OpenAI Agents SDK

Voice Agents Quickstart

0 Create a project

In this quickstart we will create a voice agent you can use in the browser. If you want to check out a new project, you can try out `Next.js` or `Vite`.

```
npm create vite@latest my-project -- --template vanilla-ts
```

1 Install the Agents SDK

```
npm install @openai/agents zod@3
```

Alternatively you can install `@openai/agents-realtime` for a standalone browser package.

2 Generate a client ephemeral token

As this application will run in the user's browser, we need a secure way to connect to the model through the Realtime API. For this we can use an ephemeral client key that should be generated on your backend server. For testing purposes you can also generate a key using `curl` and your regular OpenAI API key.

```
export OPENAI_API_KEY="sk-proj-...(your own key here)"
curl -X POST https://api.openai.com/v1/realtime/client_secrets \
   -H "Authorization: Bearer $OPENAI_API_KEY" \
   -H "Content-Type: application/json" \
   -d '{
     "session": {
       "type": "realtime",
       "model": "gpt-realtime"
     }
   }'
```

The response will contain a “value” string a the top level, which starts with “ek_” prefix. You can use this ephemeral key to establish a WebRTC connection later on. Note that this key is only valid for a short period of time and will need to be regenerated.

3 Create your first Agent

Creating a new `RealtimeAgent` is very similar to creating a regular `Agent`.

```
import { RealtimeAgent } from '@openai/agents/realtime';

const agent = new RealtimeAgent({
  name: 'Assistant',
  instructions: 'You are a helpful assistant.',
});
```

4 Create a session

Unlike a regular agent, a Voice Agent is continuously running and listening inside a `RealtimeSession` that handles the conversation and connection to the model over time. This session will also handle the audio processing, interruptions, and a lot of the other lifecycle functionality we will cover later on.

```
import { RealtimeSession } from '@openai/agents/realtime';

const session = new RealtimeSession(agent, {
  model: 'gpt-realtime',
});
```

The `RealtimeSession` constructor takes an `agent` as the first argument. This agent will be the first agent that your user will be able to interact with.

5 Connect to the session

To connect to the session you need to pass the client ephemeral token you generated earlier on.

```
await session.connect({ apiKey: 'ek_...(put your own key here)' });
```

This will connect to the Realtime API using WebRTC in the browser and automatically configure your microphone and speaker for audio input and output. If you are running your `RealtimeSession` on a backend server (like Node.js) the SDK will automatically use WebSocket as a connection. You can learn more about the different transport layers in the Realtime Transport Layer guide.

6 Putting it all together

```
import { RealtimeAgent, RealtimeSession } from '@openai/agents/realtime';

export async function setupCounter(element: HTMLButtonElement) {
  // ....
  // for quickly start, you can append the following code to the auto-generated TS code

  const agent = new RealtimeAgent({
    name: 'Assistant',
    instructions: 'You are a helpful assistant.',
  });
  const session = new RealtimeSession(agent);
  // Automatically connects your microphone and audio output in the browser via WebRTC.
  try {
    await session.connect({
      // To get this ephemeral key string, you can run the following command or implement the equivalent on the server side:
      // curl -s -X POST https://api.openai.com/v1/realtime/client_secrets -H "Authorization: Bearer $OPENAI_API_KEY" -H "Content-Type: application/json" -d '{"session": {"type": "realtime", "model": "gpt-realtime"}}' | jq .value
      apiKey: 'ek_...(put your own key here)',
    });
    console.log('You are connected!');
  } catch (e) {
    console.error(e);
  }
}
```

7 **Fire up the engines and start talking**

Start up your webserver and navigate to the page that includes your new Realtime Agent code. You should see a request for microphone access. Once you grant access you should be able to start talking to your agent.

```
npm run dev
```

Next Steps

From here you can start designing and building your own voice agent. Voice agents include a lot of the same features as regular agents, but have some of their own unique features.

- Learn how to give your voice agent:
  - Tools
  - Handoffs
  - Guardrails
  - Handle audio interruptions
  - Manage session history
- Learn more about the different transport layers.
  - WebRTC
  - WebSocket
  - Building your own transport mechanism