---

title: WebSocket Transport

description: >-

Stream audio directly via WebSockets for real-time, bidirectional

communication

slug: calls/websocket-transport

---

# WebSocket Transport

Vapi's WebSocket transport enables real-time, bidirectional audio communication directly between your application and Vapi's AI assistants. Unlike traditional phone or web calls, this transport method lets you stream raw audio data instantly with minimal latency.

## Key Benefits

- \*\*Low Latency\*\*: Direct streaming ensures minimal delays.

- \*\*Bidirectional Streaming\*\*: Real-time audio flow in both directions.

- \*\*Easy Integration\*\*: Compatible with any environment supporting WebSockets.

- \*\*Flexible Audio Formats\*\*: Customize audio parameters such as sample rate.

- \*\*Automatic Sample Rate Conversion\*\*: Seamlessly handles various audio rates.

## Creating a WebSocket Call

To initiate a call using WebSocket transport:

```bash

curl 'https://api.vapi.ai/call' \

-H 'authorization: Bearer YOUR\_API\_KEY' \

-H 'content-type: application/json' \

--data-raw '{

"assistant": { "assistantId": "YOUR\_ASSISTANT\_ID" },

"transport": {

"provider": "vapi.websocket",

"audioFormat": {

"format": "pcm\_s16le",

"container": "raw",

"sampleRate": 16000

}

}

}'

```

### Sample API Response

```json

{

"id": "7420f27a-30fd-4f49-a995-5549ae7cc00d",

"assistantId": "5b0a4a08-133c-4146-9315-0984f8c6be80",

"type": "vapi.websocketCall",

"createdAt": "2024-09-10T11:14:12.339Z",

"updatedAt": "2024-09-10T11:14:12.339Z",

"orgId": "eb166faa-7145-46ef-8044-589b47ae3b56",

"cost": 0,

"status": "queued",

"transport": {

"provider": "vapi.websocket",

"websocketCallUrl": "wss://api.vapi.ai/7420f27a-30fd-4f49-a995-5549ae7cc00d/transport"

}

}

```

## Audio Format Configuration

When creating a WebSocket call, the audio format can be customized:

| Parameter | Description | Default |

|-------------|-------------------------|---------------------|

| `format` | Audio encoding format | `pcm\_s16le` (16-bit PCM) |

| `container` | Audio container format | `raw` (Raw PCM) |

| `sampleRate`| Sample rate in Hz | `16000` (16kHz) |

Currently, Vapi supports only raw PCM (`pcm\_s16le` with `raw` container). Additional formats may be supported in future updates.

<Note>

Vapi automatically converts sample rates as needed. You can stream audio at 8kHz, 44.1kHz, etc., and Vapi will handle conversions seamlessly.

</Note>

## Connecting to the WebSocket

Use the WebSocket URL from the response to establish a connection:

```javascript

const socket = new WebSocket("wss://api.vapi.ai/7420f27a-30fd-4f49-a995-5549ae7cc00d/transport");

socket.onopen = () => console.log("WebSocket connection opened.");

socket.onclose = () => console.log("WebSocket connection closed.");

socket.onerror = (error) => console.error("WebSocket error:", error);

```

## Sending and Receiving Data

The WebSocket supports two types of messages:

- \*\*Binary audio data\*\* (PCM, 16-bit signed little-endian)

- \*\*Text-based JSON control messages\*\*

### Sending Audio Data

```javascript

function sendAudioChunk(audioBuffer) {

if (socket.readyState === WebSocket.OPEN) {

socket.send(audioBuffer);

}

}

navigator.mediaDevices.getUserMedia({ audio: true }).then(stream => {

const audioContext = new AudioContext();

const source = audioContext.createMediaStreamSource(stream);

const processor = audioContext.createScriptProcessor(1024, 1, 1);

processor.onaudioprocess = (event) => {

const pcmData = event.inputBuffer.getChannelData(0);

const int16Data = new Int16Array(pcmData.length);

for (let i = 0; i < pcmData.length; i++) {

int16Data[i] = Math.max(-32768, Math.min(32767, pcmData[i] \* 32768));

}

sendAudioChunk(int16Data.buffer);

};

source.connect(processor);

processor.connect(audioContext.destination);

});

```

### Receiving Data

```javascript

socket.onmessage = (event) => {

if (event.data instanceof Blob) {

event.data.arrayBuffer().then(buffer => {

const audioData = new Int16Array(buffer);

playAudio(audioData);

});

} else {

try {

const message = JSON.parse(event.data);

handleControlMessage(message);

} catch (error) {

console.error("Failed to parse message:", error);

}

}

};

```

### Sending Control Messages

```javascript

function sendControlMessage(messageObj) {

if (socket.readyState === WebSocket.OPEN) {

socket.send(JSON.stringify(messageObj));

}

}

// Example: hangup call

function hangupCall() {

sendControlMessage({ type: "hangup" });

}

```

## Ending the Call

To gracefully end the WebSocket call:

```javascript

sendControlMessage({ type: "hangup" });

socket.close();

```

## Comparison: WebSocket Transport vs. Call Listen Feature

Vapi provides two WebSocket options:

| WebSocket Transport | Call Listen Feature |

|-------------------------------------|------------------------------------|

| Primary communication method | Secondary, monitoring-only channel |

| Bidirectional audio streaming | Unidirectional (listen-only) |

| Replaces phone/web as transport | Supplements existing calls |

| Uses `provider: "vapi.websocket"` | Accessed via `monitor.listenUrl` |

Refer to [Live Call Control](/calls/call-features) for more on the Call Listen feature.

<Warning>

When using WebSocket transport, phone-based parameters (`phoneNumber` or `phoneNumberId`) are not permitted. These methods are mutually exclusive.

</Warning>