OpenAI Platform

Realtime API with WebSockets Beta

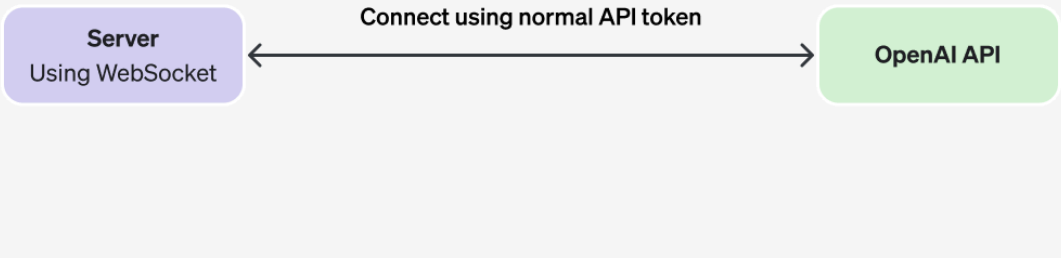
Copy page

Use WebSockets to connect to the Realtime API in server-to-server applications.

WebSockets are a broadly supported API for realtime data transfer, and a great choice for connecting to the OpenAI Realtime API in server-to-server applications. For browser and mobile clients, we recommend connecting via WebRTC. Follow this guide to connect to the Realtime API via WebSocket and start interacting with a Realtime model.

Overview

In a server-to-server integration with Realtime, your backend system will connect via WebSocket directly to the Realtime API. You can use a standard API key to authenticate this connection, since the token will only be available on your secure backend server.



WebSocket connections can also be authenticated with an ephemeral client token (as shown here in the WebRTC connection guide) if you choose to connect to the Realtime API via WebSocket on a client device.

Standard OpenAI API tokens **should only be used in secure server-side environments**.

Connection details

Connecting via WebSocket requires the following connection information:

URL	`wss://api.openai.com/v1/realtime`
Query Parameters	`model` Realtime model ID to connect to, like `gpt-4o-realtime-preview-2024-12-17`

Headers	`Authorization: Bearer YOUR_API_KEY`
	Substitute `YOUR_API_KEY` with a standard API key on the server, or an ephemeral token on insecure clients (note that WebRTC is recommended for this use case).
	`OpenAI-Beta: realtime=v1`
	This header is required during the beta period.

Below are several examples of using these connection details to initialize a WebSocket connection to the Realtime API.

ws module (Node.js) | websocket-client (Python) | WebSocket (browsers)

Connect using the ws module (Node.js) — javascript

```javascript
import WebSocket from "ws";

const url = "wss://api.openai.com/v1/realtime?model=gpt-4o-realtime-preview-2024-12-1
const ws = new WebSocket(url, {
  headers: {
    "Authorization": "Bearer " + process.env.OPENAI_API_KEY,
    "OpenAI-Beta": "realtime=v1",
  },
});

ws.on("open", function open() {
  console.log("Connected to server.");
});

ws.on("message", function incoming(message) {
  console.log(JSON.parse(message.toString()));
});
```

Sending and receiving events

To interact with the Realtime models, you will send and receive messages over the WebSocket interface. The full list of messages that clients can send, and that will be sent from the server, are found in the API reference. Once connected, you'll send and receive events which represent text, audio, function calls, interruptions, configuration updates, and more.

Below, you'll find examples of how to send and receive events over the WebSocket interface in several programming environments.

WebSocket (Node.js / browser) **websocket-client (Python)**

Send and receive events on a WebSocket (Node.js / browser) javascript

```javascript
// Server-sent events will come in as messages...
ws.on("message", function incoming(message) {
  // Message data payloads will need to be parsed from JSON:
  const serverEvent = JSON.parse(message.data)
  console.log(serverEvent);
});

// To send events, create a JSON-serializeable data structure that
// matches a client-side event (see API reference)
const event = {
  type: "response.create",
  response: {
    modalities: ["audio", "text"],
    instructions: "Give me a haiku about code.",
  }
};
ws.send(JSON.stringify(event));
```

Next steps

Now that you have a functioning WebSocket connection to the Realtime API, it's time to learn more about building applications with Realtime models.

Realtime model capabilities

Learn about sessions with a Realtime model, where you can send and receive audio, manage conversations, make one-off requests to the model, and execute function calls.

Event API reference

A complete listing of client and server events in the Realtime API