Speech-to-Text basics

cloud.google.com/speech-to-text/docs/basics

Overview

This document is a guide to the basics of using Speech-to-Text. This conceptual guide covers the types of requests you can make to Speech-to-Text, how to construct those requests, and how to handle their responses. We recommend that all users of Speech-to-Text read this guide and one of the associated tutorials before diving into the API itself.

Note: All users can send up to 60 minutes of audio to Speech-to-Text for free **each month**. If you exceed this amount, cost is incurred. See the pricing page for details.

Try it for yourself

If you're new to Google Cloud, create an account to evaluate how Speech-to-Text performs in real-world scenarios. New customers also get $300 in free credits to run, test, and deploy workloads.

Try Speech-to-Text free

Speech requests

Speech-to-Text has three main methods to perform speech recognition. These are listed below:

- **Synchronous Recognition** (REST and gRPC) sends audio data to the Speech-to-Text API, performs recognition on that data, and returns results after all audio has been processed. Synchronous recognition requests are limited to audio data of 1 minute or less in duration.

- **Asynchronous Recognition** (REST and gRPC) sends audio data to the Speech-to-Text API and initiates a *Long Running Operation*. Using this operation, you can periodically poll for recognition results. Use asynchronous requests for audio data of any duration up to 480 minutes.

- **Streaming Recognition** (gRPC only) performs recognition on audio data provided within a gRPC bi-directional stream. Streaming requests are designed for real-time recognition purposes, such as capturing live audio from a microphone. Streaming recognition provides interim results while audio is being captured, allowing result to appear, for example, while a user is still speaking.

Requests contain configuration parameters as well as audio data. The following sections describe these type of recognition requests, the responses they generate, and how to handle those responses in more detail.

Speech-to-Text API recognition

A Speech-to-Text API synchronous recognition request is the simplest method for performing recognition on speech audio data. Speech-to-Text can process up to 1 minute of speech audio data sent in a synchronous request. After Speech-to-Text processes and recognizes all of the audio, it returns a response.

A synchronous request is blocking, meaning that Speech-to-Text must return a response before processing the next request. Speech-to-Text typically processes audio faster than realtime, processing 30 seconds of audio in 15 seconds on average. In cases of poor audio quality, your recognition request can take significantly longer.

Speech-to-Text has both REST and gRPC methods for calling Speech-to-Text API synchronous and asynchronous requests. This article demonstrates the REST API because it is simpler to show and explain basic use of the API. However, the basic makeup of a REST or gRPC request is quite similar. Streaming Recognition Requests are only supported by gRPC.

Synchronous Speech Recognition Requests

A synchronous Speech-to-Text API request consists of a speech recognition configuration, and audio data. A sample request is shown below:

```
{
    "config": {
        "encoding": "LINEAR16",
        "sampleRateHertz": 16000,
        "languageCode": "en-US",
    },
    "audio": {
        "uri": "gs://bucket-name/path_to_audio_file"
    }
}
```

All Speech-to-Text API synchronous recognition requests must include a speech recognition `config` field (of type RecognitionConfig). A `RecognitionConfig` contains the following sub-fields:

- `encoding` - (required) specifies the encoding scheme of the supplied audio (of type `AudioEncoding`). If you have a choice in codec, prefer a lossless encoding such as `FLAC` or `LINEAR16` for best performance. (For more information, see Audio Encodings.) The `encoding` field is optional for `FLAC` and `WAV` files where the encoding is included in the file header.
- `sampleRateHertz` - (required) specifies the sample rate (in Hertz) of the supplied audio. (For more information on sample rates, see Sample Rates below.) The `sampleRateHertz` field is optional for `FLAC` and `WAV` files where the sample rate is included in the file header.

- `languageCode` - (required) contains the language + region/locale to use for speech recognition of the supplied audio. The language code must be a BCP-47 identifier. Note that language codes typically consist of primary language tags and secondary region subtags to indicate dialects (for example, 'en' for English and 'US' for the United States in the above example.) (For a list of supported languages, see Supported Languages.)
- `maxAlternatives` - (optional, defaults to `1`) indicates the number of alternative transcriptions to provide in the response. By default, the Speech-to-Text API provides one primary transcription. If you wish to evaluate different alternatives, set `maxAlternatives` to a higher value. Note that Speech-to-Text will only return alternatives if the recognizer determines alternatives to be of sufficient quality; in general, alternatives are more appropriate for real-time requests requiring user feedback (for example, voice commands) and therefore are more suited for streaming recognition requests.
- `profanityFilter` - (optional) indicates whether to filter out profane words or phrases. Words filtered out will contain their first letter and asterisks for the remaining characters (e.g. f***). The profanity filter operates on single words, it does not detect abusive or offensive speech that is a phrase or a combination of words.
- `speechContext` - (optional) contains additional contextual information for processing this audio. A context contains the following sub-field:

  `phrases` - contains a list of words and phrases that provide hints to the speech recognition task. For more information, see the information on speech context.

Audio is supplied to Speech-to-Text through the `audio` parameter of type RecognitionAudio. The `audio` field contains **either** of the following sub-fields:

- `content` contains the audio to evaluate, embedded within the request. See Embedding Audio Content below for more information. Audio passed directly within this field is limited to 1 minute in duration.
- `uri` contains a URI pointing to the audio content. The file must not be compressed (for example, gzip). Currently, this field must contain a Google Cloud Storage URI (of format `gs://bucket-name/path_to_audio_file`). See Passing Audio reference by a URI below.)

More information on these request and response parameters appears below.

Sample rates

You specify the sample rate of your audio in the `sampleRateHertz` field of the request configuration, and it must match the sample rate of the associated audio content or stream. Sample rates between 8000 Hz and 48000 Hz are supported within Speech-to-Text. You can specify the sample rate for a `FLAC` or `WAV` file in the file header instead of using the `sampleRateHertz` field. **A `FLAC` file must contain the sample rate in the `FLAC` header in order to be submitted to the Speech-to-Text API.**

If you have a choice when encoding the source material, capture audio using a sample rate of 16000 Hz. Values lower than this may impair speech recognition accuracy, and higher levels have no appreciable effect on speech recognition quality.

However, if your audio data has already been recorded at an existing sample rate other than 16000 Hz, do not resample your audio to 16000 Hz. Most legacy telephony audio, for example, use sample rates of 8000 Hz, which may give less accurate results. If you must use such audio, provide the audio to the Speech API at its native sample rate.

Languages

Speech-to-Text's recognition engine supports a variety of languages and dialects. You specify the language (and national or regional dialect) of your audio within the request configuration's `languageCode` field, using a BCP-47 identifier.

A full list of supported languages for each feature is available on the Language Support page.

Time offsets (timestamps)

Speech-to-Text can include time offset values (timestamps) for the beginning and end of each spoken word that is recognized in the supplied audio. A time offset value represents the amount of time that has elapsed from the beginning of the audio, in increments of 100ms.

Time offsets are especially useful for analyzing longer audio files, where you may need to search for a particular word in the recognized text and locate it (seek) in the original audio. Time offsets are supported for all our recognition methods: `recognize`, `streamingrecognize`, and `longrunningrecognize`.

Time offset values are only included for the first alternative provided in the recognition response.

To include time offsets in the results of your request, set the `enableWordTimeOffsets` parameter to true in your request configuration. For examples using the REST API or the Client Libraries, see Using Time Offsets (Timestamps). For example, you can include the `enableWordTimeOffsets` parameter in the request configuration as shown here:

```
{
"config": {
  "languageCode": "en-US",
  "enableWordTimeOffsets": true
  },
"audio":{
  "uri":"gs://gcs-test-data/gettysburg.flac"
  }
}
```

The result returned by the Speech-to-Text API will contain time offset values for each recognized word as shown following:

```
{
  "name": "6212202767953098955",
  "metadata": {
    "@type":
"type.googleapis.com/google.cloud.speech.v1.LongRunningRecognizeMetadata",
    "progressPercent": 100,
    "startTime": "2017-07-24T10:21:22.013650Z",
    "lastUpdateTime": "2017-07-24T10:21:45.278630Z"
  },
  "done": true,
  "response": {
    "@type":
"type.googleapis.com/google.cloud.speech.v1.LongRunningRecognizeResponse",
    "results": [
      {
        "alternatives": [
          {
            "transcript": "Four score and twenty...(etc)...",
            "confidence": 0.97186122,
            "words": [
              {
                "startTime": "1.300s",
                "endTime": "1.400s",
                "word": "Four"
              },
              {
                "startTime": "1.400s",
                "endTime": "1.600s",
                "word": "score"
              },
              {
                "startTime": "1.600s",
                "endTime": "1.600s",
                "word": "and"
              },
              {
                "startTime": "1.600s",
                "endTime": "1.900s",
                "word": "twenty"
              },
              ...
            ]
          }
        ]
      },
      {
        "alternatives": [
          {
            "transcript": "for score and plenty...(etc)...",
            "confidence": 0.9041967,
          }
        ]
      }
    ]
  }
}
```

Model selection

Speech-to-Text can use one of several machine learning *models* to transcribe your audio file. Google has trained these speech recognition models for specific audio types and sources.

When you send an audio transcription request to Speech-to-Text, you can improve the results that you receive by specifying the source of the original audio. This allows the Speech-to-Text API to process your audio files using a machine learning model trained to recognize speech audio from that particular type of source.

To specify a model for speech recognition, include the `model` field in the `RecognitionConfig` object for your request, specifying the model that you want to use.

Speech-to-Text can use the following types of machine learning models for transcribing your audio files.

Note: See the supported languages page to see which models are available for your language.

Type	Enum constant	Description
Latest Long	`latest_long`	Use this model for any kind of long form content such as media or spontaneous speech and conversations. Consider using this model in place of the video model, especially if the video model is not available in your target language. You can also use this in place of the default model.
Latest Short	`latest_short`	Use this model for short utterances that are a few seconds in length. It is useful for trying to capture commands or other single shot directed speech use cases. Consider using this model instead of the command and search model.
Video	`video`	Use this model for transcribing audio from video clips or other sources (such as podcasts) that have multiple speakers. This model is also often the best choice for audio that was recorded with a high-quality microphone or that has lots of background noise. For best results, provide audio recorded at 16,000Hz or greater sampling rate. **Note:** This is a premium model that costs more than the standard rate.

Phone call	`phone_call`	Use this model for transcribing audio from a phone call. Typically, phone audio is recorded at 8,000Hz sampling rate. **Note:** The enhanced phone model is a premium model that costs more than the standard rate.
ASR: Command and search	`command_and_search`	Use this model for transcribing shorter audio clips. Some examples include voice commands or voice search.
ASR: Default	`default`	Use this model if your audio does not fit any of the other models described in this table. For example, you can use this for long-form audio recordings that feature a single speaker only. The default model will produce transcription results for any type of audio, including audio such as video clips that has a separate model specifically tailored to it. However, recognizing video clip audio using the default model would like yield lower-quality results than using the video model. Ideally, the audio is high-fidelity, recorded at 16,000Hz or greater sampling rate.
Medical dictation	`medical_dictation`	Use this model to transcribe notes dictated by a medical professional.
Medical conversation	`medical_conversation`	Use this model to transcribe a conversation between a medical professional and a patient.

Embedded audio content

Embedded audio is included in the speech recognition request when passing a `content` parameter within the request's `audio` field. For embedded audio provided as content within a gRPC request, that audio must be compatible for Proto3 serialization, and provided as binary data. For embedded audio provided as content within a REST request, that audio must be compatible with JSON serialization and first be Base64-encoded. See Base64 Encoding Your Audio for more information.)

When constructing a request using a Google Cloud client library, you generally will write out this binary (or base-64 encoded) data directly within the `content` field.

Pass audio referenced by a URI

More typically, you will pass a `uri` parameter within the Speech request's `audio` field, pointing to an audio file (in binary format, not base64) located on Google Cloud Storage of the following form:

```
gs://bucket-name/path_to_audio_file
```

For example, the following part of a Speech request references the sample audio file used within the Quickstart:

```
...
    "audio": {
        "uri":"gs://cloud-samples-tests/speech/brooklyn.flac"
    }
...
```

You must have proper access permissions to read Google Cloud Storage files, such as one of the following:

- Publicly readable (such as our sample audio files)
- Readable by your service account, if using service account authorization.
- Readable by a user account, if using 3-legged OAuth for user account authorization.

More information about managing access to Google Cloud Storage is available at Creating and Managing Access Control Lists in the Google Cloud Storage documentation.

Speech-to-Text API responses

As indicated previously, a synchronous Speech-to-Text API response may take some time to return results, proportional to the length of the supplied audio. Once processed, the API will return a response as shown below:

```
{
  "results": [
    {
      "alternatives": [
        {
          "confidence": 0.98267895,
          "transcript": "how old is the Brooklyn Bridge"
        }
      ]
    }
  ]
}
```

These fields are explained below:

    `results` contains the list of results (of type `SpeechRecognitionResult`) where each result corresponds to a segment of audio (segments of audio are separated by pauses). Each result will consist of one or more of the following fields:
        `alternatives` contains a list of possible transcriptions, of type `SpeechRecognitionAlternatives`. Whether more than one alternative appears depends both on whether you requested more than one alternative (by setting `maxAlternatives` to a value greater than `1`) and on whether Speech-to-Text produced alternatives of high enough quality. Each alternative will consist of the following fields:
            - `transcript` contains the transcribed text. See Handling Transcriptions below.
            - `confidence` contains a value between 0 and 1 indicating how confident Speech-to-Text is of the given transcription. See Interpreting Confidence Values below.

If no speech from the supplied audio could be recognized, then the returned `results` list will contain no items. Unrecognized speech is commonly the result of very poor-quality audio, or from language code, encoding, or sample rate values that do not match the supplied audio.

The components of this response are explained in the following sections.

Each synchronous Speech-to-Text API response returns a list of results, rather than a single result containing all recognized audio. The list of recognized audio (within the `transcript` elements) will appear in contiguous order.

Select alternatives

Each result within a successful synchronous recognition response can contain one or more `alternatives` (if the `maxAlternatives` value for the request is greater than `1`). If Speech-to-Text determines that an alternative has a sufficient Confidence Value, then that alternative is included in the response. The first alternative in the response is always the best (most likely) alternative.

Setting `maxAlternatives` to a higher value than `1` does not imply or guarantee that multiple alternatives will be returned. In general, more than one alternative is more appropriate for providing real-time options to users getting results via a Streaming Recognition Request.

Handling transcriptions

Each alternative supplied within the response will contain a `transcript` containing the recognized text. When provided with sequential alternatives, you should concatenate these transcriptions together.

The following Python code iterates over a result list and concatenates the transcriptions together. Note that we take the first alternative (the zeroth) in all cases.

```
response = service_request.execute()
recognized_text = 'Transcribed Text: \n'
for i in range(len(response['results'])):
    recognized_text += response['results'][i]['alternatives'][0]['transcript']
```

Confidence values

The `confidence` value is an estimate between 0.0 and 1.0. It's calculated by aggregating the "likelihood" values assigned to each word in the audio. A higher number indicates an estimated greater likelihood that the individual words were recognized correctly. This field is typically provided only for the top hypothesis, and only for results where `is_final=true`. For example, you may use the `confidence` value to decide whether to show alternative results to the user or ask for confirmation from the user.

Be aware, however, that the model determines the "best", top-ranked result based on more signals than the `confidence` score alone (such as sentence context). Because of this there are occasional cases where the top result doesn't have the highest confidence score. If you haven't requested multiple alternative results, the single "best" result returned may have a lower confidence value than anticipated. This can occur, for example, in cases where rare words are being used. A word that's rarely used can be assigned a low "likelihood" value even if it's recognized correctly. If the model determines the rare word to be the most likely option based on context, that result is returned at the top even if the result's `confidence` value is lower than alternative options.

Note: Your code should not expect `confidence` as a required field as it is not guaranteed to be accurate, or even set, in any of the results.

Asynchronous Requests and Responses

An asynchronous Speech-to-Text API request to the LongRunningRecognize method is identical in form to a synchronous Speech-to-Text API request. However, instead of returning a response, the asynchronous request will initiate a *Long Running Operation* (of type Operation) and return this operation to the callee immediately. You can use asynchronous speech recognition with audio of any length up to 480 minutes.

A typical operation response is shown below:

```
{
  "name": "operation_name",
  "metadata": {
    "@type":
"type.googleapis.com/google.cloud.speech.v1.LongRunningRecognizeMetadata"
    "progressPercent": 34,
    "startTime": "2016-08-30T23:26:29.579144Z",
    "lastUpdateTime": "2016-08-30T23:26:29.826903Z"
  }
}
```

Note that no results are yet present. Speech-to-Text will continue to process the audio and use this operation to store the results. Results will appear in the `response` field of the operation returned when the `LongRunningRecognize` request is complete.

A full response after completion of the request appears below:

```
{
  "name": "1268386125834704889",
  "metadata": {
    "lastUpdateTime": "2016-08-31T00:16:32.169Z",
    "@type":
"type.googleapis.com/google.cloud.speech.v1.LongrunningRecognizeMetadata",
    "startTime": "2016-08-31T00:16:29.539820Z",
    "progressPercent": 100
  }
  "response": {
    "@type":
"type.googleapis.com/google.cloud.speech.v1.LongRunningRecognizeResponse",
    "results": [{
      "alternatives": [{
        "confidence": 0.98267895,
        "transcript": "how old is the Brooklyn Bridge"
      }]}]
  },
  "done": True,
}
```

Note that `done` has been set to `True` and that the operation's `response` contains a set of results of type SpeechRecognitionResult which is the same type returned by a synchronous Speech-to-Text API recognition request.

By default, an asynchronous REST response will set `done` to `False`, its default value; however, because JSON does not require default values to be present within a field, when testing whether an operation is completed, you should test both that the `done` field is present and that it is set to `True`.

Streaming Speech-to-Text API Recognition Requests

A streaming Speech-to-Text API recognition call is designed for real-time capture and recognition of audio, within a bi-directional stream. Your application can send audio on the request stream, and receive interim and final recognition results on the response stream in real time. Interim results represent the current recognition result for a section of audio, while the final recognition result represents the last, best guess for that section of audio.

Streaming requests

Unlike synchronous and asynchronous calls, in which you send both the configuration and audio within a single request, calling the streaming Speech API requires sending multiple requests. The first `StreamingRecognizeRequest` must contain a configuration of type StreamingRecognitionConfig without any accompanying audio. Subsequent `StreamingRecognizeRequest`s sent over the same stream will then consist of consecutive frames of raw audio bytes.

A `StreamingRecognitionConfig` consists of the following fields:

- `config` - (required) contains configuration information for the audio, of type RecognitionConfig and is the same as that shown within synchronous and asynchronous requests.
- `single_utterance` - (optional, defaults to `false`) indicates whether this request should automatically end after speech is no longer detected. If set, Speech-to-Text will detect pauses, silence, or non-speech audio to determine when to end recognition. If not set, the stream will continue to listen and process audio until either the stream is closed directly, or the stream's limit length has been exceeded. Setting `single_utterance` to `true` is useful for processing voice commands.
- `interim_results` - (optional, defaults to `false`) indicates that this stream request should return temporary results that may be refined at a later time (after processing more audio). Interim results will be noted within responses through the setting of `is_final` to `false`.

Streaming responses

Streaming speech recognition results are returned within a series of responses of type StreamingRecognitionResponse. Such a response consists of the following fields:

- `speechEventType` contains events of type SpeechEventType. The value of these events will indicate when a single utterance has been determined to have been completed. The speech events serve as markers within your stream's response.
- `results` contains the list of results, which may be either interim or final results, of type StreamingRecognitionResult. The `results` list contains following the sub-fields:
    - `alternatives` contains a list of alternative transcriptions.
    - `isFinal` indicates whether the results obtained within this list entry are interim or are final. Google might return multiple `isFinal=true` results throughout a single stream, but the `isFinal=true` result is only *guaranteed* after the write stream is closed (half-close).
    - `stability` indicates the volatility of results obtained so far, with `0.0` indicating complete instability while `1.0` indicates complete stability. Note that unlike confidence, which estimates whether a transcription is correct, `stability` estimates whether the given partial result may change. If `isFinal` is set to `true`, `stability` will not be set.