OpenAI Platform

# Realtime API with WebSocket

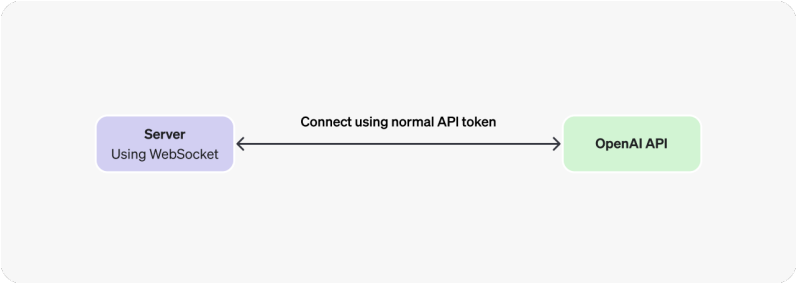
Copy page

Connect to the Realtime API using WebSockets on a server.



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.

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.




## Connect via WebSocket

Below are several examples of connecting via WebSocket to the Realtime API. In addition to using the WebSocket URL below, you will also need to pass an authentication header using your OpenAI

API key.

It is possible to use WebSocket in browsers with an ephemeral API token as shown in the WebRTC connection guide, but if you are connecting from a client like a browser or mobile app, WebRTC will be a more robust solution in most cases.

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

Connect using the ws modul… javascript

```javascript
import WebSocket from "ws";

const url = "wss://api.openai.com/v1/re
const ws = new WebSocket(url, {
  headers: {
    Authorization: "Bearer " + process.
  },
});

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

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

## Sending and receiving events

Realtime API sessions are managed using a combination of client-sent events emitted by you as the developer, and server-sent events created by the Realtime API to indicate session lifecycle events.

Over a WebSocket, you will both send and receive JSON-serialized events as strings of text, as in this Node.js example below (the same principles apply for other WebSocket libraries):

```
import WebSocket from "ws";

const url = "wss://api.openai.com/v1/re
const ws = new WebSocket(url, {
    headers: {
        Authorization: "Bearer " + proc
    },
});

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

    // Send client events over the WebS
    ws.send(
        JSON.stringify({
            type: "session.update",
            session: {
                type: "realtime",
                instructions: "Be extra
            },
        })
    );
});

// Listen for and parse server events
ws.on("message", function incoming(mess
    console.log(JSON.parse(message.toSt
});
```

The WebSocket interface is perhaps the lowest-level interface available to interact with a Realtime model, where you will be responsible for both sending and processing Base64-encoded audio chunks over the socket connection.

To learn how to send and receive audio over Websockets, refer to the Realtime conversations guide.