Amazon Transcribe Streaming SDK

Release 0.6.2

Amazon

Jun 14, 2023

CONTENTS

The Amazon Transcribe Streaming SDK allows users to directly interface with the Amazon Transcribe Streaming service and their Python programs. The goal of the project is to enable users to integrate directly with Amazon Transcribe without needing anything more than a stream of audio bytes and a basic handler. This project is still in early alpha so the interface is still subject to change and may see rapid iteration. It's highly advised to pin to strict dependencies if using this outside of local testing.

CHAPTER

ONE

API

Note: All interfaces not documented here are considered to be private.

1.1 Client

class amazon_transcribe.client.**TranscribeStreamingClient**(*, *region*, *endpoint_resolver=None*, *credential_resolver=None*)

High level client for orchestrating setup and transmission of audio streams to Amazon TranscribeStreaming service.

Parameters

- **region** (str) – An AWS region to use for Amazon Transcribe (e.g. us-east-2)
- **endpoint_resolver** (Optional[BaseEndpointResolver]) – Optional resolver for client endpoints.
- **credential_resolver** (Optional[CredentialResolver]) – Optional credential resolver for client.

async start_stream_transcription(*, *language_code*, *media_sample_rate_hz*, *media_encoding*, *vocabulary_name=None*, *session_id=None*, *vocab_filter_method=None*, *vocab_filter_name=None*, *show_speaker_label=None*, *enable_channel_identification=None*, *number_of_channels=None*, *enable_partial_results_stabilization=None*, *partial_results_stability=None*, *language_model_name=None*)

Coordinate transcription settings and start stream.

Pay careful attention to language_code and media_sample_rate_hz configurations. Incorrect setups may lead to streams hanging indefinitely. More info on constraints can be found here: https://docs.aws.amazon.com/transcribe/latest/dg/streaming.html

The service processes audio chunks for this operation in realtime, if the audio chunks provided are not coming from a realtime source (e.g. a pre-recorded file) the audio chunks should be rate limited to match the appropriate realtime bitrate for the stream. Failure to send the audio chunks in realtime can lead to signing issues for audio streams longer than 5 minutes.

Parameters

- **language_code** (str) – Indicates the source language used in the input audio stream.
- **media_sample_rate_hz** (int) – The sample rate, in Hertz, of the input audio. We suggest that you use 8000 Hz for low quality audio and 16000 Hz for high quality audio.
- **media_encoding** (str) – The encoding used for the input audio.
- **vocabulary_name** (Optional[str]) – The name of the vocabulary to use when processing the transcription job.
- **session_id** (Optional[str]) – A identifier for the transcription session. Use this parameter when you want to retry a session. If you don't provide a session ID, Amazon Transcribe will generate one for you and return it in the response.
- **vocab_filter_method** (Optional[str]) – The manner in which you use your vocabulary filter to filter words in your transcript. See Transcribe Streaming API docs for more info.
- **vocab_filter_name** (Optional[str]) – The name of the vocabulary filter you've created that is unique to your AWS account. Provide the name in this field to successfully use it in a stream.
- **show_speaker_label** (Optional[bool]) – When true, enables speaker identification in your real-time stream.
- **enable_channel_identification** (Optional[bool]) – When true, instructs Amazon Transcribe to process each audio channel separately and then merge the transcription output of each channel into a single transcription. Amazon Transcribe also produces a transcription of each item. An item includes the start time, end time, and any alternative transcriptions. You can't set both ShowSpeakerLabel and EnableChannelIdentification in the same request. If you set both, your request returns a BadRequestException.
- **number_of_channels** (Optional[int]) – The number of channels that are in your audio stream.
- **enable_partial_results_stabilization** (Optional[bool]) – When true, instructs Amazon Transcribe to present transcription results that have the partial results stabilized. Normally, any word or phrase from one partial result can change in a subsequent partial result. With partial results stabilization enabled, only the last few words of one partial result can change in another partial result.

:param partial_results_stability

You can use this field to set the stability level of the transcription results. A higher stability level means that the transcription results are less likely to change. Higher stability levels can come with lower overall transcription accuracy. Defaults to "high" if not set explicitly.

Parameters

- **language_model_name** (Optional[str]) – The name of the language model you want to use.
- **partial_results_stability** (Optional[str]) –

Return type

StartStreamTranscriptionEventStream

1.2 Response Handlers

class amazon_transcribe.handlers.**TranscriptResultStreamHandler**(*transcript_result_stream*)

Parameters
transcript_result_stream (*TranscriptResultStream*) –

async handle_events()

Process generic incoming events from Amazon Transcribe and delegate to appropriate sub-handlers.

async handle_transcript_event(*transcript_event*)

Specific handling for TranscriptionEvent responses from Amazon Transcribe.

This should be implemented by the end user with desired data handling.

Parameters
transcript_event (*TranscriptEvent*) –

1.3 Transcription Model

class amazon_transcribe.model.**Alternative**(*transcript*, *items*)

Bases: `object`

A list of possible transcriptions for the audio.

Parameters

- **transcript** – The text that was transcribed from the audio.
- **items** – One or more alternative interpretations of the input audio.

class amazon_transcribe.model.**AudioEvent**(*audio_chunk*)

Bases: `BaseEvent`

Provides a wrapper for the audio chunks that you are sending.

Parameters
audio_chunk (`Optional`[`bytes`]) – A blob of audio from your application. You audio stream consists of one or more audio events. The maximum audio chunk size is 32 KB.

property audio_chunk

class amazon_transcribe.model.**AudioStream**(*input_stream=None*, *event_serializer=None*, *eventstream_serializer=None*, *event_signer=None*, *initial_signature=None*, *credential_resolver=None*)

Bases: `BaseStream`

Input audio stream for transcription stream request.

This should never be instantiated by the end user. It will be returned from the client within a relevant wrapper object.

async send_audio_event(*audio_chunk*)

Enqueue audio bytes to be sent for transcription.

Parameters
audio_chunk (`Optional`[`bytes`]) – byte-string chunk of audio input. The maximum audio chunk size is 32 KB.

class amazon_transcribe.model.**Item**(*start_time=None*, *end_time=None*, *item_type=None*, *content=None*, *vocabulary_filter_match=None*, *speaker=None*, *confidence=None*, *stable=None*)

Bases: `object`

A word, phrase, or punctuation mark that is transcribed from the input audio.

Parameters

- **start_time** (`Optional[float]`) – The offset from the beginning of the audio stream to the beginning of the audio that resulted in the item.
- **end_time** (`Optional[float]`) – The offset from the beginning of the audio stream to the end of the audio that resulted in the item.
- **item_type** (`Optional[str]`) – The type of the item.
- **content** (`Optional[str]`) – The word or punctuation that was recognized in the input audio.
- **vocabulary_filter_match** (`Optional[bool]`) – Indicates whether a word in the item matches a word in the vocabulary filter you've chosen for your real-time stream. If True then a word in the item matches your vocabulary filter.
- **speaker** (`Optional[str]`) – If speaker identification is enabled, shows the speakers identified in the real-time stream.
- **confidence** (`Optional[float]`) – A value between 0 and 1 for an item that is a confidence score that Amazon Transcribe assigns to each word or phrase that it transcribes.
- **stable** (`Optional[bool]`) – If partial result stabilization has been enabled, indicates whether the word or phrase in the item is stable. If Stable is true, the result is stable.

class amazon_transcribe.model.**Result**(*result_id=None*, *start_time=None*, *end_time=None*, *is_partial=None*, *alternatives=None*, *channel_id=None*)

Bases: `object`

The result of transcribing a portion of the input audio stream.

Parameters

- **result_id** (`Optional[str]`) – A unique identifier for the result.
- **start_time** (`Optional[float]`) – The offset in seconds from the beginning of the audio stream to the beginning of the result.
- **end_time** (`Optional[float]`) – The offset in seconds from the beginning of the audio stream to the end of the result.
- **is_partial** (`Optional[bool]`) – Amazon Transcribe divides the incoming audio stream into segments at natural points in the audio. Transcription results are returned based on these segments. True indicates that Amazon Transcribe has additional transcription data to send, False to indicate that this is the last transcription result for the segment.
- **alternatives** (`Optional[List[Alternative]]`) – A list of possible transcriptions for the audio. Each alternative typically contains one Item that contains the result of the transcription.
- **channel_id** (`Optional[str]`) – When channel identification is enabled, Amazon Transcribe transcribes the speech from each audio channel separately. You can use ChannelId to retrieve the transcription results for a single channel in your audio stream.

class amazon_transcribe.model.**StartStreamTranscriptionEventStream**(*audio_stream*, *response*)

Bases: `object`

Event stream wrapper containing both input and output interfaces to Amazon Transcribe. This should only be created by the client.

Parameters

- **audio_stream** (*AudioStream*) – Audio input stream generated by client for new transcription requests.
- **response** – Response object from Amazon Transcribe.

property input_stream: *AudioStream*

Audio stream to Amazon Transcribe that takes input audio.

Return type
AudioStream

property output_stream: *TranscriptResultStream*

Response stream containing transcribed event output.

Return type
TranscriptResultStream

property response: *StartStreamTranscriptionResponse*

Response object from Amazon Transcribe containing metadata and response output stream.

Return type
StartStreamTranscriptionResponse

class amazon_transcribe.model.**StartStreamTranscriptionRequest**(*language_code=None*, *media_sample_rate_hz=None*, *media_encoding=None*, *vocabulary_name=None*, *session_id=None*, *vocab_filter_method=None*, *vocab_filter_name=None*, *show_speaker_label=None*, *enable_channel_identification=None*, *number_of_channels=None*, *enable_partial_results_stabilization=None*, *partial_results_stability=None*, *language_model_name=None*)

Bases: `object`

Transcription Request

Parameters

- **language_code** – Indicates the source language used in the input audio stream.
- **media_sample_rate_hz** – The sample rate, in Hertz, of the input audio. We suggest that you use 8000 Hz for low quality audio and 16000 Hz for high quality audio.
- **media_encoding** – The encoding used for the input audio.
- **vocabulary_name** – The name of the vocabulary to use when processing the transcription job.

- `session_id` – A identifier for the transcription session. Use this parameter when you want to retry a session. If you don't provide a session ID, Amazon Transcribe will generate one for you and return it in the response.
- `vocab_filter_method` – The manner in which you use your vocabulary filter to filter words in your transcript.
- `vocab_filter_name` – The name of the vocabulary filter you've created that is unique to your AWS account. Provide the name in this field to successfully use it in a stream.
- `show_speaker_label` – When true, enables speaker identification in your real-time stream.
- `enable_channel_identification` – When true, instructs Amazon Transcribe to process each audio channel separately and then merge the transcription output of each channel into a single transcription. Amazon Transcribe also produces a transcription of each item. An item includes the start time, end time, and any alternative transcriptions. You can't set both ShowSpeakerLabel and EnableChannelIdentification in the same request. If you set both, your request returns a BadRequestException.
- `number_of_channels` – The number of channels that are in your audio stream.
- `enable_partial_results_stabilization` – When true, instructs Amazon Transcribe to present transcription results that have the partial results stabilized. Normally, any word or phrase from one partial result can change in a subsequent partial result. With partial results stabilization enabled, only the last few words of one partial result can change in another partial result.
- `partial_results_stability` – You can use this field to set the stability level of the transcription results. A higher stability level means that the transcription results are less likely to change. Higher stability levels can come with lower overall transcription accuracy.
- `language_model_name` – The name of the language model you want to use.

`class amazon_transcribe.model.StartStreamTranscriptionResponse`(*transcript_result_stream*, *request_id=None*, *language_code=None*, *media_sample_rate_hz=None*, *media_encoding=None*, *vocabulary_name=None*, *session_id=None*, *vocab_filter_name=None*, *vocab_filter_method=None*, *show_speaker_label=None*, *enable_channel_identification=None*, *number_of_channels=None*, *enable_partial_results_stabilization=None*, *partial_results_stability=None*, *language_model_name=None*)

Bases: `object`

Transcription Response

Parameters

- `transcript_result_stream` – Represents the stream of transcription events from Amazon Transcribe to your application.
- `request_id` – An identifier for the streaming transcription.
- `language_code` – Indicates the source language used in the input audio stream.

- **media_sample_rate_hz** – The sample rate, in Hertz, of the input audio. We suggest that you use 8000 Hz for low quality audio and 16000 Hz for high quality audio.
- **media_encoding** – The encoding used for the input audio.
- **session_id** – A identifier for the transcription session. Use this parameter when you want to retry a session. If you don't provide a session ID, Amazon Transcribe will generate one for you and return it in the response.
- **vocab_filter_name** – The name of the vocabulary filter used in your real-time stream.
- **vocab_filter_method** – The manner in which you use your vocabulary filter to filter words in your transcript.
- **show_speaker_label** – Shows whether speaker identification was enabled in the stream.
- **enable_channel_identification** – Shows whether channel identification has been enabled in the stream.
- **number_of_channels** – The number of channels identified in the stream.
- **enable_partial_results_stabilization** – Shows whether partial results stabilization has been enabled in the stream.
- **partial_results_stability** – If partial results stabilization has been enabled in the stream, shows the stability level.
- **language_model_name** – The name of the custom language model used in the transcription.

`class amazon_transcribe.model.`**`Transcript`**`(`*results*`)`

Bases: `object`

The transcription in a TranscriptEvent.

Parameters

results (List[*Result*]) – Result objects that contain the results of transcribing a portion of the input audio stream. The array can be empty.

`class amazon_transcribe.model.`**`TranscriptEvent`**`(`*transcript*`)`

Bases: `BaseEvent`

Represents a set of transcription results from the server to the client. It contains one or more segments of the transcription.

Parameters

transcript (*Transcript*) – The transcription of the audio stream. The transcription is composed of all of the items in the results list.

`class amazon_transcribe.model.`**`TranscriptResultStream`**`(`*raw_stream*, *parser*`)`

Bases: `EventStream`

Transcription result stream containing returned TranscriptEvent output.

Results are surfaced through the async iterator interface (i.e. async for)

Raises

- **BadRequestException** – A client error occurred when the stream was created. Check the parameters of the request and try your request again.
- **LimitExceededException** – Your client has exceeded one of the Amazon Transcribe limits, typically the limit on audio length. Break your audio stream into smaller chunks and try your request again.

- `InternalFailureException` – A problem occurred while processing the audio. Amazon Transcribe terminated processing.
- `ConflictException` – A new stream started with the same session ID. The current stream has been terminated.
- `ServiceUnavailableException` – Service is currently unavailable. Try your request later.

CHAPTER

TWO

INSTALLATION

To install from pip:

```
python -m pip install amazon-transcribe
```

To install from Github:

```
git clone https://github.com/awslabs/amazon-transcribe-streaming-sdk.git
cd amazon-transcribe-streaming-sdk
python -m pip install .
```

To use from your Python application, add *amazon-transcribe* as a dependency in your *requirements.txt* file.

NOTE: This SDK is built on top of the AWS Common Runtime (CRT), a collection of C libraries we interact with through bindings. The CRT is available on PyPI (awscrt) as precompiled wheels for common platforms (Linux, macOS, Windows). Non-standard operating systems may need to compile these libraries themselves.

CHAPTER THREE

USAGE

Setup for this SDK will require either live or prerecorded audio. Full details on the audio input requirements can be found in the Amazon Transcribe Streaming documentation.

Here's an example to get started:

```
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():
    # Setup up our client with our chosen AWS region
    client = TranscribeStreamingClient(region="us-west-2")

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

(continues on next page)

(continued from previous page)

```
    async def write_chunks():
        # An example file can be found at tests/integration/assets/test.wav
        async with aiofile.AIOFile('tests/integration/assets/test.wav', '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()
```

CHAPTER

FOUR

SECURITY

See CONTRIBUTING for more information.

CHAPTER FIVE

LICENSE

This project is licensed under the Apache-2.0 License.

PYTHON MODULE INDEX

a

INDEX

A

H

I

M

O

R

S

T