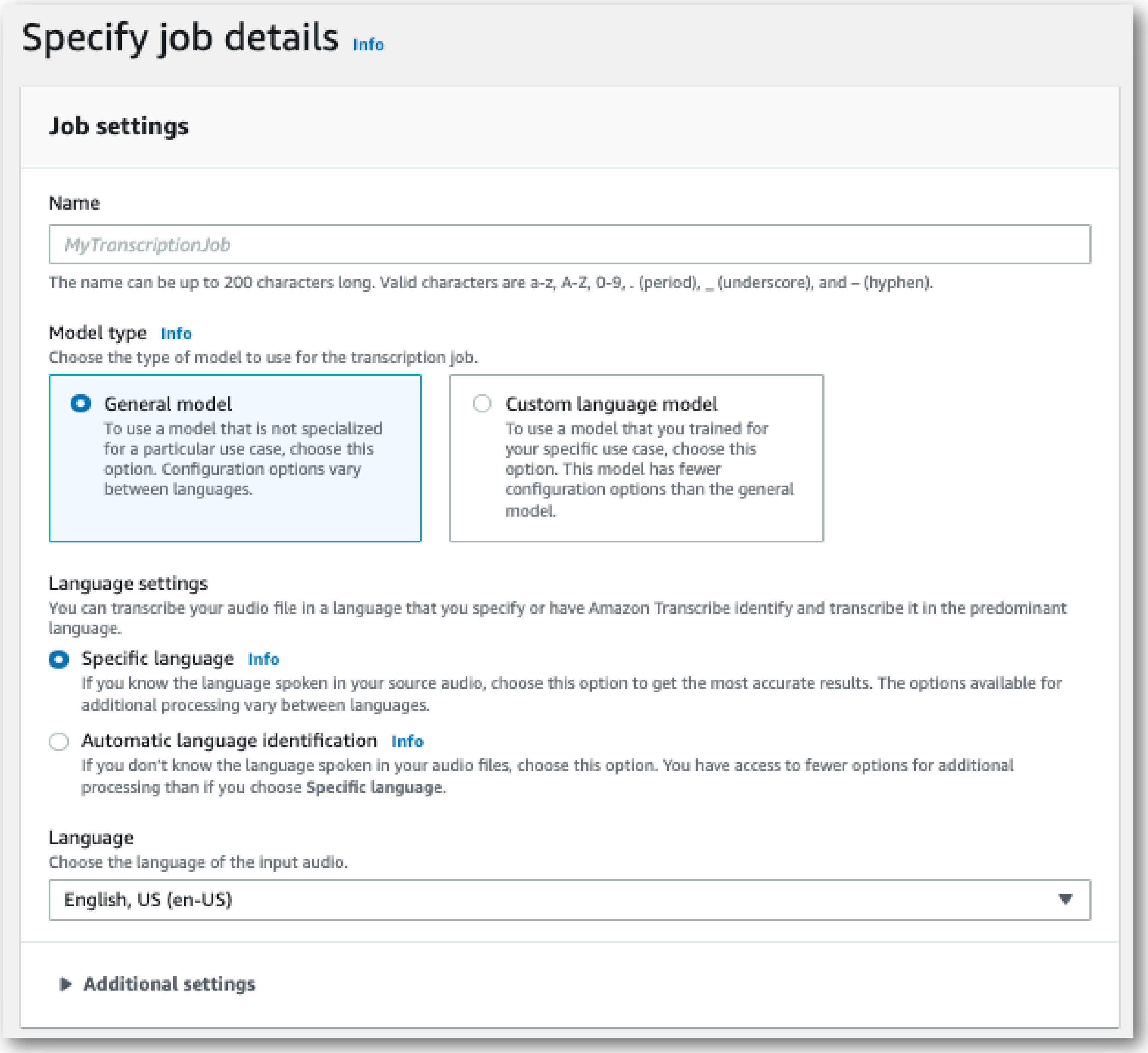
The language of the custom vocabulary filter you're including in your request must match the language code you specify for your media. If you use language identification and specify multiple language options, you can include one custom vocabulary filter per specified language. If the languages of your custom vocabulary filters don't match the language identified in your audio, your filters are not applied to your transcription and there are no warnings or errors.

Using a custom vocabulary filter in a batch transcription

To use a custom vocabulary filter with a batch transcription, see the following for examples:

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Transcription jobs**, then select **Create job** (top right). This opens the **Specify job details** page.



Specify job details Info

Job settings

Name

MyTranscriptionJob

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

Model type Info

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

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

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

Language settings

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

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

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

Language

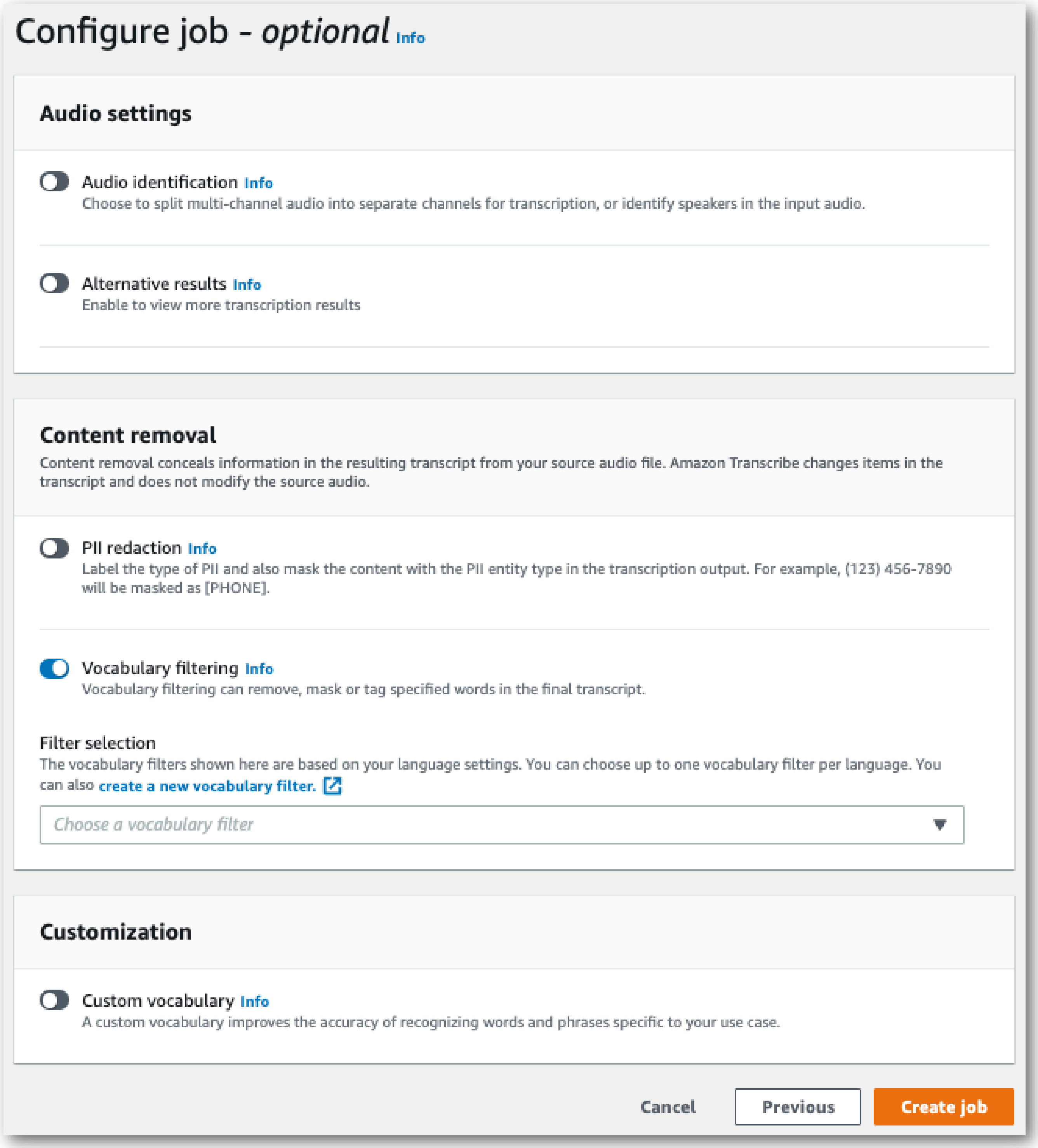
Choose the language of the input audio.

English, US (en-US)

▶ Additional settings

Name your job and specify your input media. Optionally include any other fields, then choose **Next**.

3. On the **Configure job** page, in the **Content removal** panel, toggle on **Vocabulary filtering**.



Configure job - *optional* Info

Audio settings

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

Alternative results Info
Enable to view more transcription results

Content removal
Content removal conceals information in the resulting transcript from your source audio file. Amazon Transcribe changes items in the transcript and does not modify the source audio.

PII redaction Info
Label the type of PII and also mask the content with the PII entity type in the transcription output. For example, (123) 456-7890 will be masked as [PHONE].

Vocabulary filtering Info
Vocabulary filtering can remove, mask or tag specified words in the final transcript.

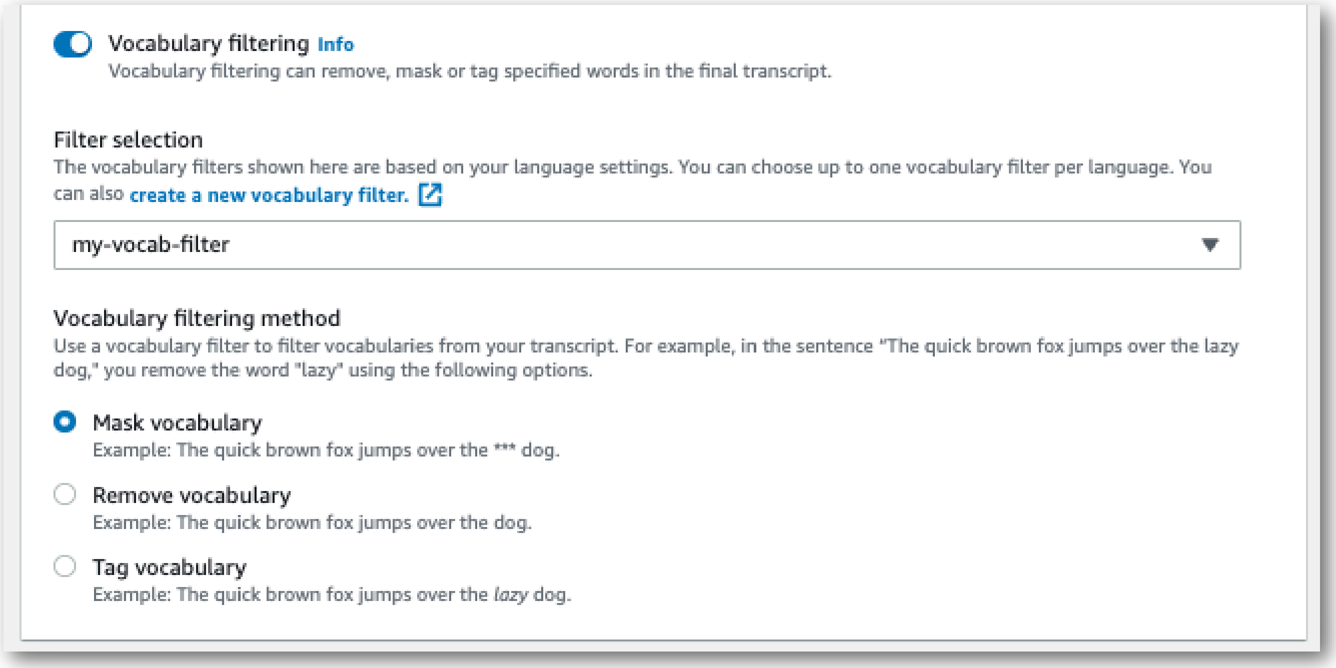
Filter selection
The vocabulary filters shown here are based on your language settings. You can choose up to one vocabulary filter per language. You can also create a new vocabulary filter.

Choose a vocabulary filter

Customization

Custom vocabulary Info
A custom vocabulary improves the accuracy of recognizing words and phrases specific to your use case.

4. Select your custom vocabulary filter from the dropdown menu and specify the filtration method.



Vocabulary filtering can remove, mask or tag specified words in the final transcript.

Filter selection

The vocabulary filters shown here are based on your language settings. You can choose up to one vocabulary filter per language. You can also create a new vocabulary filter.

my-vocab-filter

Vocabulary filtering method

Use a vocabulary filter to filter vocabularies from your transcript. For example, in the sentence "The quick brown fox jumps over the lazy dog," you remove the word "lazy" using the following options.

[✓ Mask vocabulary]
Example: The quick brown fox jumps over the *** dog.

Remove vocabulary
Example: The quick brown fox jumps over the dog.

Tag vocabulary
Example: The quick brown fox jumps over the *lazy* dog.

5. Select **Create job** to run your transcription job.

Amazon CLI

This example uses the start-transcription-job command and `Settings` parameter with the `VocabularyFilterName` and `VocabularyFilterMethod` sub-parameters. For more information, see `StartTranscriptionJob` and `Settings`.

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

Here's another example using the start-transcription-job command, and a request body that includes your custom vocabulary filter with that job.

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

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

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-bucket",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "Settings": {
        "VocabularyFilterName": "my-first-vocabulary-filter",
        "VocabularyFilterMethod": "mask"
   }
}
```

Amazon SDK for Python (Boto3)

This example uses the Amazon SDK for Python (Boto3) to include a custom vocabulary filter using the `Settings` argument for the start_transcription_job method. For more information, see `StartTranscriptionJob` and `Settings`.

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

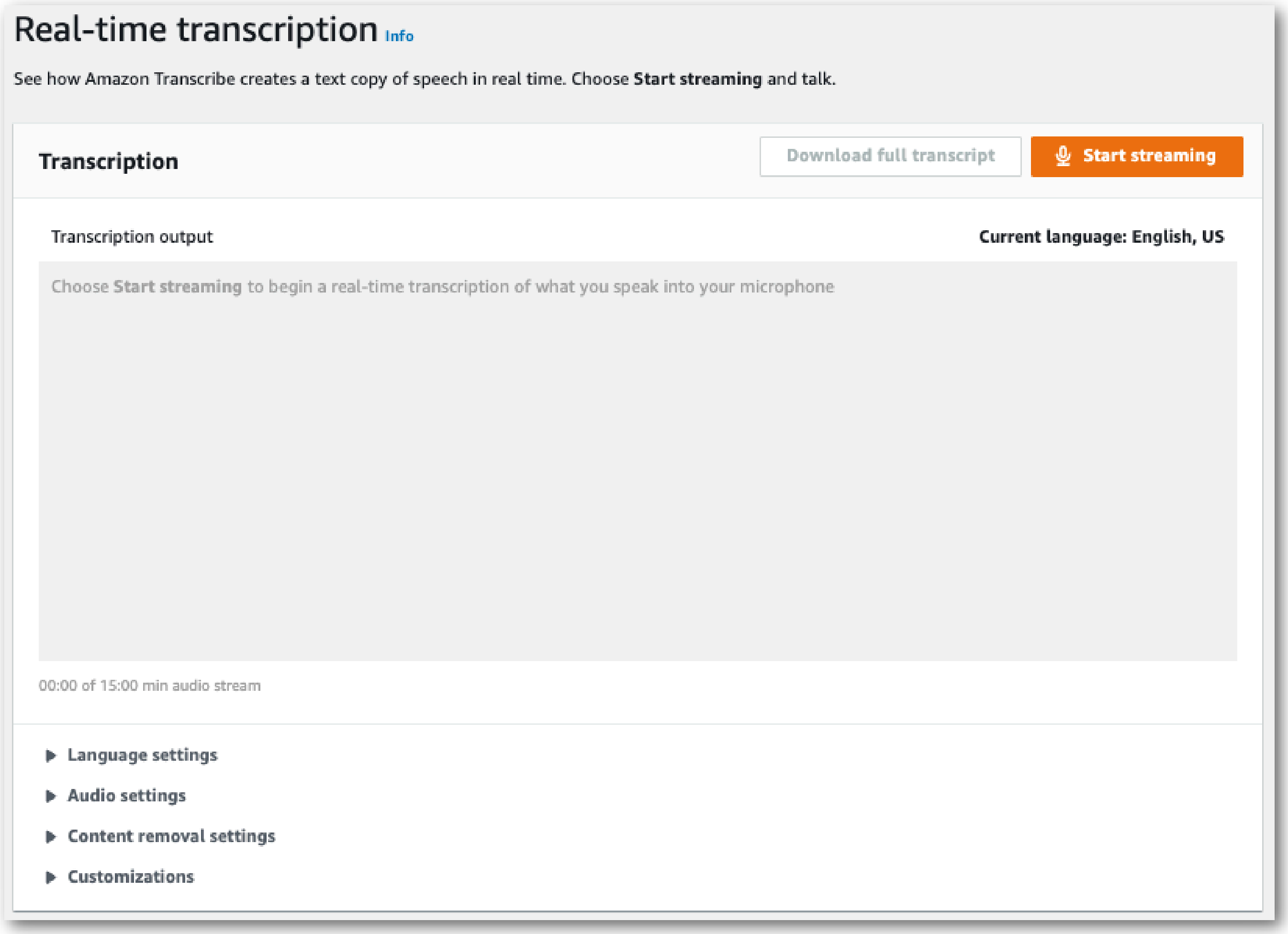
```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-bucket',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Settings = {
        'VocabularyFilterName': 'my-first-vocabulary-filter',
        'VocabularyFilterMethod': 'mask'
   }
)

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

Using a custom vocabulary filter in a streaming transcription

To use a custom vocabulary filter with a streaming transcription, see the following for examples:

Amazon Web Services Management Console

1. Sign into the Amazon Web Services Management Console.
2. In the navigation pane, choose **Real-time transcription**. Scroll down to **Content removal settings** and expand this field if it is minimized.


Real-time transcription Info

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

Transcription

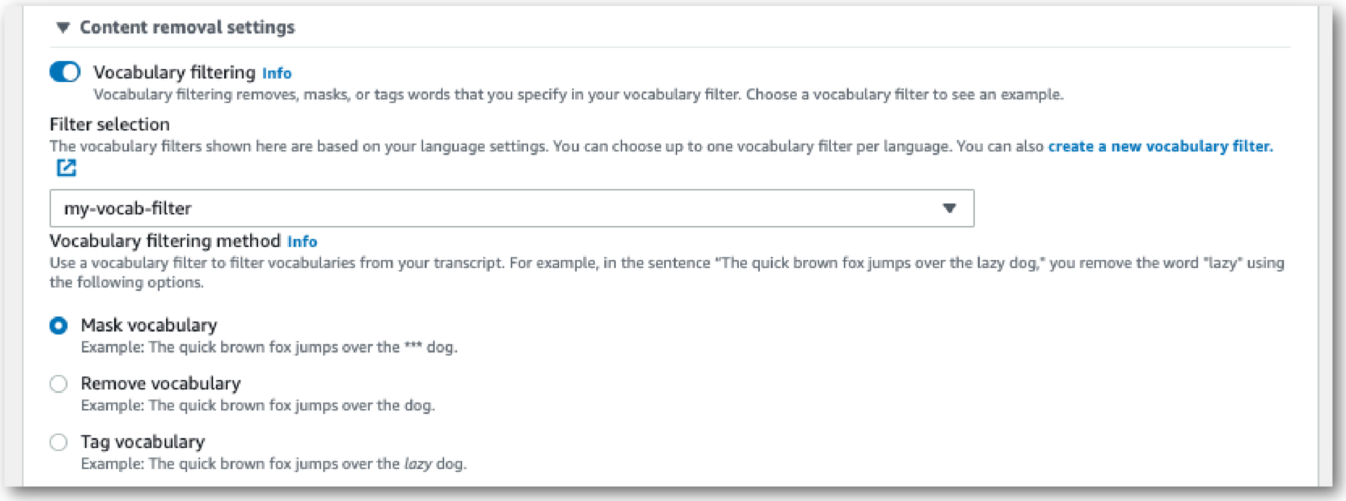
Transcription output

Current language: English, US

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

00:00 of 15:00 min audio stream

3. Toggle on **Vocabulary filtering**. Select a custom vocabulary filter from the dropdown menu and specify the filtration method.


Content removal settings

Vocabulary filtering Info

Vocabulary filtering removes, masks, or tags words that you specify in your vocabulary filter. Choose a vocabulary filter to see an example.

Filter selection

The vocabulary filters shown here are based on your language settings. You can choose up to one vocabulary filter per language. You can also create a new vocabulary filter.

my-vocab-filter

Vocabulary filtering method Info

Use a vocabulary filter to filter vocabularies from your transcript. For example, in the sentence "The quick brown fox jumps over the lazy dog," you remove the word "lazy" using the following options.

[✓ Mask vocabulary]
Example: The quick brown fox jumps over the *** dog.

Remove vocabulary
Example: The quick brown fox jumps over the dog.

Tag vocabulary
Example: The quick brown fox jumps over the *lazy* dog.

Include any other settings you want to apply to your stream.

4. You're now ready to transcribe your stream. Select **Start streaming** and begin speaking. To end your dictation, select **Stop streaming**.

HTTP/2 stream

This example creates an HTTP/2 request that includes your custom vocabulary filter and filter method. For more information on using HTTP/2 streaming with Amazon Transcribe, see Setting up an HTTP/2 stream. For more detail on parameters and headers specific to Amazon Transcribe, see `StartStreamTranscription`.

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

Parameter definitions can be found in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

WebSocket stream

This example creates a presigned URL that applies your custom vocabulary filter to a WebSocket stream. Line breaks have been added for readability. For more information on using WebSocket streams with Amazon Transcribe, see Setting up a WebSocket stream. For more detail on parameters, see `StartStreamTranscription`.

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

```
&X-Amz-Date=20220208T235959Z
&X-Amz-Expires=300
&X-Amz-Security-Token=security-token
&X-Amz-Signature=string
&X-Amz-SignedHeaders=content-type%3Bhost%3Bx-amz-date
&language-code=en-US
&media-encoding=flac
&sample-rate=16000
&vocabulary-filter-name=my-first-vocabulary-filter
&vocabulary-filter-method=mask
```

Parameter definitions can be found in the API Reference; parameters common to all Amazon API operations are listed in the Common Parameters section.

Detecting toxic speech

Toxic speech detection is designed to help moderate social media platforms that involve peer-to-peer dialogue, such as online gaming and social chat platforms. The use of toxic speech can be deeply detrimental to individuals, peer groups, and communities. Flagging harmful language helps organizations keep conversations civil and maintain a safe and inclusive online environment for users to create, share and participate freely.

Amazon Transcribe Toxicity Detection leverages both audio and text-based cues to identify and classify voice-based toxic content across seven categories including sexual harassment, hate speech, threat, abuse, profanity, insult, and graphic. In addition to text, Amazon Transcribe Toxicity Detection uses speech cues, such as tones and pitch to hone in on toxic intent in speech. This is an improvement from standard content moderation systems that are designed to focus only on specific terms, without accounting for intention.

Amazon Transcribe flags and categorizes toxic speech, which minimizes the volume of data that must be manually processed. This enables content moderators to quickly and efficiently manage the discourse on their platforms.

Toxic speech categories include:

- **Profanity**: Speech that contains words, phrases, or acronyms that are impolite, vulgar, or offensive.
- **Hate speech**: Speech that criticizes, insults, denounces, or dehumanizes a person or group on the basis of an identity (such as race, ethnicity, gender, religion, sexual orientation, ability, and national origin).
- **Sexual**: Speech that indicates sexual interest, activity, or arousal using direct or indirect references to body parts, physical traits, or sex.
- **Insults**: Speech that includes demeaning, humiliating, mocking, insulting, or belittling language. This type of language is also labeled as bullying.
- **Violence or threat**: Speech that includes threats seeking to inflict pain, injury, or hostility toward a person or group.
- **Graphic**: Speech that uses visually descriptive and unpleasantly vivid imagery. This type of language is often intentionally verbose to amplify a recipient's discomfort.
- **Harassment or abusive**: Speech intended to affect the psychological well-being of the recipient, including demeaning and objectifying terms. This type of language is also labeled as harassment.

Toxicity detection analyzes speech segments (the speech between natural pauses) and assigns confidence scores to these segments. Confidence scores are values between 0 and 1. A larger confidence score indicates a greater likelihood that the content is toxic speech in the associated category. You can use these confidence scores to set the appropriate threshold of toxicity detection for your use case.

Note

Toxicity detection is only available for batch transcriptions in US English (`en-US`).

View example output in JSON format.

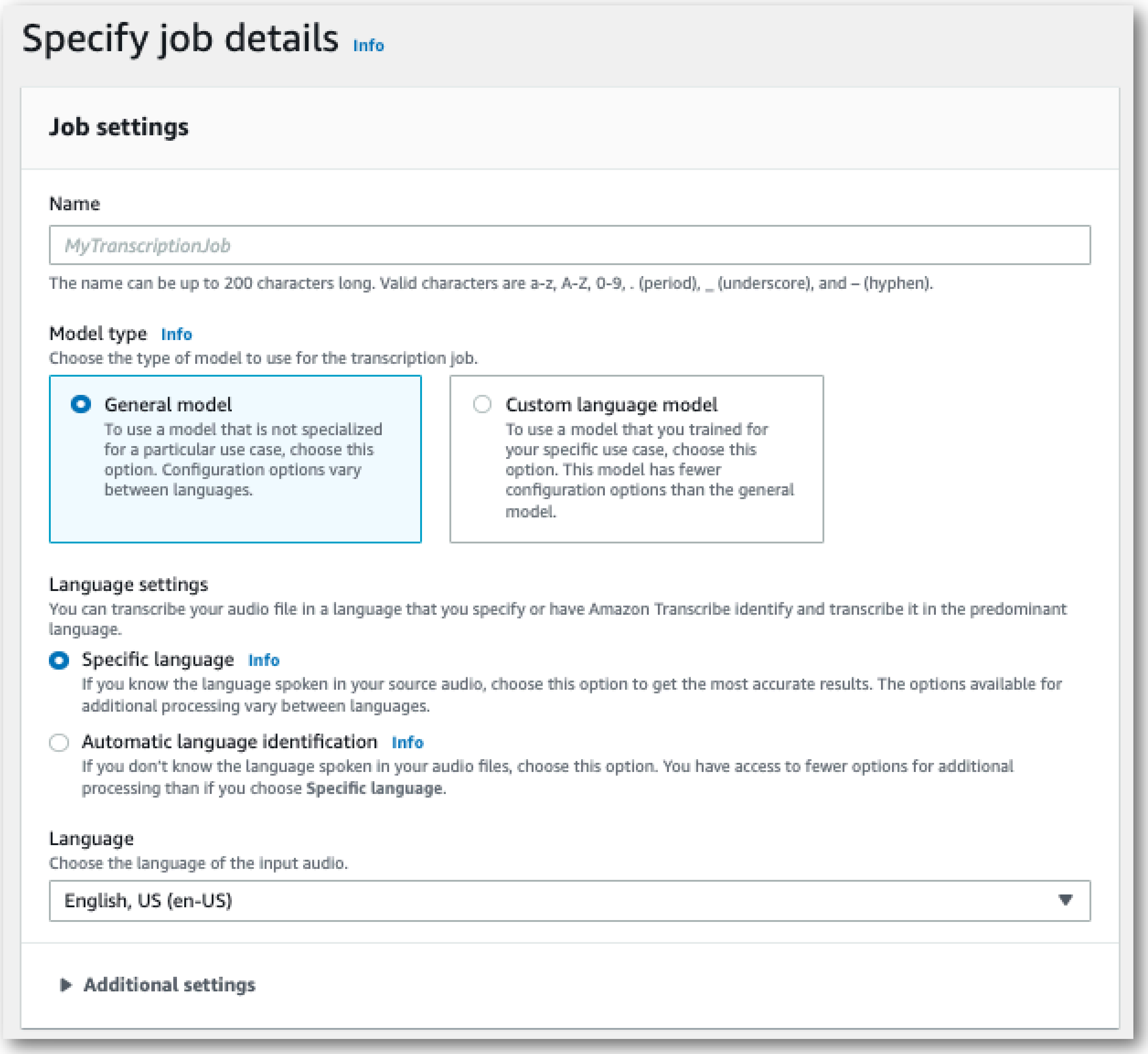
Using toxic speech detection

Using toxic speech detection in a batch transcription

To use toxic speech detection with a batch transcription, see the following for examples:

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Transcription jobs**, then select **Create job** (top right). This opens the **Specify job details** page.



Specify job details Info

Job settings

Name

MyTranscriptionJob

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

Model type Info

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

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

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

Language settings

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

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

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

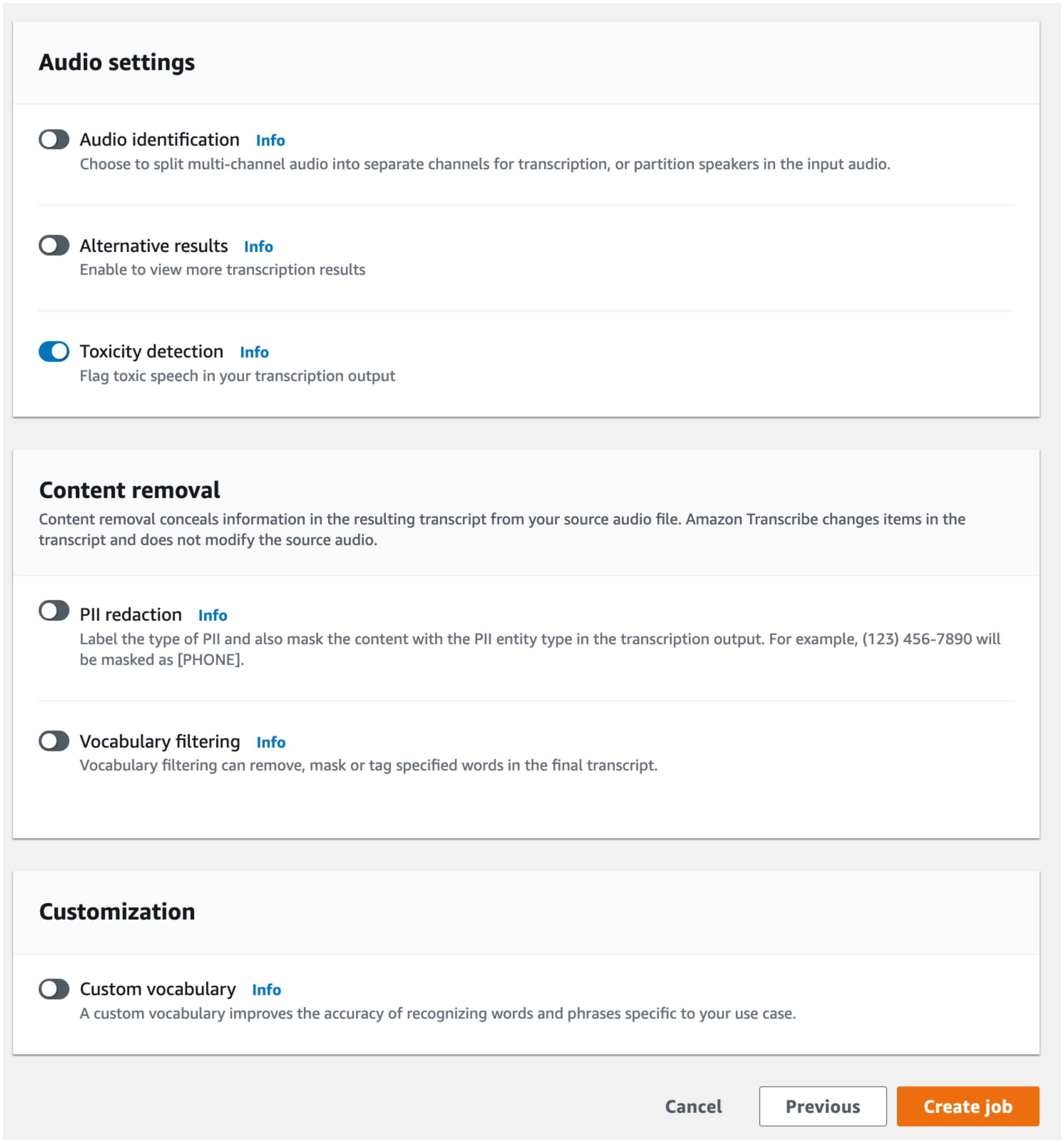
Language

Choose the language of the input audio.

English, US (en-US)

▶ Additional settings

3. On the **Specify job details** page, you can also enable PII redaction if you want. Note that the other listed options are not supported with Toxicity detection. Select **Next**. This takes you to the **Configure job - optional** page. In the **Audio settings** panel, select **Toxicity detection**.



Audio settings

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

Alternative results Info
Enable to view more transcription results

Toxicity detection Info
Flag toxic speech in your transcription output

Content removal
Content removal conceals information in the resulting transcript from your source audio file. Amazon Transcribe changes items in the transcript and does not modify the source audio.

PII redaction Info
Label the type of PII and also mask the content with the PII entity type in the transcription output. For example, (123) 456-7890 will be masked as [PHONE].

Vocabulary filtering Info
Vocabulary filtering can remove, mask or tag specified words in the final transcript.

Customization

Custom vocabulary Info
A custom vocabulary improves the accuracy of recognizing words and phrases specific to your use case.

4. Select **Create job** to run your transcription job.
5. Once your transcription job is complete, you can download your transcript from the **Download** drop-down menu in the transcription job's detail page.

Amazon CLI

This example uses the start-transcription-job command and `ToxicityDetection` parameter. For more information, see `StartTranscriptionJob` and `ToxicityDetection`.

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

Here's another example using the start-transcription-job command, and a request body that includes toxicity detection.

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

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

```
{
  "TranscriptionJobName": "my-first-transcription-job",
  "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
  },
  "OutputBucketName": "amzn-s3-demo-bucket",
  "OutputKey": "my-output-files/",
  "LanguageCode": "en-US",
  "ToxicityDetection": [
      {
         "ToxicityCategories": [ "ALL" ]
      }
   ]
}
```

Amazon SDK for Python (Boto3)

This example uses the Amazon SDK for Python (Boto3) to enable `ToxicityDetection` for the start_transcription_job method. For more information, see `StartTranscriptionJob` and `ToxicityDetection`.

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

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

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

Example output

Toxic speech is tagged and categorized in your transcription output. Each instance of toxic speech is categorized and assigned a confidence score (a value between 0 and 1). A larger confidence value indicates a greater likelihood that the content is toxic speech within the specified category.

Example output (JSON)

The following is an example output in JSON format showing categorized toxic speech with associated confidence scores.

```
{
    "jobName": "my-toxicity-job",
    "accountId": "111122223333",
    "results": {
        "transcripts": [...],
        "items":[...],
        "toxicity_detection": [
            {
                "text": "What the * are you doing man? That's why I didn't want to play 
 with your * .  man it was a no, no I'm not calming down * man. I well I spent I spent 
 too much * money on this game.",
                "toxicity": 0.7638,
                "categories": {
                    "profanity": 0.9913,
                    "hate_speech": 0.0382,
                    "sexual": 0.0016,
                    "insult": 0.6572,
                    "violence_or_threat": 0.0024,
                    "graphic": 0.0013,
                    "harassment_or_abuse": 0.0249
                },
                "start_time": 8.92,
                "end_time": 21.45
            },
            Items removed for brevity
            {
                "text": "What? Who? What the * did you just say to me? What's your 
 address? What is your * address? I will pull up right now on your * * man. Take your * 
 back to , tired of this **.",
                "toxicity": 0.9816,
                "categories": {
                    "profanity": 0.9865,
                    "hate_speech": 0.9123,
                    "sexual": 0.0037,
                    "insult": 0.5447,
                    "violence_or_threat": 0.5078,
                    "graphic": 0.0037,
                    "harassment_or_abuse": 0.0613
                },
                "start_time": 43.459,
                "end_time": 54.639
            },
        ]
    },
    ...
    "status": "COMPLETED"
}
```

Creating video subtitles

Amazon Transcribe supports WebVTT (*.vtt) and SubRip (*.srt) output for use as video subtitles. You can select one or both file types when setting up your batch video transcription job. When using the subtitle feature, your selected subtitle file(s) and a regular transcript file (containing additional information) are produced. Subtitle and transcription files are output to the same destination.

Subtitles are displayed at the same time text is spoken, and remain visible until there is a natural pause or the speaker finishes talking. Note that if you enable subtitles in your transcription request and your audio contains no speech, a subtitle file is not created.

Important

Amazon Transcribe uses a default start index of `0` for subtitle output, which differs from the more widely used value of 1. If you require a start index of 1, you can specify this in the Amazon Web Services Management Console or in your API request using the `OutputStartIndex` parameter.

Using the incorrect start index can result in compatibility errors with other services, so be sure to verify which start index you require before creating your subtitles. If you're uncertain which value to use, we recommend choosing 1. Refer to `Subtitles` for more information.

Features supported with subtitles:

- **Content redaction** — Any redacted content is reflected as '`PII`' in both your subtitle and regular transcript output files. The audio is not altered.
- **Vocabulary filters** — Subtitle files are generated from the transcription file, so any words you filter in your standard transcription output are also filtered in your subtitles. Filtered content is displayed as whitespace or `***` in your transcript and subtitle files. The audio is not altered.
- **Speaker diarization** — If there are multiple speakers in a given subtitle segment, dashes are used to distinguish each speaker. This applies to both WebVTT and SubRip formats; for example:
  - -- Text spoken by Person 1
  - -- Text spoken by Person 2

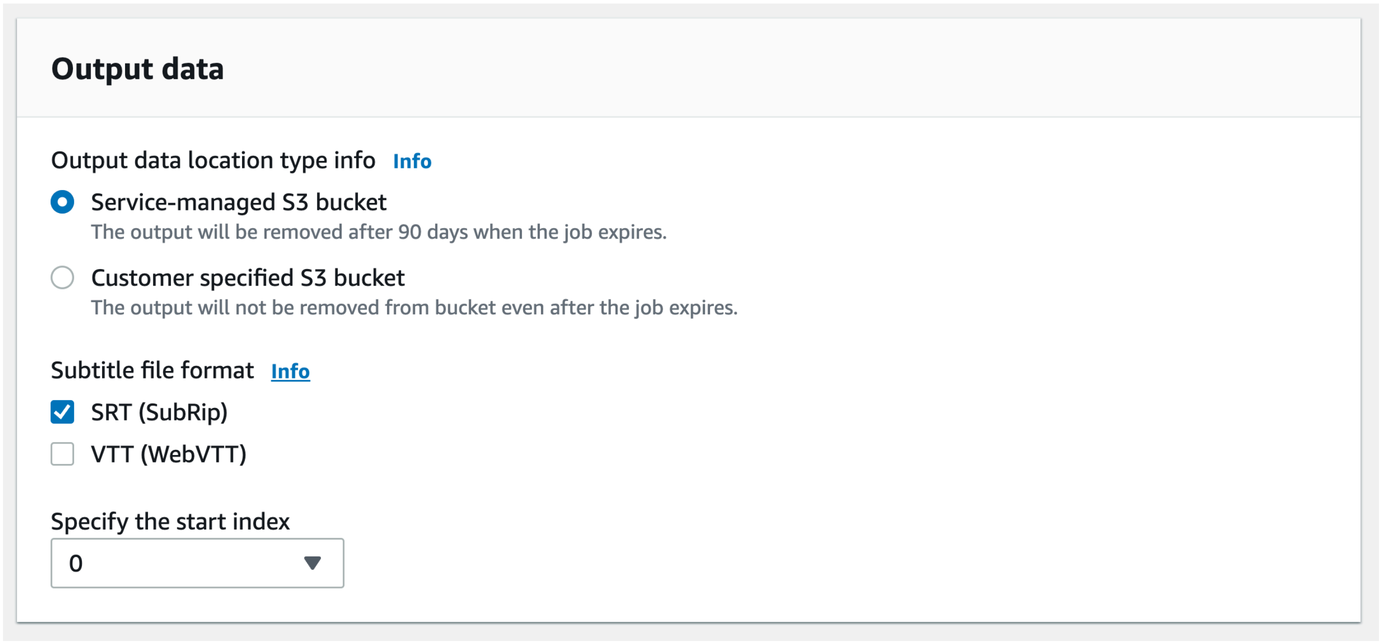
Subtitle files are stored in the same Amazon S3 location as your transcription output.

Generating subtitle files

You can create subtitle files using the **Amazon Web Services Management Console**, **Amazon CLI**, or **Amazon SDKs**; see the following examples:

Amazon Web Services Management Console

1. Sign in to the Amazon Web Services Management Console.
2. In the navigation pane, choose **Transcription jobs**, then select **Create job** (top right). This opens the **Specify job details** page. Subtitle options are located in the **Output data** panel.
3. Select the formats you want for your subtitles files, then choose a value for your start index. Note that the Amazon Transcribe default is `0`, but `1` is more widely used. If you're uncertain which value to use, we recommend choosing `1`, as this may improve compatibility with other services.



Output data

Output data location type info Info

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

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

Subtitle file format Info

☑ SRT (SubRip)
☐ VTT (WebVTT)

Specify the start index
0

4. Fill in any other fields you want to include on the **Specify job details** page, then select **Next**. This takes you to the **Configure job - *optional* page**.
5. Select **Create job** to run your transcription job.

Amazon CLI

This example uses the start-transcription-job command and `Subtitles` parameter. For more information, see `StartTranscriptionJob` and `Subtitles`.

```
aws transcribe start-transcription-job \
--region us-west-2 \
--transcription-job-name my-first-transcription-job \
--media MediaFileUri=s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac \
--output-bucket-name amzn-s3-demo-bucket \
--output-key my-output-files/ \
--language-code en-US \
--subtitles Formats=vtt,srt,OutputStartIndex=1
```

Here's another example using the start-transcription-job command, and a request body that adds subtitles to that job.

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

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

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

Amazon SDK for Python (Boto3)

This example uses the Amazon SDK for Python (Boto3) to add subtitles using the `Subtitles` argument for the start_transcription_job method. For more information, see `StartTranscriptionJob` and `Subtitles`.

For additional examples using the Amazon SDKs, including feature-specific, scenario, and cross-service examples, refer to the Code examples for Amazon Transcribe using Amazon SDKs chapter.

```
from __future__ import print_function
import time
import boto3
transcribe = boto3.client('transcribe', 'us-west-2')
job_name = "my-first-transcription-job"
job_uri = "s3://amzn-s3-demo-bucket/my-input-files/my-media-file.flac"
transcribe.start_transcription_job(
    TranscriptionJobName = job_name,
    Media = {
        'MediaFileUri': job_uri
    },
    OutputBucketName = 'amzn-s3-demo-bucket',
    OutputKey = 'my-output-files/',
    LanguageCode = 'en-US',
    Subtitles = {
        'Formats': [
            'vtt','srt'
        ],
        'OutputStartIndex': 1
   }
)

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

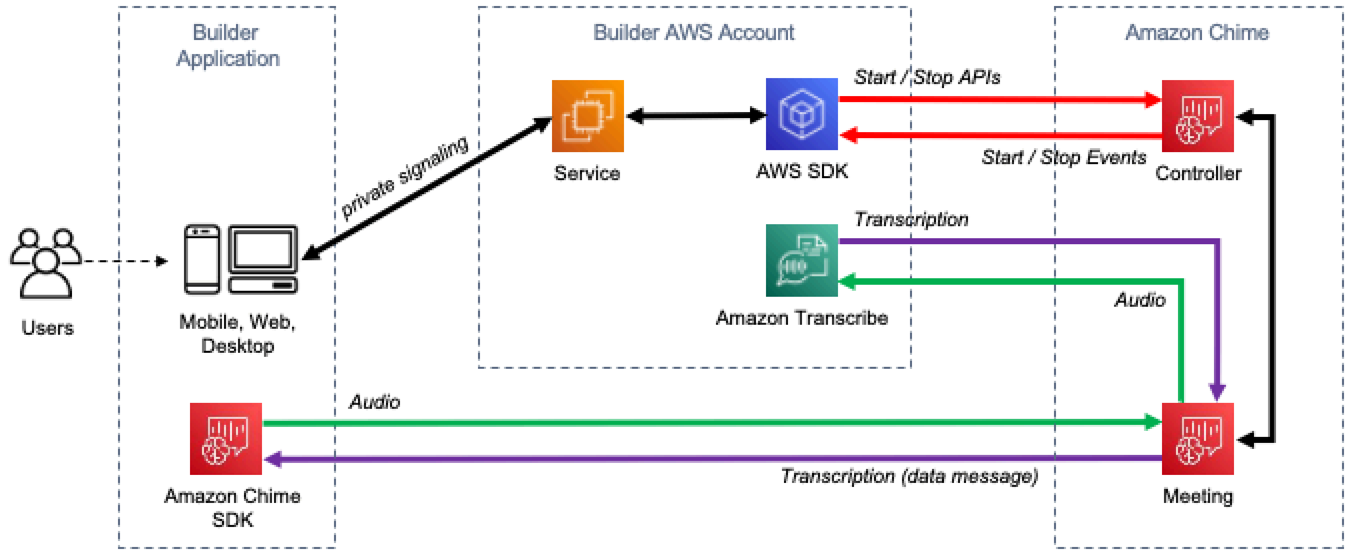
Transcribing your Amazon Chime calls in real time

Amazon Transcribe is integrated with the Amazon Chime SDK, facilitating real-time transcriptions of your Amazon Chime calls.

When you request a transcription using the Amazon Chime SDK API, Amazon Chime begins streaming audio to Amazon Transcribe and continues to do so for the duration of the call.

The Amazon Chime SDK uses its 'active talker' algorithm to select the top two active talkers, and then sends their audio to Amazon Transcribe as two separate channels via a single stream. Meeting participants receive user-attributed transcriptions via Amazon Chime SDK data messages. You can view delivery examples in the *Amazon Chime SDK Developer Guide*.

The data flow of an Amazon Chime transcription is depicted in the following diagram:



For additional information and detailed instructions on how to set up real-time Amazon Chime transcriptions, refer to Using Amazon Chime SDK live transcription in the *Amazon Chime SDK Developer Guide*. For API operations, refer to the *Amazon Chime SDK API Reference*.

Code examples for Amazon Transcribe using Amazon SDKs

The following code examples show how to use Amazon Transcribe with an Amazon software development kit (SDK).

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other Amazon Web Services services.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Code examples

- Convert text to speech and back to text using an Amazon SDK
- Create and refine an Amazon Transcribe custom vocabulary using an Amazon SDK
- Transcribe audio and get job data with Amazon Transcribe using an Amazon SDK

Basic examples for Amazon Transcribe using Amazon SDKs

The following code examples show how to use the basics of Amazon Transcribe with Amazon SDKs.

Examples

Actions for Amazon Transcribe using Amazon SDKs

The following code examples demonstrate how to perform individual Amazon Transcribe actions with Amazon SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

These excerpts call the Amazon Transcribe API and are code excerpts from larger programs that must be run in context. You can see actions in context in Scenarios for Amazon Transcribe using Amazon SDKs .

The following examples include only the most commonly used actions. For a complete list, see the Amazon Transcribe API Reference.

Examples

- Use CreateVocabulary with an Amazon SDK or CLI
- Use DeleteMedicalTranscriptionJob with an Amazon SDK or CLI
- Use DeleteTranscriptionJob with an Amazon SDK or CLI
- Use DeleteVocabulary with an Amazon SDK or CLI
- Use GetTranscriptionJob with an Amazon SDK or CLI
- Use GetVocabulary with an Amazon SDK or CLI
- Use ListMedicalTranscriptionJobs with an Amazon SDK or CLI
- Use ListTranscriptionJobs with an Amazon SDK or CLI
- Use ListVocabularies with an Amazon SDK or CLI
- Use StartMedicalTranscriptionJob with an Amazon SDK or CLI
- Use StartTranscriptionJob with an Amazon SDK or CLI
- Use UpdateVocabulary with an Amazon SDK or CLI

Use `CreateVocabulary` with an Amazon SDK or CLI

The following code examples show how to use `CreateVocabulary`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Create and refine a custom vocabulary

.NET

Amazon SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// Create a custom vocabulary using a list of phrases. Custom vocabularies
    /// improve transcription accuracy for one or more specific words.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> CreateCustomVocabulary(LanguageCode
 languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.CreateVocabularyAsync(
            new CreateVocabularyRequest
            {
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- For API details, see CreateVocabulary in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To create a custom vocabulary

The following `create-vocabulary` example creates a custom vocabulary. To create a custom vocabulary, you must have created a text file with all the terms that you want to transcribe more accurately. For vocabulary-file-uri, specify the Amazon Simple Storage Service (Amazon S3) URI of that text file. For language-code, specify a language code corresponding to the language of your custom vocabulary. For vocabulary-name, specify what you want to call your custom vocabulary.

```
aws transcribe create-vocabulary \
    --language-code language-code \
    --vocabulary-name cli-vocab-example \
    --vocabulary-file-uri s3://amzn-s3-demo-bucket/Amazon-S3-prefix/the-text-
file-for-the-custom-vocabulary.txt
```

Output:

```
{
    "VocabularyName": "cli-vocab-example",
    "LanguageCode": "language-code",
    "VocabularyState": "PENDING"
}
```

For more information, see Custom Vocabularies in the *Amazon Transcribe Developer Guide*.

- For API details, see CreateVocabulary in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary
 processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.",
vocabulary_name)
        raise
    else:
        return response
```

- For API details, see CreateVocabulary in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

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

- For API details, see CreateVocabulary in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteMedicalTranscriptionJob` with an Amazon SDK or CLI

The following code examples show how to use `DeleteMedicalTranscriptionJob`.

.NET

Amazon SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// Delete a medical transcription job. Also deletes the transcript
 associated with the job.
    /// </summary>
    /// <param name="jobName">Name of the medical transcription job to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMedicalTranscriptionJob(string jobName)
    {
        var response = await
 _amazonTranscribeService.DeleteMedicalTranscriptionJobAsync(
            new DeleteMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- For API details, see DeleteMedicalTranscriptionJob in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To delete a medical transcription job

The following `delete-medical-transcription-job` example deletes a medical transcription job.

```
aws transcribe delete-medical-transcription-job \
    --medical-transcription-job-name medical-transcription-job-name
```

This command produces no output.

For more information, see DeleteMedicalTranscriptionJob in the *Amazon Transcribe Developer Guide*.

- For API details, see DeleteMedicalTranscriptionJob in *Amazon CLI Command Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

Create the client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Delete a medical transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteMedicalTranscriptionJobCommand } from "@aws-sdk/client-
transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // For example,
 'medical_transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see Amazon SDK for JavaScript Developer Guide.
- For API details, see DeleteMedicalTranscriptionJob in *Amazon SDK for JavaScript API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteTranscriptionJob` with an Amazon SDK or CLI

The following code examples show how to use `DeleteTranscriptionJob`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Create and refine a custom vocabulary

.NET

Amazon SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// Delete a transcription job. Also deletes the transcript associated with
 the job.
    /// </summary>
    /// <param name="jobName">Name of the transcription job to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTranscriptionJob(string jobName)
    {
        var response = await
 _amazonTranscribeService.DeleteTranscriptionJobAsync(
            new DeleteTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- For API details, see DeleteTranscriptionJob in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To delete one of your transcription jobs

The following `delete-transcription-job` example deletes one of your transcription jobs.

```
aws transcribe delete-transcription-job \
    --transcription-job-name your-transcription-job
```

This command produces no output.

For more information, see DeleteTranscriptionJob in the *Amazon Transcribe Developer Guide*.

- For API details, see DeleteTranscriptionJob in *Amazon CLI Command Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

Delete a transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME", // Required. For example, 'transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Create the client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- For more information, see Amazon SDK for JavaScript Developer Guide.
- For API details, see DeleteTranscriptionJob in *Amazon SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise
```

- For API details, see DeleteTranscriptionJob in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

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

- For API details, see DeleteTranscriptionJob in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteVocabulary` with an Amazon SDK or CLI

The following code examples show how to use `DeleteVocabulary`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Create and refine a custom vocabulary

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
/// <summary>
/// Delete an existing custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.DeleteVocabularyAsync(
        new DeleteVocabularyRequest
        {
            VocabularyName = vocabularyName
        });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- For API details, see DeleteVocabulary in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To delete a custom vocabulary

The following `delete-vocabulary` example deletes a custom vocabulary.

```
aws transcribe delete-vocabulary \
    --vocabulary-name vocabulary-name
```

This command produces no output.

For more information, see Custom Vocabularies in the *Amazon Transcribe Developer Guide*.

- For API details, see DeleteVocabulary in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

- For API details, see DeleteVocabulary in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

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

- For API details, see DeleteVocabulary in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetTranscriptionJob` with an Amazon SDK or CLI

The following code examples show how to use `GetTranscriptionJob`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Create and refine a custom vocabulary

- Transcribe audio and get job data

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// Get details about a transcription job.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <returns>A TranscriptionJob instance with information on the requested
 job.</returns>
    public async Task<TranscriptionJob> GetTranscriptionJob(string jobName)
    {
        var response = await _amazonTranscribeService.GetTranscriptionJobAsync(
            new GetTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.TranscriptionJob;
    }
```

- For API details, see GetTranscriptionJob in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To get information about a specific transcription job

The following `get-transcription-job` example gets information about a specific transcription job. To access the transcription results, use the TranscriptFileUri parameter. Use the MediaFileUri parameter to see which audio file you transcribed with this job. You can use the Settings object to see the optional features you've enabled in the transcription job.

```
aws transcribe get-transcription-job \
    --transcription-job-name your-transcription-job
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "your-transcription-job",
        "TranscriptionJobStatus": "COMPLETED",
        "LanguageCode": "language-code",
        "MediaSampleRateHertz": 48000,
        "MediaFormat": "mp4",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.file-
extension"
        },
        "Transcript": {
            "TranscriptFileUri": "https://Amazon-S3-file-location-of-
transcription-output"
        },
        "StartTime": "2020-09-18T22:27:23.970000+00:00",
        "CreationTime": "2020-09-18T22:27:23.948000+00:00",
        "CompletionTime": "2020-09-18T22:28:21.197000+00:00",
        "Settings": {
            "ChannelIdentification": false,
            "ShowAlternatives": false
        },
        "IdentifyLanguage": true,
        "IdentifiedLanguageScore": 0.8672199249267578
    }
}
```

For more information, see Getting Started (Amazon Command Line Interface) in the *Amazon Transcribe Developer Guide*.

- For API details, see GetTranscriptionJob in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job
```

- For API details, see GetTranscriptionJob in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

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

- For API details, see GetTranscriptionJob in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetVocabulary` with an Amazon SDK or CLI

The following code examples show how to use `GetVocabulary`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Create and refine a custom vocabulary

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// Get information about a custom vocabulary.
    /// </summary>
    /// <param name="vocabularyName">Name of the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> GetCustomVocabulary(string vocabularyName)
    {
        var response = await _amazonTranscribeService.GetVocabularyAsync(
            new GetVocabularyRequest()
            {
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- For API details, see GetVocabulary in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To get information about a custom vocabulary

The following `get-vocabulary` example gets information on a previously created custom vocabulary.

```
aws transcribe get-vocabulary \
    --vocabulary-name cli-vocab-1
```

Output:

```
{
    "VocabularyName": "cli-vocab-1",
    "LanguageCode": "language-code",
    "VocabularyState": "READY",
    "LastModifiedTime": "2020-09-19T23:22:32.836000+00:00",
    "DownloadUri": "https://link-to-download-the-text-file-used-to-create-your-
custom-vocabulary"
}
```

For more information, see Custom Vocabularies in the *Amazon Transcribe Developer Guide*.

- For API details, see GetVocabulary in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response =
 transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

```

```

- For API details, see GetVocabulary in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

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

- For API details, see GetVocabulary in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListMedicalTranscriptionJobs` with an Amazon SDK or CLI

The following code examples show how to use `ListMedicalTranscriptionJobs`.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// List medical transcription jobs, optionally with a name filter.
    /// </summary>
    /// <param name="jobNameContains">Optional name filter for the medical 
 transcription jobs.</param>
    /// <returns>A list of summaries about medical transcription jobs.</returns>
    public async Task<List<MedicalTranscriptionJobSummary>> 
 ListMedicalTranscriptionJobs(
        string? jobNameContains = null)
    {
        var response = await 
 _amazonTranscribeService.ListMedicalTranscriptionJobsAsync(
            new ListMedicalTranscriptionJobsRequest()
            {
                JobNameContains = jobNameContains
            });
        return response.MedicalTranscriptionJobSummaries;
    }
```

- For API details, see ListMedicalTranscriptionJobs in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To list your medical transcription jobs

The following `list-medical-transcription-jobs` example lists the medical transcription jobs associated with your Amazon account and Region. To get more information about a particular transcription job, copy the value of a MedicalTranscriptionJobName parameter in the transcription output, and specify that value for the `MedicalTranscriptionJobName` option of the `get-medical-transcription-job` command. To see more of your transcription jobs, copy the value of the NextToken parameter, run the `list-medical-transcription-jobs` command again, and specify that value in the `--next-token` option.

```
aws transcribe list-medical-transcription-jobs
```

Output:

```
{
    "NextToken": "3/PblzkiGhzjER3KHuQt2fmbPLF7cDYafjFMEoGn44ON/
gsuUSTIkGyanvRE6WMXFd/ZTEc2EZj+P9eii/
z1O2FDYli6RLI0WoRX4RwMisVrh9G0Kie0Y8ikBCdtqlZB10Wa9McC+ebOl
+LaDtZPC4u6ttoHLRlEfzqstHXSgapXg3tEBtm9piIaPB6MOM5BB6t86+qtmocTR/
qrteHZBBudhTfbCwhsxaqujHiiUvFdm3BQbKKWIW06yV9b+4f38oD2lVIan
+vfUs3gBYAl5VTDmXXzQPBQOHPjtwmFI+IWX15nSUjWuN3TUylHgPWzDaYT8qBtu0Z+3UG4V6b
+K2CC0XszXg5rBq9hYgNzy4XoFh/6s5DoSnzq49Q9xHgHdT2yBADFmvFK7myZBsj75+2vQZOSVpWUPy3WT/32zFAc
+mFYfUjtTZ8n/jq7aQEjQ42A
+X/7K6JgOcdVPtEg8PlDr5kgYYG3q3OmYXX37U3FZuJmnTI63VtIXsNnOU5eGoYObtpk00Nq9UkzgSJxqj84ZD5n
+S0EGy9ZUYBJRRcGeYUM3Q4DbSJfUwSAqcFdLIWZdp8qIREMQIBWy7BLwSdyqsQo2vRrd53hm5aWM7SVf6pPq6X/
IXR5+1eUOOD8/coaTT4ES2DerbV6RkV4o0VT1d0SdVX/
MmtkNG8nYj8PqU07w7988quh1ZP6D80veJS1q73tUUR9MjnGernW2tAnvnLNhdefBcD
+sZVfYq3iBMFY7wTy1P1G6NqW9GrYDYoX3tTPWlD7phpbVSyKrh/
PdYrps5UxnsGoA1b7L/FfAXDfUoGrGUB4N3JsPYXX9D++g+6gV1qBBs/
WfF934aKqfD6UTggm/zV3GAOWiBpfvAZRvEb924i6yGHyMC7y54O1ZAwSBupmI
+FFd13CaPO4kN1vJlth6aM5vUPXg4BpyUhtbRhwD/KxCvf9K0tLJGyL1A==",
    "MedicalTranscriptionJobSummaries": [
        {
            "MedicalTranscriptionJobName": "vocabulary-dictation-medical-
transcription-job",
            "CreationTime": "2020-09-21T21:17:27.016000+00:00",
            "StartTime": "2020-09-21T21:17:27.045000+00:00",
            "CompletionTime": "2020-09-21T21:17:59.561000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "DICTATION"
        },
        {
            "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
            "CreationTime": "2020-09-21T21:01:14.569000+00:00",
            "StartTime": "2020-09-21T21:01:14.592000+00:00",
            "CompletionTime": "2020-09-21T21:01:43.606000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "DICTATION"
        },
        {
            "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-21T19:09:18.171000+00:00",
            "StartTime": "2020-09-21T19:09:18.199000+00:00",
            "CompletionTime": "2020-09-21T19:10:22.516000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        },
        {
            "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-21T18:43:37.157000+00:00",
            "StartTime": "2020-09-21T18:43:37.265000+00:00",
            "CompletionTime": "2020-09-21T18:44:21.192000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        },
        {
            "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-20T23:46:44.053000+00:00",
            "StartTime": "2020-09-20T23:46:44.081000+00:00",
            "CompletionTime": "2020-09-20T23:47:35.851000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        }
    ]
}
```

For more information, see https://docs.aws.amazon.com/transcribe/latest/dg/batch-med-transcription.html> in the *Amazon Transcribe Developer Guide*.

- For API details, see ListMedicalTranscriptionJobs in *Amazon CLI Command Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

Create the client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

List medical transcription jobs.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and
 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the
 same region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see Amazon SDK for JavaScript Developer Guide.
- For API details, see ListMedicalTranscriptionJobs in *Amazon SDK for JavaScript API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListTranscriptionJobs` with an Amazon SDK or CLI

The following code examples show how to use `ListTranscriptionJobs`.

.NET

Amazon SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// List transcription jobs, optionally with a name filter.
    /// </summary>
    /// <param name="jobNameContains">Optional name filter for the transcription
 jobs.</param>
    /// <returns>A list of transcription job summaries.</returns>
    public async Task<List<TranscriptionJobSummary>>
 ListTranscriptionJobs(string? jobNameContains = null)
    {
        var response = await _amazonTranscribeService.ListTranscriptionJobsAsync(
            new ListTranscriptionJobsRequest()
            {
                JobNameContains = jobNameContains
            });
        return response.TranscriptionJobSummaries;
    }
```

- For API details, see ListTranscriptionJobs in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To list your transcription jobs

The following `list-transcription-jobs` example lists the transcription jobs associated with your Amazon account and Region.

```
aws transcribe list-transcription-jobs
```

Output:

```
{
    "NextToken": "NextToken",
    "TranscriptionJobSummaries": [
        {
            "TranscriptionJobName": "speak-id-job-1",
            "CreationTime": "2020-08-17T21:06:15.391000+00:00",
            "StartTime": "2020-08-17T21:06:15.416000+00:00",
            "CompletionTime": "2020-08-17T21:07:05.098000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "job-1",
            "CreationTime": "2020-08-17T20:50:24.207000+00:00",
            "StartTime": "2020-08-17T20:50:24.230000+00:00",
            "CompletionTime": "2020-08-17T20:52:18.737000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "sdk-test-job-4",
            "CreationTime": "2020-08-17T20:32:27.917000+00:00",
            "StartTime": "2020-08-17T20:32:27.956000+00:00",
            "CompletionTime": "2020-08-17T20:33:15.126000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "Diarization-speak-id",
            "CreationTime": "2020-08-10T22:10:09.066000+00:00",
            "StartTime": "2020-08-10T22:10:09.116000+00:00",
            "CompletionTime": "2020-08-10T22:26:48.172000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "your-transcription-job-name",
            "CreationTime": "2020-07-29T17:45:09.791000+00:00",
            "StartTime": "2020-07-29T17:45:09.826000+00:00",
            "CompletionTime": "2020-07-29T17:46:20.831000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        }
    ]
}
```

For more information, see Getting Started (Amazon Command Line Interface) in the *Amazon Transcribe Developer Guide*.

- For API details, see ListTranscriptionJobs in *Amazon CLI Command Reference*.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();

            listTranscriptionJobs(transcribeClient);
        }

        public static void listTranscriptionJobs(TranscribeClient
 transcribeClient) {
            ListTranscriptionJobsRequest listJobsRequest =
 ListTranscriptionJobsRequest.builder()
                .build();
 transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
                .flatMap(response ->
 response.transcriptionJobSummaries().stream())
                .forEach(jobSummary -> {
                    System.out.println("Job Name: " +
 jobSummary.transcriptionJobName());
                    System.out.println("Job Status: " +
 jobSummary.transcriptionJobStatus());
                    System.out.println("Output Location: " +
 jobSummary.outputLocationType());
                    // Add more information as needed

                    // Retrieve additional details for the job if necessary
                    GetTranscriptionJobResponse jobDetails =
 transcribeClient.getTranscriptionJob(
                        GetTranscriptionJobRequest.builder()

 .transcriptionJobName(jobSummary.transcriptionJobName())
                            .build());

                    // Display additional details
                    System.out.println("Language Code: " +
 jobDetails.transcriptionJob().languageCode());
                    System.out.println("Media Format: " +
 jobDetails.transcriptionJob().mediaFormat());
                    // Add more details as needed

                    System.out.println("--------------");
                });
        }
    }
```

- For API details, see [ListTranscriptionJobs]() in *Amazon SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

List transcription jobs.

```
// Import the required AWS SDK clients and commands for Node.js

import { ListTranscriptionJobsCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  JobNameContains: "KEYWORD", // Not required. Returns only transcription
  // job names containing this string
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new ListTranscriptionJobsCommand(params),
    );
    console.log("Success", data.TranscriptionJobSummaries);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Create the client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
```

```
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- For more information, see Amazon SDK for JavaScript Developer Guide.
- For API details, see ListTranscriptionJobs in *Amazon SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
def list_jobs(job_filter, transcribe_client):
    """
    Lists summaries of the transcription jobs for the current AWS account.

    :param job_filter: The list of returned jobs must contain this string in
 their
                       names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved transcription job summaries.
    """
    try:
        response =
 transcribe_client.list_transcription_jobs(JobNameContains=job_filter)
        jobs = response["TranscriptionJobSummaries"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_transcription_jobs(
                JobNameContains=job_filter, NextToken=next_token
            )
            jobs += response["TranscriptionJobSummaries"]
            next_token = response.get("NextToken")
        logger.info("Got %s jobs with filter %s.", len(jobs), job_filter)
    except ClientError:
        logger.exception("Couldn't get jobs with filter %s.", job_filter)
        raise
    else:
        return jobs
```

- For API details, see ListTranscriptionJobs in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

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

- For API details, see ListTranscriptionJobs in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListVocabularies` with an Amazon SDK or CLI

The following code examples show how to use `ListVocabularies`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Create and refine a custom vocabulary

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// List custom vocabularies for the current account. Optionally specify a
 name
    /// filter and a specific state to filter the vocabularies list.
    /// </summary>
    /// <param name="nameContains">Optional string the vocabulary name must
 contain.</param>
    /// <param name="stateEquals">Optional state of the vocabulary.</param>
    /// <returns>List of information about the vocabularies.</returns>
    public async Task<List<VocabularyInfo>> ListCustomVocabularies(string?
 nameContains = null,
        VocabularyState? stateEquals = null)
    {
        var response = await _amazonTranscribeService.ListVocabulariesAsync(
            new ListVocabulariesRequest()
            {
                NameContains = nameContains,
                StateEquals = stateEquals
            });
        return response.Vocabularies;
    }
```

- For API details, see ListVocabularies in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To list your custom vocabularies

The following `list-vocabularies` example lists the custom vocabularies associated with your Amazon account and Region.

```
aws transcribe list-vocabularies
```

Output:

```
{
    "NextToken": "NextToken",
    "Vocabularies": [
        {
            "VocabularyName": "ards-test-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-27T22:00:27.330000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "sample-test",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T23:04:11.044000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-test-3-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T22:12:22.277000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-test-2",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T21:53:50.455000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-1-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T21:39:33.356000+00:00",
            "VocabularyState": "READY"
        }
    ]
}
```

For more information, see Custom Vocabularies in the *Amazon Transcribe Developer Guide*.

- For API details, see ListVocabularies in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string
 in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs
```

- For API details, see ListVocabularies in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

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

- For API details, see ListVocabularies in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StartMedicalTranscriptionJob` with an Amazon SDK or CLI

The following code examples show how to use `StartMedicalTranscriptionJob`.

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// Start a medical transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the medical transcription job.</
param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon
 S3 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="outputBucketName">Location for the output, typically an
Amazon S3 location.</param>
    /// <param name="transcriptionType">Conversation or dictation transcription
type.</param>
    /// <returns>A MedicalTransactionJob instance with information on the new
job.</returns>
    public async Task<MedicalTranscriptionJob> StartMedicalTranscriptionJob(
        string jobName, string mediaFileUri,
        MediaFormat mediaFormat, string outputBucketName,
Amazon.TranscribeService.Type transcriptionType)
    {
        var response = await
_amazonTranscribeService.StartMedicalTranscriptionJobAsync(
            new StartMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode =
                    LanguageCode
                        .EnUS, // The value must be en-US for medical
transcriptions.
                OutputBucketName = outputBucketName,
                OutputKey =
                    jobName, // The value is a key used to fetch the output of
the transcription.
                Specialty = Specialty.PRIMARYCARE, // The value PRIMARYCARE must
be set.
                Type = transcriptionType
            });
        return response.MedicalTranscriptionJob;
    }
```

- For API details, see StartMedicalTranscriptionJob in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

Example 1: To transcribe a medical dictation stored as an audio file

The following `start-medical-transcription-job` example transcribes an audio file. You specify the location of the transcription output in the `OutputBucketName` parameter.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfile.json
```

Contents of `myfile.json`:

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

Output:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-
job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T00:35:22.256000+00:00",
        "CreationTime": "2020-09-20T00:35:22.218000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

For more information, see Batch Transcription Overview in the *Amazon Transcribe Developer Guide*.

Example 2: To transcribe a clinician-patient dialogue stored as an audio file

The following `start-medical-transcription-job` example transcribes an audio file containing a clinician-patient dialogue. You specify the location of the transcription output in the OutputBucketName parameter.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysecondfile.json
```

Contents of `mysecondfile.json`:

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

Output:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T23:19:49.965000+00:00",
        "CreationTime": "2020-09-20T23:19:49.941000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

For more information, see Batch Transcription Overview in the *Amazon Transcribe Developer Guide*.

Example 3: To transcribe a multichannel audio file of a clinician-patient dialogue

The following `start-medical-transcription-job` example transcribes the audio from each channel in the audio file and merges the separate transcriptions from each channel into a single transcription output. You specify the location of the transcription output in the `OutputBucketName` parameter.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mythirdfile.json
```

Contents of `mythirdfile.json`:

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

Output:

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

For more information, see Channel Identification in the *Amazon Transcribe Developer Guide*.

Example 4: To transcribe an audio file of a clinician-patient dialogue and identify the speakers in the transcription output

The following `start-medical-transcription-job` example transcribes an audio file and labels the speech of each speaker in the transcription output. You specify the location of the transcription output in the `OutputBucketName` parameter.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfourthfile.json
```

Contents of `myfourthfile.json`:

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

Output:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T18:43:37.265000+00:00",
        "CreationTime": "2020-09-21T18:43:37.157000+00:00",
        "Settings": {
            "ShowSpeakerLabels": true,
            "MaxSpeakerLabels": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

For more information, see Identifying Speakers in the *Amazon Transcribe Developer Guide*.

Example 5: To transcribe a medical conversation stored as an audio file with up to two transcription alternatives

The following `start-medical-transcription-job` example creates up to two alternative transcriptions from a single audio file. Every transcriptions has a level of confidence associated with it. By default, Amazon Transcribe returns the transcription with the highest confidence level. You can specify that Amazon Transcribe return additional transcriptions with lower confidence levels. You specify the location of the transcription output in the `OutputBucketName` parameter.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfifthfile.json
```

Contents of `myfifthfile.json`:

```
{
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    },
    "Settings":{
        "ShowAlternatives": true,
        "MaxAlternatives": 2
    }
}
```

Output:

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

For more information, see Alternative Transcriptions in the *Amazon Transcribe Developer Guide*.

Example 6: To transcribe an audio file of a medical dictation with up to two alternative transcriptions

The following `start-medical-transcription-job` example transcribes an audio file and uses a vocabulary filter to mask any unwanted words. You specify the location of the transcription output in the OutputBucketName parameter.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysixthfile.json
```

Contents of `mysixthfile.json`:

```
{
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    },
    "Settings":{
          "ShowAlternatives": true,
          "MaxAlternatives": 2
    }
}
```

Output:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T21:01:14.592000+00:00",
        "CreationTime": "2020-09-21T21:01:14.569000+00:00",
        "Settings": {
            "ShowAlternatives": true,
            "MaxAlternatives": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

For more information, see Alternative Transcriptions in the *Amazon Transcribe Developer Guide*.

Example 7: To transcribe an audio file of a medical dictation with increased accuracy by using a custom vocabulary

The following `start-medical-transcription-job` example transcribes an audio file and uses a medical custom vocabulary you've previously created to increase the transcription accuracy. You specify the location of the transcription output in the `OutputBucketName` parameter.

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

Contents of `mysixthfile.json`:

```
{
    "MedicalTranscriptionJobName": "vocabulary-dictation-medical-transcription-
job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    },
    "Settings":{
        "VocabularyName": "cli-medical-vocab-1"
    }
}
```

Output:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "vocabulary-dictation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T21:17:27.045000+00:00",
        "CreationTime": "2020-09-21T21:17:27.016000+00:00",
        "Settings": {
            "VocabularyName": "cli-medical-vocab-1"
        },
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

For more information, see Medical Custom Vocabularies in the *Amazon Transcribe Developer Guide*.

- For API details, see StartMedicalTranscriptionJob in *Amazon CLI Command Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

Create the client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Start a medical transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
```

```
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and
 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the
 same region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see Amazon SDK for JavaScript Developer Guide.
- For API details, see StartMedicalTranscriptionJob in *Amazon SDK for JavaScript API Reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StartTranscriptionJob` with an Amazon SDK or CLI

The following code examples show how to use `StartTranscriptionJob`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Create and refine a custom vocabulary
- Transcribe audio and get job data

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// Start a transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon
 S3 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="languageCode">The language code of the media file, such as
 en-US.</param>
    /// <param name="vocabularyName">Optional name of a custom vocabulary.</
param>
    /// <returns>A TranscriptionJob instance with information on the new job.</
returns>
    public async Task<TranscriptionJob> StartTranscriptionJob(string jobName,
 string mediaFileUri,
        MediaFormat mediaFormat, LanguageCode languageCode, string?
 vocabularyName)
    {
        var response = await _amazonTranscribeService.StartTranscriptionJobAsync(
            new StartTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode = languageCode,
                Settings = vocabularyName != null ? new Settings()
                {
                    VocabularyName = vocabularyName
                } : null
            });
        return response.TranscriptionJob;
    }
```

- For API details, see StartTranscriptionJob in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

Example 1: To transcribe an audio file

The following `start-transcription-job` example transcribes your audio file.

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

Contents of `myfile.json`:

```
{
    "TranscriptionJobName": "cli-simple-transcription-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
}
```

For more information, see Getting Started (Amazon Command Line Interface) in the *Amazon Transcribe Developer Guide*.

Example 2: To transcribe a multi-channel audio file

The following `start-transcription-job` example transcribes your multi-channel audio file.

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

Contents of `mysecondfile.json`:

```
{
    "TranscriptionJobName": "cli-channelid-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "ChannelIdentification":true
    }
}
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-channelid-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-17T16:07:56.817000+00:00",
        "CreationTime": "2020-09-17T16:07:56.784000+00:00",
        "Settings": {
            "ChannelIdentification": true
        }
    }
}
```

For more information, see Transcribing Multi-Channel Audio in the *Amazon Transcribe Developer Guide*.

Example 3: To transcribe an audio file and identify the different speakers

The following `start-transcription-job` example transcribes your audio file and identifies the speakers in the transcription output.

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

Contents of `mythirdfile.json`:

```
{
    "TranscriptionJobName": "cli-speakerid-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
    "ShowSpeakerLabels": true,
    "MaxSpeakerLabels": 2
    }
}
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-speakerid-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-17T16:22:59.696000+00:00",
        "CreationTime": "2020-09-17T16:22:59.676000+00:00",
        "Settings": {
            "ShowSpeakerLabels": true,
            "MaxSpeakerLabels": 2
        }
    }
}
```

For more information, see Identifying Speakers in the *Amazon Transcribe Developer Guide*.

Example 4: To transcribe an audio file and mask any unwanted words in the transcription output

The following `start-transcription-job` example transcribes your audio file and uses a vocabulary filter you've previously created to mask any unwanted words.

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

Contents of `myfourthfile.json`:

```
{
    "TranscriptionJobName": "cli-filter-mask-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
          "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyFilterName": "your-vocabulary-filter",
        "VocabularyFilterMethod": "mask"
    }
}
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-filter-mask-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyFilterName": "your-vocabulary-filter",
            "VocabularyFilterMethod": "mask"
        }
    }
}
```

For more information, see Filtering Transcriptions in the *Amazon Transcribe Developer Guide*.

Example 5: To transcribe an audio file and remove any unwanted words in the transcription output

The following `start-transcription-job` example transcribes your audio file and uses a vocabulary filter you've previously created to mask any unwanted words.

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

Contents of `myfifthfile.json`:

```
{
    "TranscriptionJobName": "cli-filter-remove-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyFilterName": "your-vocabulary-filter",
        "VocabularyFilterMethod": "remove"
    }
}
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-filter-remove-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyFilterName": "your-vocabulary-filter",
            "VocabularyFilterMethod": "remove"
        }
    }
}
```

For more information, see Filtering Transcriptions in the *Amazon Transcribe Developer Guide*.

Example 6: To transcribe an audio file with increased accuracy using a custom vocabulary

The following `start-transcription-job` example transcribes your audio file and uses a vocabulary filter you've previously created to mask any unwanted words.

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

Contents of `mysixthfile.json`:

```
{
    "TranscriptionJobName": "cli-vocab-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyName": "your-vocabulary"
    }
}
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-vocab-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyName": "your-vocabulary"
        }
    }
}
```

For more information, see Filtering Transcriptions in the *Amazon Transcribe Developer Guide*.

Example 7: To identify the language of an audio file and transcribe it

The following `start-transcription-job` example transcribes your audio file and uses a vocabulary filter you've previously created to mask any unwanted words.

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

Contents of `myseventhfile.json`:

```
{
    "TranscriptionJobName": "cli-identify-language-transcription-job",
    "IdentifyLanguage": true,
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
}
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-identify-language-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-18T22:27:23.970000+00:00",
        "CreationTime": "2020-09-18T22:27:23.948000+00:00",
        "IdentifyLanguage": true
    }
}
```

For more information, see Identifying the Language in the *Amazon Transcribe Developer Guide*.

Example 8: To transcribe an audio file with personally identifiable information redacted

The following `start-transcription-job` example transcribes your audio file and redacts any personally identifiable information in the transcription output.

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

Contents of `myeigthfile.json`:

```
{
    "TranscriptionJobName": "cli-redaction-job",
    "LanguageCode": "language-code",
    "Media": {
        "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
    },
    "ContentRedaction": {
        "RedactionOutput":"redacted",
        "RedactionType":"PII"
    }
}
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-redaction-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-25T23:49:13.195000+00:00",
        "CreationTime": "2020-09-25T23:49:13.176000+00:00",
        "ContentRedaction": {
            "RedactionType": "PII",
            "RedactionOutput": "redacted"
        }
    }
}
```

For more information, see Automatic Content Redaction in the *Amazon Transcribe Developer Guide*.

Example 9: To generate a transcript with personally identifiable information (PII) redacted and an unredacted transcript

The following `start-transcription-job` example generates two transcrptions of your audio file, one with the personally identifiable information redacted, and the other without any redactions.

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

Contents of `myninthfile.json`:

```
{
    "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
    "LanguageCode": "language-code",
    "Media": {
          "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
        },
    "ContentRedaction": {
        "RedactionOutput":"redacted_and_unredacted",
        "RedactionType":"PII"
    }
}
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-25T23:59:47.677000+00:00",
        "CreationTime": "2020-09-25T23:59:47.653000+00:00",
        "ContentRedaction": {
            "RedactionType": "PII",
            "RedactionOutput": "redacted_and_unredacted"
        }
    }
}
```

For more information, see Automatic Content Redaction in the *Amazon Transcribe Developer Guide*.

Example 10: To use a custom language model you've previously created to transcribe an audio file.

The following `start-transcription-job` example transcribes your audio file with a custom language model you've previously created.

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

Contents of `mytenthfile.json`:

```
{
    "TranscriptionJobName": "cli-clm-2-job-1",
    "LanguageCode": "language-code",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.file-extension"
    },
    "ModelSettings": {
        "LanguageModelName":"cli-clm-2"
    }
}
```

Output:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-clm-2-job-1",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.file-
extension"
        },
        "StartTime": "2020-09-28T17:56:01.835000+00:00",
        "CreationTime": "2020-09-28T17:56:01.801000+00:00",
        "ModelSettings": {
            "LanguageModelName": "cli-clm-2"
        }
    }
}
```

For more information, see Improving Domain-Specific Transcription Accuracy with Custom Language Models in the *Amazon Transcribe Developer Guide*.

- For API details, see StartTranscriptionJob in *Amazon CLI Command Reference*.

JavaScript

SDK for JavaScript (v3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

Start a transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
```

```
import { StartTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME",
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_LOCATION",
    // For example, "https://transcribe-demo.s3-REGION.amazonaws.com/
hello_world.wav"
  },
  OutputBucketName: "OUTPUT_BUCKET_NAME",
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Create the client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- For more information, see Amazon SDK for JavaScript Developer Guide.
- For API details, see StartTranscriptionJob in *Amazon SDK for JavaScript API Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is
 started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when
 transcribing
                            the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job
```

- For API details, see StartTranscriptionJob in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

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

        oo_result = lo_tnb->starttranscriptionjob(
          iv_transcriptionjobname = iv_job_name
          io_media = lo_media
          iv_mediaformat = iv_media_format
          iv_languagecode = iv_language_code
          io_settings = lo_settings ).
        MESSAGE 'Transcription job started.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- For API details, see StartTranscriptionJob in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateVocabulary` with an Amazon SDK or CLI

The following code examples show how to use `UpdateVocabulary`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Create and refine a custom vocabulary

.NET

Amazon SDK for .NET

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    /// <summary>
    /// Update a custom vocabulary with new values. Update overwrites all
 existing information.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> UpdateCustomVocabulary(LanguageCode
 languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.UpdateVocabularyAsync(
            new UpdateVocabularyRequest()
            {
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- For API details, see UpdateVocabulary in *Amazon SDK for .NET API Reference*.

CLI

Amazon CLI

To update a custom vocabulary with new terms.

The following `update-vocabulary` example overwrites the terms used to create a custom vocabulary with the new ones that you provide. Prerequisite: to replace the terms in a custom vocabulary, you need a file with new terms.

```
aws transcribe update-vocabulary \
    --vocabulary-file-uri s3://amzn-s3-demo-bucket/Amazon-S3-Prefix/custom-
vocabulary.txt \
    --vocabulary-name custom-vocabulary \
    --language-code language-code
```

Output:

```
{
    "VocabularyName": "custom-vocabulary",
    "LanguageCode": "language",
    "VocabularyState": "PENDING"
}
```

For more information, see Custom Vocabularies in the *Amazon Transcribe Developer Guide*.

- For API details, see UpdateVocabulary in *Amazon CLI Command Reference*.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.",
 vocabulary_name)
        raise
```

- For API details, see UpdateVocabulary in *Amazon SDK for Python (Boto3) API Reference*.

SAP ABAP

SDK for SAP ABAP

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
    TRY.
        IF it_phrases IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            it_phrases = it_phrases ).
        ELSEIF iv_vocab_file_uri IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            iv_vocabularyfileuri = iv_vocab_file_uri ).
        ENDIF.
        MESSAGE 'Vocabulary updated.' TYPE 'I'.
```

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

- For API details, see UpdateVocabulary in *Amazon SDK for SAP ABAP API reference*.

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Scenarios for Amazon Transcribe using Amazon SDKs

The following code examples show you how to implement common scenarios in Amazon Transcribe with Amazon SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within Amazon Transcribe or combined with other Amazon Web Services services. Each scenario includes a link to the complete source code, where you can find instructions on how to set up and run the code.

Scenarios target an intermediate level of experience to help you understand service actions in context.

Examples

Build an Amazon Transcribe streaming app

The following code example shows how to build an app that records, transcribes, and translates live audio in real-time, and emails the results.

JavaScript

SDK for JavaScript (v3)

Shows how to use Amazon Transcribe to build an app that records, transcribes, and translates live audio in real-time, and emails the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Convert text to speech and back to text using an Amazon SDK

The following code example shows how to:

- Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file.
- Upload the audio file to an Amazon S3 bucket.
- Use Amazon Transcribe to convert the audio file to text.
- Display the text.

Rust

SDK for Rust

Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file, upload the audio file to an Amazon S3 bucket, use Amazon Transcribe to convert that audio file to text, and display the text.

For complete source code and instructions on how to set up and run, see the full example on GitHub.

Services used in this example

- Amazon Polly
- Amazon S3
- Amazon Transcribe

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Create and refine an Amazon Transcribe custom vocabulary using an Amazon SDK

The following code example shows how to:

- Upload an audio file to Amazon S3.
- Run an Amazon Transcribe job to transcribe the file and get the results.
- Create and refine a custom vocabulary to improve transcription accuracy.
- Run jobs with custom vocabularies and get the results.

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

Transcribe an audio file that contains a reading of Jabberwocky by Lewis Carroll. Start by creating functions that wrap Amazon Transcribe actions.

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is
 started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when
 transcribing
                            the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job



def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job



def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.
    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise



def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary
 processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.",
 vocabulary_name)
        raise
    else:
        return response



def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response =
 transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response



def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
```

```
                        vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.",
 vocabulary_name)
        raise



def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string
 in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
 transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
                "Couldn't list vocabularies with filter %s.", vocabulary_filter
            )
            raise
        else:
            return vocabs



def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

Call the wrapper functions to transcribe audio without a custom vocabulary and then with different versions of a custom vocabulary to see improved results.

```
def usage_demo():
    """Shows how to use the Amazon Transcribe service."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    s3_resource = boto3.resource("s3")
    transcribe_client = boto3.client("transcribe")

    print("-" * 88)
    print("Welcome to the Amazon Transcribe demo!")
    print("-" * 88)

    bucket_name = f"jabber-bucket-{time.time_ns()}"
    print(f"Creating bucket {bucket_name}.")
    bucket = s3_resource.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": transcribe_client.meta.region_name
        },
    )
    media_file_name = ".media/Jabberwocky.mp3"
    media_object_key = "Jabberwocky.mp3"
    print(f"Uploading media file {media_file_name}.")
    bucket.upload_file(media_file_name, media_object_key)
    media_uri = f"s3://{bucket.name}/{media_object_key}"

    job_name_simple = f"Jabber-{time.time_ns()}"
    print(f"Starting transcription job {job_name_simple}.")
    start_job(
        job_name_simple,
        f"s3://{bucket_name}/{media_object_key}",
        "mp3",
        "en-US",
        transcribe_client,
    )
    transcribe_waiter = TranscribeCompleteWaiter(transcribe_client)
    transcribe_waiter.wait(job_name_simple)
    job_simple = get_job(job_name_simple, transcribe_client)
    transcript_simple = requests.get(
        job_simple["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_simple['jobName']}:")
    print(transcript_simple["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Creating a custom vocabulary that lists the nonsense words to try to "
        "improve the transcription."
    )
    vocabulary_name = f"Jabber-vocabulary-{time.time_ns()}"
    create_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        phrases=[
            "brillig",
            "slithy",
            "borogoves",
            "mome",
            "raths",
            "Jub-Jub",
            "frumious",
            "manxome",
            "Tumtum",
            "uffish",
            "whiffling",
            "tulgey",
            "thou",
            "frabjous",
            "callooh",
            "callay",
            "chortled",
        ],
    )
    vocabulary_ready_waiter = VocabularyReadyWaiter(transcribe_client)
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocabulary_list = f"Jabber-vocabulary-list-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocabulary_list}.")
    start_job(
        job_name_vocabulary_list,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
        vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocabulary_list)
    job_vocabulary_list = get_job(job_name_vocabulary_list, transcribe_client)
    transcript_vocabulary_list = requests.get(
        job_vocabulary_list["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocabulary_list['jobName']}:")
    print(transcript_vocabulary_list["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Updating the custom vocabulary with table data that provides additional
 "
        "pronunciation hints."
    )
    table_vocab_file = "jabber-vocabulary-table.txt"
    bucket.upload_file(table_vocab_file, table_vocab_file)
    update_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        table_uri=f"s3://{bucket.name}/{table_vocab_file}",
    )
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocab_table = f"Jabber-vocab-table-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocab_table}.")
    start_job(
        job_name_vocab_table,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
        vocabulary_name=vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocab_table)
    job_vocab_table = get_job(job_name_vocab_table, transcribe_client)
    transcript_vocab_table = requests.get(
        job_vocab_table["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocab_table['jobName']}:")
    print(transcript_vocab_table["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print("Getting data for jobs and vocabularies.")
    jabber_jobs = list_jobs("Jabber", transcribe_client)
    print(f"Found {len(jabber_jobs)} jobs:")
    for job_sum in jabber_jobs:
        job = get_job(job_sum["TranscriptionJobName"], transcribe_client)
        print(
            f"\t{job['TranscriptionJobName']}, {job['Media']['MediaFileUri']}, "
            f"{job['Settings'].get('VocabularyName')}"
        )

    jabber_vocabs = list_vocabularies("Jabber", transcribe_client)
    print(f"Found {len(jabber_vocabs)} vocabularies:")
    for vocab_sum in jabber_vocabs:
        vocab = get_vocabulary(vocab_sum["VocabularyName"], transcribe_client)
        vocab_content = requests.get(vocab["DownloadUri"]).text
        print(f"\t{vocab['VocabularyName']} contents:")
        print(vocab_content)
```

```
    print("-" * 88)
    print("Deleting demo jobs.")
    for job_name in [job_name_simple, job_name_vocabulary_list,
 job_name_vocab_table]:
        delete_job(job_name, transcribe_client)
    print("Deleting demo vocabulary.")
    delete_vocabulary(vocabulary_name, transcribe_client)
    print("Deleting demo bucket.")
    bucket.objects.delete()
    bucket.delete()
    print("Thanks for watching!")
```

- For API details, see the following topics in *Amazon SDK for Python (Boto3) API Reference*.
  - CreateVocabulary
  - DeleteTranscriptionJob
  - DeleteVocabulary
  - GetTranscriptionJob
  - GetVocabulary
  - ListVocabularies
  - StartTranscriptionJob
  - UpdateVocabulary

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Transcribe audio and get job data with Amazon Transcribe using an Amazon SDK

The following code examples show how to:

- Start a transcription job with Amazon Transcribe.
- Wait for the job to complete.
- Get the URI where the transcript is stored.

For more information, see Getting started with Amazon Transcribe.

Java

SDK for Java 2.x

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

Transcribes a PCM file.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

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

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
 example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }
        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

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

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

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

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

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

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

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

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

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

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

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

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

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

            demand.getAndAdd(n);

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

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

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

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

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

            return audioBuffer;
        }

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

Transcribes streaming audio from your computer's microphone.

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

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

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();
        CompletableFuture<Void> result = 
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

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

    private static InputStream getStreamFromMic() throws LineUnavailableException
{

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

            demand.getAndAdd(n);

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

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

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

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

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

            return audioBuffer;
        }

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

- For API details, see the following topics in *Amazon SDK for Java 2.x API Reference*.
  - GetTranscriptionJob
  - StartTranscriptionJob

Python

SDK for Python (Boto3)

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the Amazon Code Examples Repository.

```
import time
import boto3


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

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

```
def main():
    transcribe_client = boto3.client("transcribe")
    file_uri = "s3://test-transcribe/answer2.wav"
    transcribe_file("Example-job", file_uri, transcribe_client)


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

- For API details, see the following topics in *Amazon SDK for Python (Boto3) API Reference*.
  - GetTranscriptionJob
  - StartTranscriptionJob

For a complete list of Amazon SDK developer guides and code examples, see Using this service with an Amazon SDK. This topic also includes information about getting started and details about previous SDK versions.

Security in Amazon Transcribe

Cloud security at Amazon is the highest priority. As an Amazon customer, you benefit from a data center and network architecture built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between Amazon and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud**: Amazon is responsible for protecting the infrastructure that runs Amazon services in the Amazon Web Services Cloud. Amazon also provides you with services you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the Amazon Compliance Programs. To learn about the compliance programs that apply to Amazon Transcribe, see Amazon Services in Scope by Compliance Program.
- **Security in the cloud**: Your responsibility is determined by the Amazon 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 documentation 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 Amazon services to monitor and secure your Amazon Transcribe resources.

Topics

Identity and Access Management for Amazon Transcribe

Amazon Identity and Access Management (IAM) is an Amazon Web Services service that helps an administrator securely control access to Amazon resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Transcribe resources. IAM is an Amazon Web Services service that you can use with no additional charge.

Topics

Audience

How you use Amazon Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting Amazon Transcribe identity and access)
- **Service administrator** - determine user access and submit permission requests (see How Amazon Transcribe works with IAM)
- **IAM administrator** - write policies to manage access (see Amazon Transcribe identity-based policy examples)

Authenticating with identities

Authentication is how you sign in to Amazon using your identity credentials. You must be authenticated as the Amazon Web Services account root user, an IAM user, or by assuming an IAM role.

For programmatic access, Amazon provides an SDK and CLI to cryptographically sign requests. For more information, see Amazon Signature Version 4 for API requests in the *IAM User Guide*.

Amazon Web Services account root user

When you create an Amazon Web Services account, you begin with one sign-in identity called the Amazon Web Services account *root user* that has complete access to all Amazon Web Services services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access Amazon Web Services services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Amazon Directory Service that accesses Amazon Web Services services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access Amazon using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an Amazon CLI or Amazon API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in Amazon by creating policies and attaching them to Amazon identities or resources. A policy defines permissions when associated with an identity or resource. Amazon evaluates these policies when a principal makes a request. Most policies are stored in Amazon as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose 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 include 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. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use Amazon managed policies from IAM in a resource-based policy.

Other policy types

Amazon supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.

- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in Amazon Organizations. For more information, see Service control policies in the *Amazon Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *Amazon Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. 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 Amazon determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon Transcribe works with IAM

Before you use IAM to manage access to Amazon Transcribe, learn what IAM features are available to use with Amazon Transcribe.

IAM features you can use with Amazon Transcribe

IAM feature	Amazon Transcribe support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes

IAM feature	Amazon Transcribe support
Principal permissions	Yes
Service roles	Yes
Service-linked roles	No

To get a high-level view of how Amazon Transcribe and other Amazon services work with most IAM features, see Amazon services that work with IAM in the *IAM User Guide*.

Identity-based policies for Amazon Transcribe

Supports identity-based policies: Yes

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 Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Amazon Transcribe

To view examples of Amazon Transcribe identity-based policies, see Amazon Transcribe identity-based policy examples.

Resource-based policies within Amazon Transcribe

Supports resource-based policies: No

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 Amazon Web Services services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for Amazon Transcribe

Supports policy actions: Yes

Administrators can use Amazon 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. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Amazon Transcribe actions, see Actions defined by Amazon Transcribe in the *Service Authorization Reference*.

Policy actions in Amazon Transcribe use the `transcribe` prefix before the action. To specify multiple actions in a single statement, separate them with commas.

```
"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 view examples of Amazon Transcribe identity-based policies, see Amazon Transcribe identity-based policy examples.

Policy resources for Amazon Transcribe

Supports policy resources: Yes

Administrators can use Amazon JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Amazon Transcribe resource types and their ARNs, see Resources defined by Amazon Transcribe in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon Transcribe.

To view examples of Amazon Transcribe identity-based policies, see Amazon Transcribe identity-based policy examples.

Policy condition keys for Amazon Transcribe

Supports service-specific policy condition keys: Yes

Administrators can use Amazon JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all Amazon global condition keys, see Amazon global condition context keys in the *IAM User Guide*.

To see a list of Amazon Transcribe condition keys, see Condition keys for Amazon Transcribe in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon Transcribe.

To view examples of Amazon Transcribe identity-based policies, see Amazon Transcribe identity-based policy examples.

ACLs in Amazon Transcribe

Supports ACLs: No

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.

ABAC with Amazon Transcribe

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and Amazon resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

For more information about tagging Amazon Transcribe resources, see Tagging resources. For more detailed information on tag-based access control, see Controlling access to Amazon resources using tags.

Using temporary credentials with Amazon Transcribe

Supports temporary credentials: Yes

Temporary credentials provide short-term access to Amazon resources and are automatically created when you use federation or switch roles. Amazon recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and Amazon Web Services services that work with IAM in the *IAM User Guide*.

Cross-service principal permissions for Amazon Transcribe

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an Amazon Web Services service, combined with the requesting Amazon Web Services service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for Amazon Transcribe

Supports service roles: Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an Amazon Web Services service in the *IAM User Guide*.

Warning

Changing the permissions for a service role might break Amazon Transcribe functionality. Edit service roles only when Amazon Transcribe provides guidance to do so.

Service-linked roles for Amazon Transcribe

Supports service-linked roles: No

A service-linked role is a type of service role that is linked to an Amazon Web Services service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your Amazon Web Services account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

Amazon Transcribe doesn't support service-linked roles.

For details about creating or managing service-linked roles for other services, see Amazon services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Cross-service confused deputy prevention

A confused deputy is an entity (a service or an account) that is coerced by a different entity to perform an action. This type of impersonation can happen cross-account and cross-service.

To prevent confused deputies, Amazon provides tools that help you protect your data for all services using service principals that have been given access to resources in your Amazon Web Services account. This section focuses on cross-service confused deputy prevention specific to Amazon Transcribe; however, you can learn more about this topic in the confused deputy problem section of the *IAM User Guide*.

To limit the permissions IAM gives to Amazon Transcribe to access your resources, we recommend using the global condition context keys `aws:SourceArn` and `aws:SourceAccount` in your resource policies.

If you use both of these global condition context keys, and the `aws:SourceArn` value contains the Amazon Web Services account ID, the `aws:SourceAccount` value and the Amazon Web Services account in `aws:SourceArn` must use the same Amazon Web Services account ID when used in the same policy statement.

If you want only one resource to be associated with the cross-service access, use `aws:SourceArn`. If you want to associate any resource in that Amazon Web Services account with cross-service access, use `aws:SourceAccount`.

Note

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the **full ARN** of the resource. If you don't know the full ARN, or if you're specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:transcribe::`*`123456789012`*`:*`.

For an example of an assume role policy that shows how you can prevent a confused deputy issue, see Confused deputy prevention policy.

Amazon Transcribe identity-based policy examples

By default, users and roles don't have permission to create or modify Amazon Transcribe resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Amazon Transcribe, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Transcribe in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon Transcribe resources in your account. These actions can incur costs for your Amazon Web Services account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with Amazon managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *Amazon managed policies* that grant permissions for many common use cases. They are available in your Amazon Web Services account. We recommend that you reduce permissions further by defining Amazon customer managed policies that are specific to your use cases. For more information, see Amazon managed policies or Amazon managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific Amazon Web Services service, such as Amazon CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies

adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your Amazon Web Services account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the Amazon Web Services Management Console

To access the Amazon Transcribe console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon Transcribe resources in your Amazon Web Services account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the Amazon CLI or the Amazon API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that an entity (users and roles) can use the Amazon Web Services Management Console, attach one of the following Amazon-managed policies to them.

- `AmazonTranscribeFullAccess`: Grants full access to create, read, update, delete, and run all Amazon Transcribe resources. It also allows access to Amazon S3 buckets with `transcribe` in the bucket name.
- `AmazonTranscribeReadOnlyAccess`: Grants read-only access to Amazon Transcribe resources so that you can get and list transcription jobs and custom vocabularies.

Note

You can review the managed permission policies by signing in to the IAM Amazon Web Services Management Console and searching by policy name. A search for "transcribe" returns both policies listed above (*AmazonTranscribeReadOnly* and *AmazonTranscribeFullAccess*).

You can also create your own custom IAM policies to allow permissions for Amazon Transcribe API actions. You can attach these custom policies to the entities that require those permissions.

Permissions required for IAM roles

If you create an IAM role to call Amazon Transcribe, it must have permission to access the Amazon S3 bucket. If applicable, the KMS key must also be used to encrypt the contents of the bucket. Refer to the following sections for example policies.

Trust policies

The IAM entity you use to make your transcription request must have a trust policy that enables Amazon Transcribe to assume that role. Use the following Amazon Transcribe trust policy. Note that if you're making a real-time Call Analytics request with post-call analytics enabled, you must use 'Trust policy for real-time Call Analytics'.

Trust policy for Amazon Transcribe

Trust policy for real-time Call Analytics

Amazon S3 input bucket policy

The following policy gives an IAM role permission to access files from the specified input bucket.

JSON

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

Amazon S3 output bucket policy

The following policy gives an IAM role permission to write files to the specified output bucket.

JSON

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

Permissions required for Amazon S3 encryption keys

If you're using a KMS key to encrypt an Amazon S3 bucket, include the following in the KMS key policy. This gives Amazon Transcribe access to the contents of the bucket. For more information about allowing access to KMS keys, see Allowing external Amazon Web Services accounts to access an KMS key in the *Amazon KMS Developer Guide*.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
      },
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-west-2:111122223333:key/KMS-Example-KeyId"
    }
  ]
}
```

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 Amazon CLI or Amazon API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws-cn:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon KMS encryption context policy

The following policy grants the IAM role "`ExampleRole`" permission to use the Amazon KMS *Decrypt* and *Encrypt* operations for this particular KMS key. This policy works **only** for requests with at least one encryption context pair, in this case "`color:indigoBlue`". For more information on Amazon KMS encryption context, see Amazon KMS encryption context.

Confused deputy prevention policy

Here's an example of an assume role policy that shows how you can use `aws:SourceArn` and `aws:SourceAccount` with Amazon Transcribe to prevent a confused deputy issue. For more information on confused deputy prevention, see Cross-service confused deputy prevention.

Viewing transcription jobs based on tags

You can use conditions in your identity-based policy to control access to Amazon Transcribe resources based on tags. This example shows how you might create a policy that allows viewing a transcription job. However, permission is granted only if the transcription job tag `Owner` has the value of that user's user name. This policy also grants the permissions necessary to complete this action using the Amazon Web Services Management Console.

You can attach this policy to the IAM entities in your account. If a role named `test-role` attempts to view a transcription job, the transcription job must be tagged `Owner=test-role` or `owner=test-role` (condition key names are not case-sensitive), otherwise they are denied access. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

For more information on tagging in Amazon Transcribe, see Tagging resources.

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 Amazon Identity and Access Management (IAM).

Topics

I am not authorized to perform an action in Amazon Transcribe

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `transcribe:`*`GetWidget`* permissions.

```
User: arn:aws-cn:iam::123456789012:user/mateojackson is not authorized to perform:
 transcribe:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `transcribe:`*`GetWidget`* action.

If you need help, contact your Amazon administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon Transcribe.

Some Amazon Web Services services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon Transcribe. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

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

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your Amazon administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my Amazon Web Services 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.
- To learn how to provide access to your resources across Amazon Web Services accounts that you own, see Providing access to an IAM user in another Amazon Web Services account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party Amazon Web Services accounts, see Providing access to Amazon Web Services 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 Cross account resource access in IAM in the *IAM User Guide*.

Data protection in Amazon Transcribe

The Amazon shared responsibility model applies to data protection in Amazon Transcribe. As described in this model, Amazon is responsible for protecting the global infrastructure that runs all of the Amazon Web Services Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the Amazon Web Services services that you use. For more information about data privacy, see the Data Privacy FAQ.

For data protection purposes, we recommend that you protect Amazon Web Services account credentials and set up individual users with Amazon IAM Identity Center or Amazon Identity and Access Management (IAM). That way, 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 Amazon resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with Amazon CloudTrail. For information about using CloudTrail trails to capture Amazon activities, see Working with CloudTrail trails in the *Amazon CloudTrail User Guide*.
- Use Amazon encryption solutions, along with all default security controls within Amazon Web Services services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing Amazon 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) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Amazon Transcribe or other Amazon Web Services services using the console, API, Amazon CLI, or Amazon SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Inter-network traffic privacy

An Amazon Virtual Private Cloud (Amazon VPC) endpoint for Amazon Transcribe is a logical entity within a VPC that allows connectivity only to Amazon Transcribe. Amazon VPC routes requests to Amazon Transcribe and routes responses back to the VPC. For more information, see Amazon

PrivateLink concepts. For information about using Amazon VPC endpoints with Amazon Transcribe see Amazon Transcribe and interface VPC endpoints (Amazon PrivateLink).

Data encryption

Data encryption refers to protecting data while in transit and at rest. You can protect your data by using Amazon S3-managed keys or KMS keys at rest, alongside standard Transport Layer Security (TLS) while in transit.

Encryption at rest

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

When you use the `StartTranscriptionJob` operation, you can specify your own KMS key to encrypt the output from a transcription job.

Amazon Transcribe uses an Amazon EBS volume encrypted with the default key.

Encryption in transit

Amazon Transcribe uses TLS 1.2 with Amazon certificates to encrypt data in transit. This includes streaming transcriptions.

Key management

Amazon Transcribe works with KMS keys to provide enhanced encryption for your data. With Amazon S3, you can encrypt your input media when creating a transcription job. Integration with Amazon KMS allows encryption of the output from a `StartTranscriptionJob` request.

If you don't specify a KMS key, the output of the transcription job is encrypted with the default Amazon S3 key (SSE-S3).

For more information on Amazon KMS, see the *Amazon Key Management Service Developer Guide*.

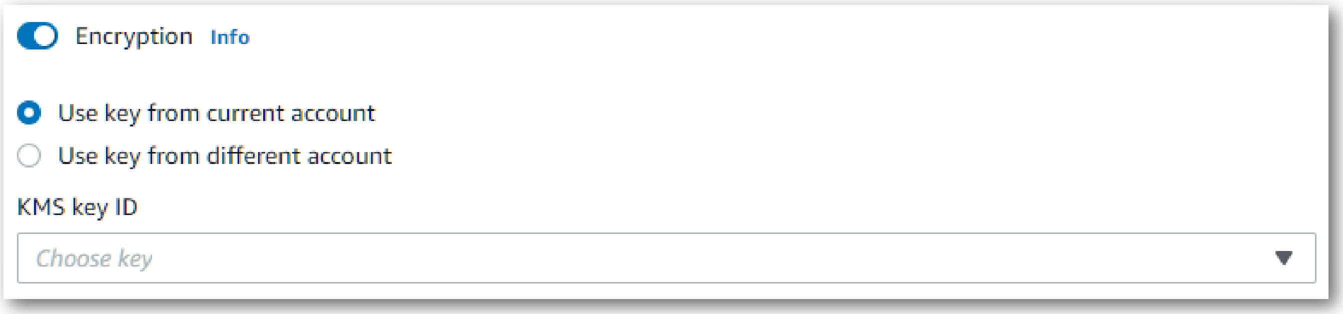
Key management using the Amazon Web Services Management Console

To encrypt the output of your transcription job, you can choose between using a KMS key for the Amazon Web Services account that is making the request, or a KMS key from another Amazon Web Services account.

If you don't specify a KMS key, the output of the transcription job is encrypted with the default Amazon S3 key (SSE-S3).

To enable output encryption:

1. Under **Output data** choose **Encryption**.



2. Choose whether the KMS key is from the Amazon Web Services account you're currently using or from a different Amazon Web Services account. If you want to use a key from the current Amazon Web Services account, choose the key from **KMS key ID**. If you're using a key from a different Amazon Web Services account, you must enter the key's ARN. To use a key from a different Amazon Web Services account, the caller must have `kms:Encrypt` permissions for the KMS key. Refer to Creating a key policy for more information.

Key management using the API

To use output encryption with the API, you must specify your KMS key using the `OutputEncryptionKMSKeyId` parameter of the `StartCallAnalyticsJob`, `StartMedicalTranscriptionJob`, or `StartTranscriptionJob` operation.

If using a key located in the **current** Amazon Web Services account, you can specify your KMS key in one of four ways:

1. Use the KMS key ID itself. For example, `1234abcd-12ab-34cd-56ef-1234567890ab`.
2. Use an alias for the KMS key ID. For example, `alias/ExampleAlias`.
3. Use the Amazon Resource Name (ARN) for the KMS key ID. For example, `arn:aws:kms:region:account-ID:key/1234abcd-12ab-34cd-56ef-1234567890ab`.
4. Use the ARN for the KMS key alias. For example, `arn:aws:kms:region:account-ID:alias/ExampleAlias`.

If using a key located in a **different** Amazon Web Services account than the current Amazon Web Services account, you can specify your KMS key in one of two ways:

1. Use the ARN for the KMS key ID. For example, `arn:aws:kms:region:account-ID:key/1234abcd-12ab-34cd-56ef-1234567890ab`.
2. Use the ARN for the KMS key alias. For example, `arn:aws:kms:region:account-ID:alias/ExampleAlias`.

Note that the entity making the request must have permission to use the specified KMS key.

Amazon KMS encryption context

Amazon KMS encryption context is a map of plain text, non-secret key:value pairs. This map represents additional authenticated data, known as encryption context pairs, which provide an added layer of security for your data. Amazon Transcribe requires a symmetric encryption key to encrypt transcription output into a customer-specified Amazon S3 bucket. To learn more, see Asymmetric keys in Amazon KMS.

When creating your encryption context pairs, **do not** include sensitive information. Encryption context is not secret—it's visible in plain text within your CloudTrail logs (so you can use it to identify and categorize your cryptographic operations).

Your encryption context pair can include special characters, such as underscores (_), dashes (-), slashes (/, \) and colons (:).

Tip

It can be useful to relate the values in your encryption context pair to the data being encrypted. Although not required, we recommend you use non-sensitive metadata related to your encrypted content, such as file names, header values, or unencrypted database fields.

To use output encryption with the API, set the `KMSEncryptionContext` parameter in the `StartTranscriptionJob` operation. In order to provide encryption context for the output encryption operation, the `OutputEncryptionKMSKeyId` parameter must reference a symmetric KMS key ID.

You can use Amazon KMS condition keys with IAM policies to control access to a symmetric encryption KMS key based on the encryption context that was used in the request for a cryptographic operation. For an example encryption context policy, see Amazon KMS encryption context policy.

Using encryption context is optional, but recommended. For more information, see Encryption context.

Opting out of using your data for service improvement

By default, Amazon Transcribe stores and uses voice inputs that it has processed to develop the service and continuously improve your experience. You can opt out of having your content used to develop and improve Amazon Transcribe by using an Amazon Organizations opt-out policy. For information about how to opt out, see AI services opt-out policies.

Monitoring Amazon Transcribe

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Transcribe and your other Amazon solutions. Amazon provides the following monitoring tools to watch Amazon Transcribe, report when something is wrong, and take automatic actions when appropriate:

- **Amazon CloudWatch** monitors your Amazon resources and the applications that you run on Amazon in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics on your Amazon EC2 instances and automatically launch new instances when needed.
- **Amazon CloudWatch Logs** can monitor, store, and access your log files from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage.
- **Amazon CloudTrail** captures API calls and related events made by or on behalf of your Amazon Web Services account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called Amazon, the source IP address from which the calls were made, and when the calls occurred.

For more information, see the *Amazon CloudWatch User Guide*.

Amazon EventBridge is a serverless service that uses events to connect application components together, making it easier for you to build scalable event-driven applications. EventBridge delivers a stream of real-time data from your own applications, Software as a Service (SaaS) applications, and Amazon services and routes that data to targets such as Lambda. You can monitor events that happen in services, and build event-driven architectures. For more information, see the *Amazon EventBridge User Guide*.

Topics

- Monitoring Amazon Transcribe with Amazon CloudWatch
- Monitoring Amazon Transcribe with Amazon CloudTrail
- Using Amazon EventBridge with Amazon Transcribe

Monitoring Amazon Transcribe with Amazon CloudWatch

You can monitor Amazon Transcribe using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the *CloudWatch User Guide*.

Using Amazon CloudWatch metrics and dimensions with Amazon Transcribe

Amazon Transcribe supports CloudWatch metrics and dimensions, which are data that can help you monitor performance. Supported metrics categories include traffic, errors, data transfer, and latency associated with your transcription jobs. Supported metrics are located through CloudWatch in the **Amazon Web Services/Transcribe** namespace.

Note

CloudWatch monitoring metrics are free of charge and don't count against CloudWatch service quotas.

For more information on CloudWatch metrics, see Using Amazon CloudWatch metrics.

Monitoring Amazon Transcribe with Amazon CloudTrail

Amazon Transcribe is integrated with Amazon CloudTrail, a service that provides a record of actions taken in Amazon Transcribe by an Amazon Identity and Access Management (IAM) user or role, or by an Amazon service. CloudTrail captures all API calls for Amazon Transcribe. That includes calls from the Amazon Web Services Management Console and code calls to the Amazon Transcribe APIs, as events. By creating a trail, you can enable continuous delivery of CloudTrail events, including events for Amazon Transcribe, to an Amazon S3 bucket. If you don't create a trail, you can still view the most recent events in the CloudTrail Amazon Web Services Management Console in **Event history**. Using the information collected by CloudTrail, you can see each request that is made to Amazon Transcribe, the IP address from which the request is made, who made the request, when it is made, and additional details.

To learn more about CloudTrail, refer to the *Amazon CloudTrail User Guide*.

Amazon Transcribe and CloudTrail

CloudTrail is enabled on your Amazon Web Services account when you create the account. When activity occurs in Amazon Transcribe, that activity is recorded in a CloudTrail event along with other Amazon Web Services service events in the CloudTrail **Event history**. You can view, search, and download recent events in your Amazon Web Services account. For more information, see Viewing Events with CloudTrail Event History.

To get an ongoing record of events in your Amazon Web Services account, including events for Amazon Transcribe, create a trail. A *trail* is a configuration that enables CloudTrail to deliver events as log files to a specified Amazon S3 bucket. CloudTrail log files contain one or more log entries. An *event* represents a single request from any source. It 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.

By default, when you create a trail in the Amazon Web Services Management Console, the trail applies to all Amazon Web Services Regions. The trail logs events from all Amazon Web Services Regions in the Amazon partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other Amazon Web Services 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. For example, the `CreateVocabulary`, `GetTranscriptionJob`, and `StartTranscriptionJob` operations generate entries in the CloudTrail log files. When CloudTrail logs Amazon Transcribe API operations, the CloudTrail log entry uses empty strings for sensitive information in the request and response parameters, such as Amazon S3 URI values.

Every event or log entry contains information about who generated the request. This information helps you determine the following:

- Whether the request is made with root or IAM user credentials
- Whether the request is made with temporary security credentials for an IAM role or federated user
- Whether the request is made by another Amazon Web Services service

For more information, see the CloudTrail userIdentity Element.

You can also aggregate Amazon Transcribe log files from multiple Amazon Web Services Regions and multiple Amazon Web Services accounts into a single Amazon S3 bucket. For more information, see Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts.

Example: Amazon Transcribe log file entries

A *trail* is a configuration that enables delivery of events as log files to a specified Amazon S3 bucket. CloudTrail log files contain one or more log entries. An *event* represents a single request from any source. It includes such information about the requested action as the date and time of the action, and request parameters. 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` API operations create the following entry.

Note

When CloudTrail logs Amazon Transcribe API operations, the CloudTrail log entry uses empty strings for sensitive information in the request and response parameters, such as Amazon S3 URI values.

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "111122223333",
                "arn": "arn:aws:iam:us-west-2:111122223333:user/my-user-name",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "my-user-name"
            },
            "eventTime": "2022-03-07T15:03:45Z",
            "eventSource": "transcribe.amazonaws.com",
            "eventName": "StartTranscriptionJob",
            "awsRegion": "us-west-2",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "[]",
            "requestParameters": {
                "mediaFormat": "flac",
                "languageCode": "en-US",
                "transcriptionJobName": "my-first-transcription-job",
                "media": {
                    "mediaFileUri": ""
                }
            },
            "responseElements": {
                "transcriptionJob": {
                    "transcriptionJobStatus": "IN_PROGRESS",
                    "mediaFormat": "flac",
                    "creationTime": "2022-03-07T15:03:44.229000-08:00",
                    "transcriptionJobName": "my-first-transcription-job",
                    "languageCode": "en-US",
                    "media": {
                        "mediaFileUri": ""
                    }
                }
            },
            "requestID": "47B8E8D397DCE7A6",
            "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "111122223333",
                "arn": "arn:aws:iam:us-west-2:111122223333:user/my-user-name",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "my-user-name"
            },
            "eventTime": "2022-03-07T15:07:11Z",
            "eventSource": "transcribe.amazonaws.com",
            "eventName": "GetTranscriptionJob",
            "awsRegion": "us-west-2",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "[]",
            "requestParameters": {
                "transcriptionJobName": "my-first-transcription-job"
            },
            "responseElements": {
                "transcriptionJob": {
                    "settings": {

                    },
                    "transcriptionJobStatus": "COMPLETED",
                    "mediaFormat": "flac",
                    "creationTime": "2022-03-07T15:03:44.229000-08:00",
                    "transcriptionJobName": "my-first-transcription-job",
                    "languageCode": "en-US",
                    "media": {
                        "mediaFileUri": ""
                    },
                    "transcript": {
                        "transcriptFileUri": ""
                    }
                }
            },
            "requestID": "BD8798EACDD16751",
            "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        }
    ]
}
```

Using Amazon EventBridge with Amazon Transcribe

With Amazon EventBridge, you can respond to state changes in your Amazon Transcribe jobs by initiating events in other Amazon Web Services services. When a transcription job changes state, EventBridge automatically sends an event to an event stream. You create rules that define the events that you want to monitor in the event stream and the action that EventBridge should take when those events occur. For example, routing the event to another service (or target), which can then take an action. You could, for example, configure a rule to route an event to an Amazon Lambda function when a transcription job has completed successfully. To define EventBridge rules, refer to the following sections.

You can receive notifications for events through multiple channels, including email, Amazon Q Developer in chat applications chat notifications, or Amazon Console Mobile Application push notifications. You can also see notifications in the Console Notifications Center. If you want to set up notifications, you can use Amazon User Notifications. Amazon User Notifications supports aggregation, which can reduce the number of notifications you receive during specific events.

Defining EventBridge rules

To define EventBridge rules, use the Amazon Web Services Management Console. When you define a rule, use Amazon Transcribe as the service name. For an example of how to create an EventBridge rule, see Amazon EventBridge rules.

Before using EventBridge, note the following definitions:

- **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 Amazon Web Services service that processes an event. For example, Amazon Lambda or Amazon Simple Notification Service (Amazon SNS). A target receives events in JSON format.
- **Rule**–A rule matches incoming events that you want EventBridge to watch for and routes them to a target or targets for processing. If a rule routes an event to multiple targets, all of the targets process the event in parallel. A rule can customize the JSON sent to the target.

Amazon EventBridge events are emitted on a best-effort basis. For more information about creating and managing events in EventBridge, see Amazon EventBridge events in the *Amazon EventBridge User Guide*.

The following is an example of an EventBridge rule for Amazon Transcribe that's initiated when a transcription job's status changes to `COMPLETED` or `FAILED`.

```
{
    "source": [
        "aws.transcribe"
    ],
    "detail-type": [
        "Transcribe Job State Change"
    ],
    "detail": {
        "TranscriptionJobStatus": [
            "COMPLETED",
            "FAILED"
        ]
    }
}
```

The rule contains the following fields:

- `source`–The source of the event. For Amazon Transcribe, this is always `aws.transcribe`.
- `detail-type`–An identifier for the details of the event. For Amazon Transcribe, this is always `Transcribe Job State Change`.
- `detail`–The new job status of the transcription job. In this example, the rule initiates an event when the job status changes to `COMPLETED` or `FAILED`.

Amazon Transcribe events

Amazon EventBridge logs several Amazon Transcribe events:

- Transcription job events
- Language identification events
- Call Analytics events
- Call Analytics post-call events
- Vocabulary events

These events all contain the following shared fields:

- `version`: The version of the event data. This value is always `0`.
- `id`: A unique identifier generated by EventBridge for the event.
- `detail-type`: An identifier for the details of the event. For example, `Transcribe Job State Change`.
- `source`: The source of the event. For Amazon Transcribe this is always `aws.transcribe`.
- `account`: The Amazon Web Services account ID of the account that generated the API call.
- `time`: The date and time the event is delivered.
- `region`: The Amazon Web Services Region in which the request is made.
- `resources`: The resources used by the API call. For Amazon Transcribe, this field is always empty.
- `detail`: Additional details about the event.
  - `FailureReason`: This field is present if the state or status changes to `FAILED`, and describes the reason for the `FAILED` state or status.
  - Each event type has additional unique fields that are displayed under `detail`. These unique fields are defined in the following sections after each event example.

Transcription job events

When a job's state changes from `IN_PROGRESS` to `COMPLETED` or `FAILED`, Amazon Transcribe generates an event. To identify the job that changed state and initiate the event in your target, use the event's `TranscriptionJobName` field. An Amazon Transcribe event contains the following information. A `FailureReason` field is added under `detail` if your transcription job status is `FAILED`.

Note that this event applies only to the `StartTranscriptionJob` API operation.

```
{
    "version": "0",
    "id": "event ID",
    "detail-type":"Transcribe Job State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "TranscriptionJobName": "my-first-transcription-job",
            "TranscriptionJobStatus": "COMPLETED" (or "FAILED")
    }
}
```

- `TranscriptionJobName`: The unique name you chose for your transcription job.
- `TranscriptionJobStatus` : The status of the transcription job. This can be `COMPLETED` or `FAILED`.

Language identification events

When you enable automatic language identification, Amazon Transcribe generates an event when the language identification state is `COMPLETED` or `FAILED`. To identify the job that changed state and initiate the event in your target, use the event's `JobName` field. An Amazon Transcribe event contains the following information. A `FailureReason` field is added under `detail` if your language identification status is `FAILED`.

Note that this event applies only to the `StartTranscriptionJob` API operation when the `LanguageIdSettings` parameter is included.

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Language Identification State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "JobType": "TranscriptionJob",
        "JobName": "my-first-lang-id-job",
        "LanguageIdentificationStatus": "COMPLETED" (or "FAILED")
    }
}
```

- `JobType`: For transcription jobs, this value must be `TranscriptionJob`.
- `JobName`: The unique name of your transcription job.
- `LanguageIdentificationStatus`: The status of language identification in a transcription job. This can be `COMPLETED` or `FAILED`.

Call Analytics events

When a Call Analytics job state changes from IN_PROGRESS to COMPLETED or FAILED, Amazon Transcribe generates an event. To identify the Call Analytics job that changed state and initiates the event in your target, use the event's JobName field. An Amazon Transcribe event contains the following information. A FailureReason field is added under detail if your Call Analytics job status is FAILED.

Note that this event applies only to the StartCallAnalyticsJob API operation.

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Call Analytics Job State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "JobName": "my-first-analytics-job",
        "JobStatus": "COMPLETED" (or "FAILED"),
        "FailureReason": "failure reason", // only present when JobStatus is FAILED
        "AnalyticsJobDetails": { // only when you enable optional features such as
 Generative Call Summarization
            "Skipped": []
        }
    }
}
```

- JobName: The unique name of your Call Analytics transcription job.
- JobStatus: The status of your Call Analytics transcription job. This can be either COMPLETED or FAILED.
- FailureReason: This field is present only when the JobStatus is FAILED, and describes the reason for the failure.
- AnalyticsJobDetails: The details of your Call Analytics transcription job, including information about skipped analytics features.

Call Analytics post-call events

When a post-call analytics transcription changes state from `IN_PROGRESS` to `COMPLETED` or `FAILED`, Amazon Transcribe generates an event. To identify the Call Analytics post-call job that changed state and initiate the event in your target, use the event's `StreamingSessionId` field.

Note that this event applies only to the `StartCallAnalyticsStreamTranscription` API operation when the `PostCallAnalyticsSettings` parameter is included.

A `COMPLETED` event contains the following information:

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Call Analytics Post Call Job State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "StreamingSessionId": "session-id",
        "PostCallStatus": "COMPLETED",
        "Transcript": {
            "RedactedTranscriptFileUri": "s3://amzn-s3-demo-bucket/my-output-files/my-
redacted-file.JSON",
            "TranscriptFileUri": "s3://amzn-s3-demo-bucket/my-output-files/my-
file.JSON"
        },
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/my-output-files/my-redacted-
file.WAV",
            "RedactedMediaFileUri": "s3://amzn-s3-demo-bucket/my-output-files/my-
redacted-file.WAV"
        }
    }
}
```

A `FAILED` event contains the following information:

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Call Analytics Post Call Job State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "StreamingSessionId": "session-id",
        "PostCallStatus": "FAILED"
    }
}
```

- `StreamingSessionId`: The identification number assigned to your real-time Call Analytics transcription request.
- `PostCallStatus`: The status of your post-call Call Analytics transcription. This can be either `COMPLETED` or `FAILED`.
- `Transcript`: The URI of your redacted and unredacted transcripts.
- `Media`: The URI of your redacted and unredacted audio files.

Amazon HealthScribe post stream analytics events

When a state changes for a Amazon HealthScribe post-stream analytics operation, such as a ClinicalNoteGenerationResult changing from `IN_PROGRESS` to `COMPLETED`, Amazon HealthScribe generates an event with the following information:

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

- `UpdateType`: The type of post-stream analytics operation that generated the event. The content of the result object varies depending on the `UpdateType`.
- `SessionId`: The identification number for your Amazon HealthScribe stream. Use this ID to identify originating streaming session and then find the post-stream analytics that generated the event.
- `Status`: The status of the post-stream analytics operation. This can be `IN_PROGRESS`, `COMPLETED`, or `FAILED`.
- `ClinicalNoteOutputLocation`: The URI of the output Amazon S3 bucket for the `ClinicalNoteGenerationResult`.
- `TranscriptOutputLocation`: The URI of your transcript.

Vocabulary events

When a custom vocabulary's state changes from `PENDING` to `READY` or `FAILED`, Amazon Transcribe generates an event. To identify the custom vocabulary that changed state and initiate the event in your target, use the event's `VocabularyName` field. An Amazon Transcribe event contains the following information. A `FailureReason` field is added under `detail` if your custom vocabulary state is `FAILED`.

Note

This event applies only to the `CreateVocabulary` API operation.

```
{
    "version": "0",
    "id": "event ID",
    "detail-type": "Vocabulary State Change",
    "source": "aws.transcribe",
    "account": "111122223333",
    "time": "timestamp",
    "region": "us-west-2",
    "resources": [],
    "detail": {
        "VocabularyName": "unique-vocabulary-name",
        "VocabularyState": "READY" (or "FAILED")
    }
}
```

- `VocabularyName`: The unique name of your custom vocabulary.
- `VocabularyState`: The processing state of your custom vocabulary. This can be `READY` or `FAILED`.

Compliance validation for Amazon Transcribe

To learn whether an Amazon Web Services service is within the scope of specific compliance programs, see Amazon Web Services services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see Amazon Web Services Compliance Programs.

You can download third-party audit reports using Amazon Artifact. For more information, see Downloading Reports in Amazon Artifact.

Your compliance responsibility when using Amazon Web Services services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using Amazon Web Services services, see Amazon Security Documentation.

Resilience in Amazon Transcribe

The Amazon global infrastructure is built around Amazon Web Services Regions and Availability Zones. Amazon Web Services 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 zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about Amazon Web Services Regions and Availability Zones, see Amazon Global Infrastructure.

Infrastructure security in Amazon Transcribe

As a managed service, Amazon Transcribe is protected by Amazon global network security. For information about Amazon security services and how Amazon protects infrastructure, see Amazon Cloud Security. To design your Amazon environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar Amazon Well‐Architected Framework*.

You use Amazon published API calls to access Amazon Transcribe through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Vulnerability analysis and management in Amazon Transcribe

Configuration and IT controls are a shared responsibility between Amazon and you, our customer. For more information, see the Amazon shared responsibility model.

Amazon Transcribe and interface VPC endpoints (Amazon PrivateLink)

You can establish a private connection between your VPC and Amazon Transcribe by creating an *interface VPC endpoint*. Interface endpoints are powered by Amazon PrivateLink, a technology that you can use to privately access Amazon Transcribe APIs without an internet gateway, NAT device, VPN connection, or Amazon 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 (Amazon PrivateLink) in the *Amazon VPC User Guide*.

Considerations for Amazon Transcribe VPC endpoints

Before you set up an interface VPC endpoint for Amazon Transcribe, make sure 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 the Amazon VPC Amazon Web Services Management Console or Amazon CLI. For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

For batch transcriptions in Amazon Transcribe, create a VPC endpoint using the following service name:

- com.amazonaws.*us-west-2*.transcribe

For streaming transcriptions in Amazon Transcribe, create a VPC endpoint using the following service name:

- com.amazonaws.*us-west-2*.transcribestreaming

If you enable private DNS for the endpoint, you can make API requests to Amazon Transcribe using its default DNS name for the Amazon Web Services 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 the streaming service or 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 batch transcription actions

The following is an example of an endpoint policy for a batch transcription in Amazon Transcribe. When attached to an endpoint, this policy grants access to the listed Amazon Transcribe actions for all principals on all resources.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartTranscriptionJob",
            "transcribe:ListTranscriptionJobs"
         ],
         "Resource":"*"
      }
   ]
}
```

Example: VPC endpoint policy for Amazon Transcribe streaming transcription actions

The following is an example of an endpoint policy for a streaming transcription in Amazon Transcribe. When attached to an endpoint, this policy grants access to the listed Amazon Transcribe actions for all principals on all resources.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transcribe:StartStreamTranscription",
            "transcribe:StartStreamTranscriptionWebsocket"
         ],
         "Resource":"*"
      }
   ]
}
```

Shared subnets

You cannot create, describe, modify, or delete VPC endpoints in subnets that are shared with you. However, you can use the VPC endpoints in subnets that are shared with you. For information about VPC sharing, see Share your VPC with other accounts in the Amazon Virtual Private Cloud guide.

Security best practices for Amazon Transcribe

The following best practices are general guidelines and don’t represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, use them as helpful considerations rather than prescriptions.

- **Use data encryption, such as Amazon KMS encryption context**

  Amazon KMS encryption context is a map of plain text, non-secret key:value pairs. This map represents additional authenticated data, known as encryption context pairs, which provide an added layer of security for your data.

  For more information, refer to Amazon KMS encryption context.
- **Use temporary credentials whenever possible**

  Where possible, use temporary credentials instead of long-term credentials, such as access keys. For scenarios in which you need IAM users with programmatic access and long-term credentials, we recommend that you rotate access keys. Regularly rotating long-term credentials helps you familiarize yourself with the process. This is useful in case you are ever in a situation where you must rotate credentials, such as when an employee leaves your company. We recommend that you use *IAM access last used information* to rotate and remove access keys safely.

  For more information, see Rotating access keys and Security best practices in IAM.
- **Use IAM roles for applications and Amazon services that require Amazon Transcribe access**

  Use an IAM role to manage temporary credentials for applications or services that need to access Amazon Transcribe. When you use a role, you don't have to distribute long-term credentials, such as passwords or access keys, to an Amazon EC2 instance or Amazon service. IAM roles can supply temporary permissions that applications can use when they make requests to Amazon resources.

  For more information, refer to IAM roles and Common scenarios for roles: Users, applications, and services.
- **Use tag-based access control**

You can use tags to control access within your Amazon Web Services accounts. In Amazon Transcribe. tags can be added to: transcription jobs, custom vocabularies, custom vocabulary filters, and custom language models.

For more information, refer to Tag-based access control.

- **Use Amazon monitoring tools**

  Monitoring is an important part of maintaining the reliability, security, availability, and performance of Amazon Transcribe and your Amazon solutions. You can monitor Amazon Transcribe using CloudTrail.

  For more information, refer to Monitoring Amazon Transcribe with Amazon CloudTrail.
- **Enable Amazon Config**

  Amazon Config can assess, audit, and evaluate the configurations of your Amazon resources. Using Amazon Config, you can review changes in configurations and relationships between Amazon resources. You can also investigate detailed resource configuration histories and determine your overall compliance against the configurations specified in your internal guidelines. This can help you simplify compliance auditing, security analysis, change management, and operational troubleshooting.

  For more information, refer to What Is Amazon Config?

Document history for Amazon Transcribe

- **Latest documentation update:** 13 November 2023

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.

Change	Description	Date
New feature	Amazon HealthScribe now supports new templates for the clinical note summary: BIRP, SIRP, DAP, BH_SOAP, and PH_SOAP. For more information, see Amazon HealthScribe Clinical Documentation file.	May 30, 2025
New feature	Amazon HealthScribe now supports a GIRPP template for the clinical note summary. For more information, see Amazon HealthScribe Clinical Documentation file.	February 4, 2025
New feature	Amazon HealthScribe now supports streaming for transcribing medical conversations in real time.	January 29, 2025
Section Update	Updating ar-SA language support for 8000 frequency.	January 16, 2025
Section Update	Updating ar-SA language support for 8000 frequency.	January 16, 2025

Section Update	Updating the supported language section with unsupported language code that we do not support in Amazon GovCloud (US) (US-West, us-gov-west-1), Amazon GovCloud (US) (US-East, us-gov-east-1), or Africa (Cape Town, af-south-1) regions.	January 14, 2025
Section Update	Extend Transcribe Batch json output with new field called "audio_segments".	July 15, 2024
Feature Update	Update Max Speakers in diarization to be 30 instead of 10.	May 10, 2024
Section Update	Updates to generative call summarization and add error output details.	April 30, 2024
Section Update	Updates on custom vocabular y columns - IPA and SoundsLike.	April 30, 2024
Feature Update	Amazon Transcribe Call Analytics now supports generative call summariza tion.	November 29, 2023
Section Update	Update new PII redaction and Language Identification output format.	November 13, 2023

Feature Update	Diarization can now be combined with channel identification.	March 6, 2023
Feature Update	Channel identification can now be combined with diarization.	March 6, 2023
Section Update	IAM best practices have been updated.	February 13, 2023
New languages	Amazon Transcribe now supports Vietnamese and Swedish.	December 6, 2022
Guide Update	The Amazon Transcribe API Reference is now a standalone guide.	April 1, 2022
New chapter	A new SDK code examples chapter is included.	March 21, 2022
Feature update	Call Analytics now provides call summarization.	March 21, 2022
Chapter update	The introductory chapter now showcases Amazon Transcribe use cases.	March 21, 2022
Chapter update	The Getting started chapter has been updated to be method-specific.	March 21, 2022
Chapter update	The Streaming chapter has been updated and restructu red.	March 21, 2022
New event	There is a new event type: Vocabulary events.	February 7, 2022

Section update	Updates have been made to the custom vocabularies section.	January 20, 2022
New feature	Amazon Transcribe now supports custom language models with streaming transcriptions.	October 20, 2021
New feature	Amazon Transcribe can now generate subtitles for your video files.	September 16, 2021
New feature	Amazon Transcribe now supports PII redaction and identification for streaming.	September 14, 2021
New feature	Amazon Transcribe now supports Amazon KMS encryption context for an added level of security for your Amazon Web Services account resources.	September 10, 2021
New languages	Amazon Transcribe now supports Afrikaans, Danish, Mandarin Chinese (Traditio nal), Thai, New Zealand English, and South African English.	August 26, 2021
New feature	Amazon Transcribe now supports resource tagging.	August 24, 2021
New feature	Amazon Transcribe now supports Call Analytics for batch transcription jobs.	August 4, 2021

New feature	Amazon Transcribe now supports partial result stabilization for streaming transcription.	May 11, 2021
New languages	Amazon Transcribe adds support for Italian and German for streaming audio transcription.	November 4, 2020
Amazon Web Services Region expansion	Amazon Transcribe is now available in the Frankfurt (eu-central-1) and London (eu-west-2).	November 4, 2020
New feature	Amazon Transcribe adds support for interface VPC endpoints in batch transcrip tion.	October 9, 2020
New feature	Amazon Transcribe adds support for channel identific ation in streaming.	September 17, 2020
New feature	Amazon Transcribe adds support for speaker partition ing in streaming.	August 19, 2020
New feature	Amazon Transcribe adds support for interface VPC endpoints in streaming.	June 26, 2020
New feature	Amazon Transcribe adds support for vocabulary filtering in streaming.	May 20, 2020

New feature	Amazon Transcribe adds support for creating a custom vocabulary of words to filter from a transcription.	December 20, 2019
New feature	Amazon Transcribe adds support for queueing transcription jobs.	December 19, 2019
New languages	Amazon Transcribe adds support for Gulf Arabic, Hebrew, Japanese, Malay, Swiss German, Telugu, and Turkish.	November 21, 2019
Amazon Web Services Region expansion	Amazon Transcribe is now available in the Asia Pacific (Tokyo) (ap-northeast-1).	November 21, 2019
New feature	Amazon Transcribe adds support for alternative transcriptions.	November 20, 2019
New languages	Amazon Transcribe adds support for Dutch, Farsi, Indonesian, Irish English, Portuguese, Scottish English, Tamil, and Welsh English.	November 12, 2019
New language	Amazon Transcribe now supports streaming transcrip tion for Australian English (en-AU).	October 25, 2019
Amazon Web Services Region expansion	Amazon Transcribe is now available in the China (Beijing) (cn-north-1) and China (Ningxia) (cn-northwest-1).	October 9, 2019

New feature	Amazon Transcribe allows you to provide your own KMS key to encrypt your transcription output files. For more information, see the OutputEncryptionKMSKeyId parameter of the StartStreamTranscription API.	September 24, 2019
New languages	Amazon Transcribe adds support for Chinese (Mandarin), Simplified, Mainland China and Russian.	August 23, 2019
New feature	Amazon Transcribe adds support for streaming audio transcription using the WebSocket protocol.	July 19, 2019
New feature	Amazon CloudTrail now records events for the StartStreamTranscription API.	July 19, 2019
Amazon Web Services Region expansion	Amazon Transcribe is now available in the US West (N. California) (us-west-1).	June 27, 2019
New language	Amazon Transcribe adds support for Modern Standard Arabic.	May 28, 2019
New feature	Amazon Transcribe now transcribes numeric words into numbers for US English. For example, "forty-two" is transcribed as "42".	May 23, 2019

New language	Amazon Transcribe adds support for Hindi and Indian English.	May 15, 2019
New SDK	The Amazon SDK for C++ now supports Amazon Transcribe.	May 8, 2019
New language	Amazon Transcribe adds support for Spanish.	April 19, 2019
Amazon Web Services Region expansion	Amazon Transcribe is now available in the EU (Frankfurt) (eu-central-1) and Asia Pacific (Seoul) (ap-northeast-2).	April 18, 2019
New language	Amazon Transcribe adds support for streaming transcription in British English, French, and Canadian French.	April 5, 2019
New feature	The Amazon SDK for Ruby V3 now supports Amazon Transcribe	March 25, 2019
New feature	Amazon Transcribe allows custom vocabularies, which are lists of specific words that you want Amazon Transcrib e to recognize in your audio input.	March 25, 2019
New languages	Amazon Transcribe adds support for German and Korean.	March 22, 2019

New language	Amazon Transcribe now supports streaming transcrip tion for US Spanish (es-US).	February 7, 2019
Amazon Web Services Region expansion	Amazon Transcribe is now available in the South America (São Paulo) (sa-east-1).	February 7, 2019
Amazon Web Services Region expansion	Amazon Transcribe is now available in the Asia Pacific (Mumbai) (ap-south-1), Asia Pacific (Singapore) (ap-south east-1), EU (London) (eu-west-2), and EU (Paris) (eu-west3).	January 24, 2019
New languages	Amazon Transcribe adds support for French, Italian, and Brazilian Portuguese.	December 20, 2018
New feature	Amazon Transcribe now supports transcription of audio streams.	November 19, 2018
New languages	Amazon Transcribe adds support for Australian English, British English, and Canadian French.	November 15, 2018
Amazon Web Services Region expansion	Amazon Transcribe is now available in Canada (Central) (ca-central-1) and Asia Pacific (Sydney) (ap-southeast-2).	July 17, 2018
New feature	You can now specify your own location to store the output from a transcription job.	July 11, 2018

New feature	Added Amazon CloudTrail and Amazon CloudWatch Events integration.	June 28, 2018
New feature	Amazon Transcribe adds support for custom vocabular ies.	April 4, 2018
New guide	This is the first release of the *Amazon Transcribe Developer Guide*.	November 29, 2017

Amazon Glossary

For the latest Amazon terminology, see the Amazon glossary in the *Amazon Web Services Glossary Reference*.