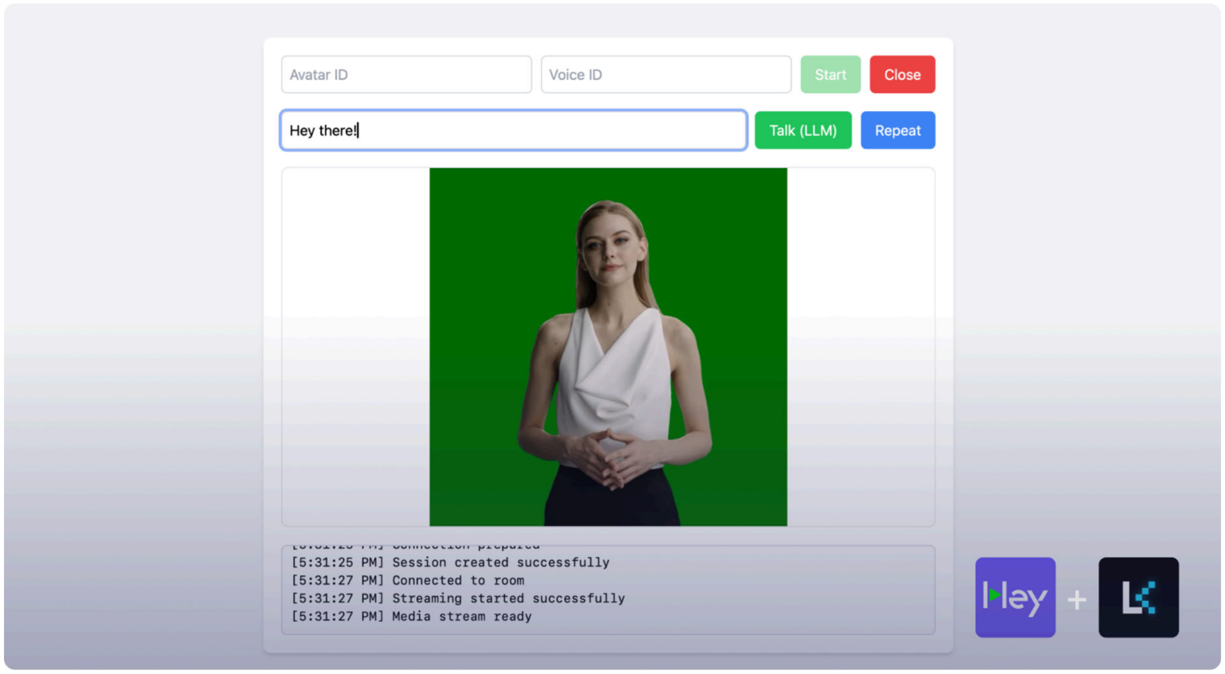
Streaming API v2 Integration: using LiveKit

This guide demonstrates how to use the Streaming API endpoints with version v2 and the LiveKit client SDK for real-time video streaming, which provides a simpler development interface.

> *For Node.js environments, we strongly recommend using the Streaming Avatar SDK package, as it offers a more robust solution. This guide focuses on the raw LiveKit implementation, intended for basic use cases as well as for developers who require more customization options or wish to integrate with existing LiveKit infrastructure.*



```
[5:31:25 PM] Session created successfully
[5:31:27 PM] Connected to room
[5:31:27 PM] Streaming started successfully
[5:31:27 PM] Media stream ready
```

Implementation Guide

Overview

In this guide:

- We will use the LiveKit CDN client for easy setup without requiring npm packages.
- Simplified WebSocket handling.
- Support for both Talk (LLM) and Repeat modes.
- Real-time event monitoring for both WebSocket and LiveKit events.

Prerequisites

- API Token from HeyGen
- Basic understanding of JavaScript and LiveKit

Step 1: Basic HTML Setup

Create an HTML file with the necessary elements and include the LiveKit JS Client SDK minified CDN version :

HTML

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://cdn.jsdelivr.net/npm/livekit-client/dist/livekit-client.umd.min.js"></script>
  </head>

  <body>
    <div>
      <div>
        <div>
          <button id="startBtn">Start</button>
          <button id="closeBtn">Close</button>
        </div>
      </div>
      <div>
        <input id="taskInput" type="text" placeholder="Enter text" />
        <button id="talkBtn">Talk</button>
      </div>
    </div>
    <video id="mediaElement" autoplay></video>

    <script>
      // JavaScript code goes here
    </script>
  </body>
</html>
```

Step 2. Configuration

JavaScript

```javascript
const API_CONFIG = {
  serverUrl: "https://api.heygen.com",
  token: "YOUR_API_TOKEN"
};

// Global state
let sessionInfo = null;
let room = null;
let mediaStream = null;

// DOM elements
const mediaElement = document.getElementById("mediaElement");
const taskInput = document.getElementById("taskInput");
```

Step 3. Core Implementation

3.1 Create and Start Session

JavaScript

```javascript
async function createSession() {
  // Create new session
  const response = await fetch(`${API_CONFIG.serverUrl}/v1/streaming.new`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${API_CONFIG.token}`
    },
    body: JSON.stringify({
      version: "v2",
      avatar_id: "YOUR_AVATAR_ID"
    })
  });

  sessionInfo = await response.json();

  // Start streaming
  await fetch(`${API_CONFIG.serverUrl}/v1/streaming.start`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${API_CONFIG.token}`
    },
    body: JSON.stringify({
      session_id: sessionInfo.session_id
    })
  });

  // Connect to LiveKit room
  room = new LiveKitClient.Room();
  await room.connect(sessionInfo.url, sessionInfo.access_token);

  // Handle media streams
  room.on(LiveKitClient.RoomEvent.TrackSubscribed, (track) => {
    if (track.kind === "video" || track.kind === "audio") {
      mediaStream.addTrack(track.mediaStreamTrack);
      mediaElement.srcObject = mediaStream;
    }
  });


}
```

- The LiveKit CDN version is accessed through the `LivekitClient` global namespace.
- All LiveKit classes and constants must be prefixed with `LivekitClient` (e.g., `LivekitClient.Room`, `LivekitClient.RoomEvent`

3.2 Send Text to Avatar

JavaScript

```
async function sendText(text) {
  await fetch(`${API_CONFIG.serverUrl}/v1/streaming.task`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${API_CONFIG.token}`
    },
    body: JSON.stringify({
      session_id: sessionInfo.session_id,
      text: text,
      task_type: "talk"  // or "repeat" to make avatar repeat exactly what you say
    })
  });
}
```

3.3 Close Session

JavaScript

```
async function closeSession() {
  await fetch(`${API_CONFIG.serverUrl}/v1/streaming.stop`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${API_CONFIG.token}`
    },
    body: JSON.stringify({
      session_id: sessionInfo.session_id
    })
  });

  if (room) {
    room.disconnect();
  }

  mediaElement.srcObject = null;
  sessionInfo = null;
  room = null;
  mediaStream = null;
}
```

Step 4. Event Listeners

JavaScript

```
// Start session
document.querySelector("#startBtn").addEventListener("click", async () => {
  await createSession();
});

// Close session
document.querySelector("#closeBtn").addEventListener("click", closeSession);

// Send text
document.querySelector("#talkBtn").addEventListener("click", () => {
  const text = taskInput.value.trim();
  if (text) {
    sendText(text);
    taskInput.value = "";
  }
});
```

Further Features

1. Task Types

The task endpoint supports different task types:

- `talk` : Avatar processes text through LLM before speaking
- `repeat` : Avatar repeats the exact input text

2. WebSocket Events

Monitor avatar state through WebSocket events:

JavaScript

```
const wsUrl = `wss://api.heygen.com/v1/ws/streaming.chat?session_id=${sessionId}&session_token=${token}&si
const ws = new WebSocket(wsUrl);
```

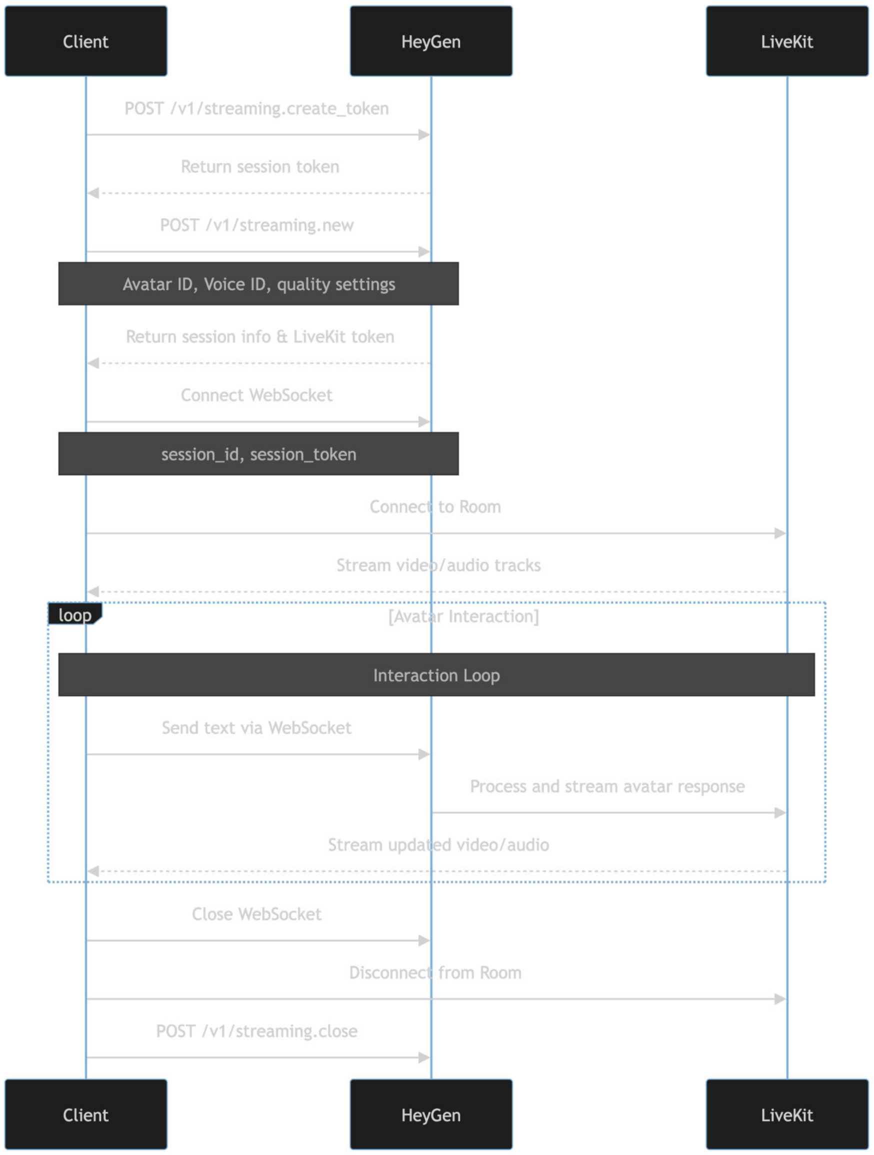
```
ws.addEventListener("message", (event) => {
  const data = JSON.parse(event.data);
  console.log("Event:", data);
});
```

3. LiveKit Room Events

Monitor room state and media tracks:

```JavaScript
room.on(LivekitClient.RoomEvent.DataReceived, (message) => {
  const data = new TextDecoder().decode(message);
  console.log("Room message:", JSON.parse(data));
});
```

System Flow



- Session setup (steps 1-3)
- Video streaming (step 4)

- Avatar interaction loop (step 5)
- Session closure (step 6)

Complete Demo Code

Here's a full working HTML & JS implementation combining all the components with a basic frontend:

HTML

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>HeyGen Streaming API LiveKit (V2)</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <script src="https://cdn.jsdelivr.net/npm/livekit-client/dist/livekit-client.umd.min.js"></script>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  </head>

  <body class="bg-gray-100 p-5 font-sans">
    <div class="max-w-3xl mx-auto bg-white p-5 rounded-lg shadow-md">
      <div class="flex flex-wrap gap-2.5 mb-5">
        <input
          id="avatarID"
          type="text"
          placeholder="Avatar ID"
          class="flex-1 min-w-[200px] p-2 border border-gray-300 rounded-md"
        />
        <input
          id="voiceID"
          type="text"
          placeholder="Voice ID"
          class="flex-1 min-w-[200px] p-2 border border-gray-300 rounded-md"
        />
        <button
          id="startBtn"
          class="px-4 py-2 bg-green-500 text-white rounded-md hover:bg-green-600 transition-colors disa
        >
          Start
        </button>
        <button
          id="closeBtn"
          class="px-4 py-2 bg-red-500 text-white rounded-md hover:bg-red-600 transition-colors"
        >
          Close
        </button>
      </div>

      <div class="flex flex-wrap gap-2.5 mb-5">
        <input
          id="taskInput"
          type="text"
          placeholder="Enter text for avatar to speak"
          class="flex-1 min-w-[200px] p-2 border border-gray-300 rounded-md"
        />
        <button
          id="talkBtn"
          class="px-4 py-2 bg-green-500 text-white rounded-md hover:bg-green-600 transition-colors"
        >
          Talk (LLM)
        </button>
        <button
          id="repeatBtn"
          class="px-4 py-2 bg-blue-500 text-white rounded-md hover:bg-blue-600 transition-colors"
        >
          Repeat
```

LiveKit Client SDKs

Here's the list of LiveKit client SDK repositories:

1. client-sdk-flutter: Dart, Flutter Client SDK for LiveKit
2. client-sdk-js: TypeScript, LiveKit browser client SDK (JavaScript)
3. client-sdk-swift: Swift, LiveKit Swift Client SDK for iOS, macOS, tvOS, and visionOS
4. client-sdk-android: Kotlin, LiveKit SDK for Android
5. client-sdk-unity: C#, Official Unity SDK for LiveKit
6. client-sdk-react-native: TypeScript, Official React Native SDK for LiveKit
7. client-sdk-react-native-expo-plugin: TypeScript, Expo plugin for the React Native SDK
8. client-sdk-unity-web: C#, Official LiveKit SDK for Unity WebGL
9. client-sdk-cpp: C++, C++ SDK for LiveKit

You can explore these repos for more detailed information.

Conclusion

The LiveKit-based implementation (v2) of HeyGen's Streaming API provides a streamlined approach to integrating interactive avatars into web applications. While this guide covers the basics of browser-side implementation, remember that for production Node.js environments, the @heygen/streaming-avatar npm package offers a more comprehensive solution.

Available Resources

- LiveKit Documentation
- HeyGen Streaming API Reference
- HeyGen Streaming Avatar SDK Reference

Support

For additional support or questions:

- Visit HeyGen Documentation: Discussions
- Contact our support team at support@heygen.com
- For API-specific inquiries: api@heygen.com
- Learn more about contacting support in our Help Center

Updated 9 days ago

← Streaming API v1 Integration: Raw WebRTC Approach **(deprecating 2/28/25)**

React Native Integration Guide with Streaming API + LiveKit →

Did this page help you? Yes No