Detect intent with audio input stream

This page shows how to stream audio input to a detect intent request using the API. Dialogflow processes the audio and converts it to text before attempting an intent match. This conversion is known as *audio input*, *speech recognition*, *speech-to-text*, or *STT*.

Streaming is supported by the RPC API and client libraries, but it is not supported by the REST API.

Before you begin

This feature is only applicable when using the API for end-user interactions (/dialogflow/docs/api-overview). If you are using an integration (/dialogflow/docs/integrations), you can skip this guide.

You should do the following before reading this guide:

1. Read Dialogflow basics (/dialogflow/docs/basics).
2. Perform setup steps (/dialogflow/docs/quick/setup).

Create an agent

If you have not already created an agent, create one now:

1. Go to the Dialogflow Console (https://dialogflow.cloud.google.com).
2. If requested, sign in to the Dialogflow Console. See Dialogflow console overview (/dialogflow/docs/console) for more information.
3. Click **Create Agent** in the left sidebar menu. (If you already have other agents, click the agent name, scroll to the bottom and click **Create new agent**.)
4. Enter your agent's name, default language, and default time zone.
5. If you have already created a project, enter that project. If you want to allow the Dialogflow Console to create the project, select **Create a new Google project**.

6. Click the **Create** button.

Import the example file to your agent

The steps in this guide make assumptions about your agent, so you need to import (/dialogflow/docs/agents-settings#export) an agent prepared for this guide. When importing, these steps use the *restore* option, which overwrites all agent settings, intents, and entities.

To import the file, follow these steps:

1. Download the `room-booking-agent.zip` (/dialogflow/docs/data/room-booking-agent.zip) file.
2. Go to the Dialogflow Console (https://dialogflow.cloud.google.com).
3. Select your agent.
4. Click the settings button next to the agent name.
5. Select the **Export and Import** tab.
6. Select **Restore From Zip** and follow instructions to restore the zip file that you downloaded.

Streaming basics

The `Session` (/dialogflow/docs/reference/common-types#sessions) type's `streamingDetectIntent` method returns a bidirectional gRPC streaming object. The available methods for this object vary by language, so see the reference documentation for your client library for details.

The streaming object is used to send and receive data concurrently. Using this object, your client streams audio content to Dialogflow, while concurrently listening for a `StreamingDetectIntentResponse`.

The `streamingDetectIntent` method has a `query_input.audio_config.single_utterance` parameter that affects speech recognition:

- If `false` (default), speech recognition does not cease until the client closes the stream.

- If `true`, Dialogflow will detect a single spoken utterance in input audio. When Dialogflow detects the audio's voice has stopped or paused, it ceases speech recognition and sends a `StreamingDetectIntentResponse` with a recognition result of `END_OF_SINGLE_UTTERANCE` to your client. Any audio sent to Dialogflow on the stream after receipt of `END_OF_SINGLE_UTTERANCE` is ignored by Dialogflow.

In bidirectional streaming, a client can *half-close* the stream object to signal to the server that it won't send more data. For example, in Java and Go, this method is called `closeSend`. It is important to half-close (but not abort) streams in the following situations:

- Your client has finished sending data.
- Your client is configured with `single_utterance` set to true, and it receives a `StreamingDetectIntentResponse` with a recognition result of `END_OF_SINGLE_UTTERANCE`.

After closing a stream, your client should start a new request with a new stream as needed.

Streaming detect intent

The following samples use the `Session` (/dialogflow/docs/reference/common-types#sessions) type's `streamingDetectIntent` method to stream audio.

C#Go (#go)Java (#java)Node.js (#node.js)PHP (#php)Python (#python)Ruby (#ruby)

udPlatform/dotnet-docs-samples/blob/master/dialogflow/api/DialogflowSamples/DetectIntentStream.cs)

```
public static async Task<object> DetectIntentFromStreamAsync(
    string projectId,
    string sessionId,
    string filePath)
{
    var sessionsClient = SessionsClient.Create();
    var sessionName = SessionName.FromProjectSession(projectId, sessionId).ToString(

    // Initialize streaming call, retrieving the stream object
    var streamingDetectIntent = sessionsClient.StreamingDetectIntent();

    // Define a task to process results from the API
    var responseHandlerTask = Task.Run(async () =>
```

```
{
    var responseStream = streamingDetectIntent.GetResponseStream();
    while (await responseStream.MoveNextAsync())
    {
        var response = responseStream.Current;
        var queryResult = response.QueryResult;

        if (queryResult != null)
        {
            Console.WriteLine($"Query text: {queryResult.QueryText}");
            if (queryResult.Intent != null)
            {
                Console.Write("Intent detected:");
                Console.WriteLine(queryResult.Intent.DisplayName);
            }
        }
    }
});

// Instructs the speech recognizer how to process the audio content.
// Note: hard coding audioEncoding, sampleRateHertz for simplicity.
var queryInput = new QueryInput
{
    AudioConfig = new InputAudioConfig
    {
        AudioEncoding = AudioEncoding.Linear16,
        LanguageCode = "en-US",
        SampleRateHertz = 16000
    }
};

// The first request must **only** contain the audio configuration:
await streamingDetectIntent.WriteAsync(new StreamingDetectIntentRequest
{
    QueryInput = queryInput,
    Session = sessionName
});

using (FileStream fileStream = new FileStream(filePath, FileMode.Open))
{
    // Subsequent requests must **only** contain the audio data.
    // Following messages: audio chunks. We just read the file in
    // fixed-size chunks. In reality you would split the user input
    // by time.
    var buffer = new byte[32 * 1024];
```

```
        int bytesRead;
        while ((bytesRead = await fileStream.ReadAsync(
            buffer, 0, buffer.Length)) > 0)
        {
            await streamingDetectIntent.WriteAsync(new StreamingDetectIntentRequest
            {
                Session = sessionName,
                InputAudio = Google.Protobuf.ByteString.CopyFrom(buffer, 0, bytesRea
            });
        };
    }

    // Tell the service you are done sending data
    await streamingDetectIntent.WriteCompleteAsync();

    // This will complete once all server responses have been processed.
    await responseHandlerTask;

    return 0;
}
```

Samples

See the samples page (/dialogflow/docs/tutorials/samples) for best practices on streaming from a browser microphone to Dialogflow.

Previous
← Detect intent with audio input file (/dialogflow/docs/how/detect-intent-audio)

Next
Detect intent with audio output (/dialogflow/docs/how/detect-intent-tts) →

Last updated 2020-08-20 UTC.