Realtime API with WebRTC Beta

Copy page

Use WebRTC to connect client-side applications to the Realtime API.

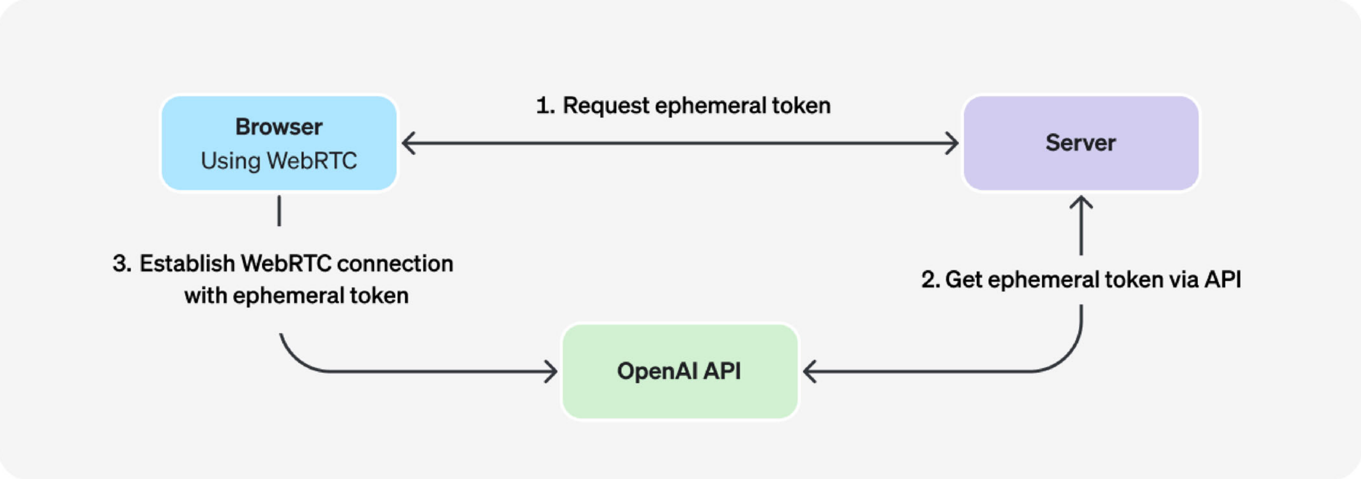
WebRTC is a powerful set of standard interfaces for building real-time applications. The OpenAI Realtime API supports connecting to realtime models through a WebRTC peer connection. Follow this guide to learn how to configure a WebRTC connection to the Realtime API.

Overview

In scenarios where you would like to connect to a Realtime model from an insecure client over the network (like a web browser), we recommend using the WebRTC connection method. WebRTC is better equipped to handle variable connection states, and provides a number of convenient APIs for capturing user audio inputs and playing remote audio streams from the model.

Connecting to the Realtime API from the browser should be done with an ephemeral API key, generated via the OpenAI REST API. The process for initializing a WebRTC connection is as follows (assuming a web browser client):

1. A browser makes a request to a developer-controlled server to mint an ephemeral API key.
2. The developer's server uses a standard API key to request an ephemeral key from the OpenAI REST API, and returns that new key to the browser. Note that ephemeral keys currently expire one minute after being issued.
3. The browser uses the ephemeral key to authenticate a session directly with the OpenAI Realtime API as a WebRTC peer connection.


Browser
Using WebRTC
1. Request ephemeral token
Server
3. Establish WebRTC connection with ephemeral token
2. Get ephemeral token via API
OpenAI API

ⓘ While it is technically possible to use a standard API key to authenticate WebRTC sessions, **this is a dangerous and insecure practice.** Standard API keys grant access to your full OpenAI API account, and should only be used in secure server-side environments. You should use ephemeral keys in client-side applications whenever possible.

Connection details

Connecting via WebRTC requires the following connection information:

URL	`https://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 EPHEMERAL_KEY` Substitute `EPHEMERAL_KEY` with an ephemeral API token - see below for details on how to generate one.

The following example shows how to initialize a WebRTC session (including the data channel to send and receive Realtime API events). It assumes you have already fetched an ephemeral API token (example server code for this can be found in the next section).

```
async function init() {
  // Get an ephemeral key from your server - see server code below
  const tokenResponse = await fetch("/session");
  const data = await tokenResponse.json();
  const EPHEMERAL_KEY = data.client_secret.value;

  // Create a peer connection
  const pc = new RTCPeerConnection();

  // Set up to play remote audio from the model
  const audioEl = document.createElement("audio");
  audioEl.autoplay = true;
  pc.ontrack = e => audioEl.srcObject = e.streams[0];

  // Add local audio track for microphone input in the browser
  const ms = await navigator.mediaDevices.getUserMedia({
    audio: true
  });
  pc.addTrack(ms.getTracks()[0]);

  // Set up data channel for sending and receiving events
  const dc = pc.createDataChannel("oai-events");
  dc.addEventListener("message", (e) => {
    // Realtime server events appear here!
    console.log(e);
  });

  // Start the session using the Session Description Protocol (SDP)
  const offer = await pc.createOffer();
  await pc.setLocalDescription(offer);

  const baseUrl = "https://api.openai.com/v1/realtime";
  const model = "gpt-4o-realtime-preview-2024-12-17";
  const sdpResponse = await fetch(`${baseUrl}?model=${model}`, {
    method: "POST",
    body: offer.sdp,
    headers: {
      Authorization: `Bearer ${EPHEMERAL_KEY}`,
      "Content-Type": "application/sdp"
    },
  });

  const answer = {
    type: "answer",
    sdp: await sdpResponse.text(),
  };
  await pc.setRemoteDescription(answer);
}

init();
```

The WebRTC APIs provide rich controls for handling media streams and input devices. For more guidance on building user interfaces on top of WebRTC, refer to the docs on MDN.

Creating an ephemeral token

To create an ephemeral token to use on the client-side, you will need to build a small server-side application (or integrate with an existing one) to make an OpenAI REST API request for an ephemeral key. You will use a standard API key to authenticate this request on your backend server.

Below is an example of a simple Node.js express server which mints an ephemeral API key using the REST API:

```
import express from "express";

const app = express();

// An endpoint which would work with the client code above - it returns
// the contents of a REST API request to this protected endpoint
app.get("/session", async (req, res) => {
  const r = await fetch("https://api.openai.com/v1/realtime/sessions", {
    method: "POST",
    headers: {
      "Authorization": `Bearer ${process.env.OPENAI_API_KEY}`,
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      model: "gpt-4o-realtime-preview-2024-12-17",
      voice: "verse",
    }),
  });
  const data = await r.json();

  // Send back the JSON we received from the OpenAI REST API
  res.send(data);
});

app.listen(3000);
```

You can create a server endpoint like this one on any platform that can send and receive HTTP requests. Just ensure that **you only use standard OpenAI API keys on the server, not in the browser.**

Sending and receiving events

To interact with the Realtime models, you will send and receive messages over the WebRTC data channel, and send and receive audio over media streams with the Realtime API as a connected peer. 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.

Here is how you can send and receive events over the data channel:

```
// Create a data channel from a peer connection
const dc = pc.createDataChannel("oai-events");

// Listen for server-sent events on the data channel - event data
// will need to be parsed from a JSON string
dc.addEventListener("message", (e) => {
  const realtimeEvent = JSON.parse(e.data);
  console.log(realtimeEvent);
});

// Send client events by serializing a valid client event to
// JSON, and sending it over the data channel
const responseCreate = {
  type: "response.create",
  response: {
    modalities: ["text"],
    instructions: "Write a haiku about code",
  },
};
dc.send(JSON.stringify(responseCreate));
```

Next steps

Now that you have a functioning WebRTC 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