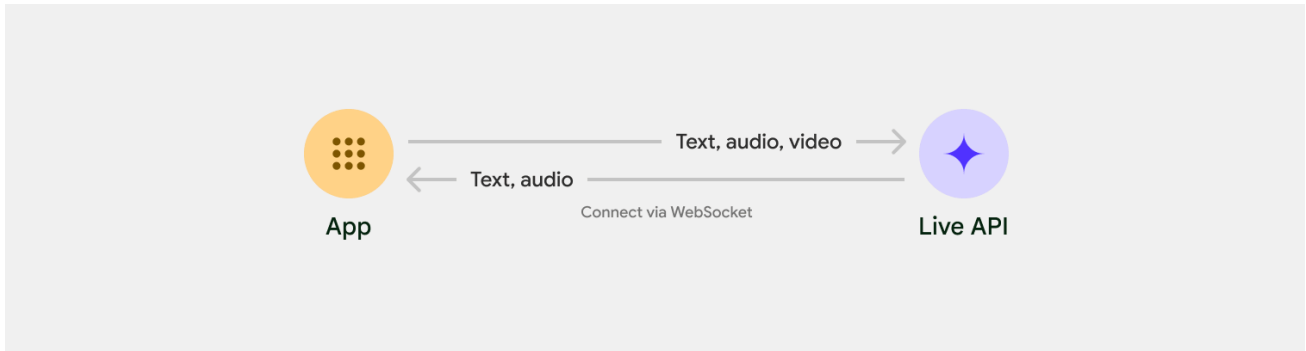


# Get started with Live API

The Live API enables low-latency, real-time voice and video interactions with Gemini. It processes continuous streams of audio, video, or text to deliver immediate, human-like spoken responses, creating a natural conversational experience for your users.



Live API offers a comprehensive set of features such as [Voice Activity Detection](/gemini-api/docs/live-guide#interruptions) (</gemini-api/docs/live-guide#interruptions>), [tool use and function calling](/gemini-api/docs/live-tools) (</gemini-api/docs/live-tools>), [session management](/gemini-api/docs/live-session) (</gemini-api/docs/live-session>) (for managing long running conversations) and [ephemeral tokens](/gemini-api/docs/ephemeral-tokens) (</gemini-api/docs/ephemeral-tokens>) (for secure client-sided authentication).

This page gets you up and running with examples and basic code samples.

Try the Live API in Google AI Studio  (<https://aistudio.google.com/live>)

## Choose an implementation approach

When integrating with Live API, you'll need to choose one of the following implementation approaches:

- **Server-to-server:** Your backend connects to the Live API using [WebSockets](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API) ([https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\\_API](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API)). Typically, your client sends stream data (audio, video, text) to your server, which then forwards it to the Live API.
- **Client-to-server:** Your frontend code connects directly to the Live API using [WebSockets](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API) ([https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\\_API](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API)) to stream data, bypassing your backend.

**Note:** Client-to-server generally offers better performance for streaming audio and video, since it bypasses the need to send the stream to your backend first. It's also easier to set up since you don't need to implement a proxy that sends data from your client to your server and then your server to the API. However, for production environments, in order to mitigate security risks, we recommend using ephemeral tokens (/gemini-api/docs/ephemeral-tokens) instead of standard API keys.

## Partner integrations

To streamline the development of real-time audio and video apps, you can use a third-party integration that supports the Gemini Live API over WebRTC or WebSockets.

### Pipecat by Daily

Create a real-time AI chatbot using Gemini Live and Pipecat.

(<https://docs.pipecat.ai/guides/features/gemini-live>)

### LiveKit

Use the Gemini Live API with LiveKit Agents.

(<https://docs.livekit.io/agents/models/realtime/plugins/gemini/>)

### Fishjam by Software Mansion

Create live video and audio streaming applications with Fishjam.

(<https://docs.fishjam.io/tutorials/gemini-live-integration>)

### Agent Development Kit (ADK)

Implement the Live API with Agent Development Kit (ADK).

(<https://google.github.io/adk-docs/streaming/>)

### Vision Agents by Stream

Build real-time vision and video AI applications with Vision Agents.

(<https://visionagents.ai/integrations/gemini>)

### Voximplant

Connect inbound and outbound calls to Live API with Voximplant.

(<https://voximplant.com/products/gemini-client>)

## Get started



Microphone stream

Audio file stream

This server-side example **streams audio from the microphone** and plays the returned audio. For complete end-to-end examples including a client application, see [Example applications](#) (#example-applications).

The input audio format should be in 16-bit PCM, 16kHz, mono format, and the received audio uses a sample rate of 24kHz.

PythonJavaScript (#javascript)  
(#python)

Install helpers for audio streaming. Additional system-level dependencies (e.g. **portaudio**) might be required. Refer to the [PyAudio docs](#) (<https://pypi.org/project/PyAudio/>) for detailed installation steps.

```
$ pip install pyaudio
```

✦ **Note: Use headphones.** This script uses the system default audio input and output, which often won't include echo cancellation. To prevent the model from interrupting itself, use headphones.

```
import asyncio
from google import genai
import pyaudio

client = genai.Client()

# --- pyaudio config ---
FORMAT = pyaudio.paInt16
CHANNELS = 1
SEND_SAMPLE_RATE = 16000
RECEIVE_SAMPLE_RATE = 24000
CHUNK_SIZE = 1024

pya = pyaudio.PyAudio()

# --- Live API config ---
MODEL = "gemini-2.5-flash-native-audio-preview-12-2025"
CONFIG = {
    "response_modalities": ["AUDIO"],
    "system_instruction": "You are a helpful and friendly AI assistant."
}

audio_queue_output = asyncio.Queue()
audio_queue_mic = asyncio.Queue(maxsize=5)
```

```

audio_stream = None

async def listen_audio():
    """Listens for audio and puts it into the mic audio queue."""
    global audio_stream
    mic_info = pya.get_default_input_device_info()
    audio_stream = await asyncio.to_thread(
        pya.open,
        format=FORMAT,
        channels=CHANNELS,
        rate=SEND_SAMPLE_RATE,
        input=True,
        input_device_index=mic_info["index"],
        frames_per_buffer=CHUNK_SIZE,
    )
    kwargs = {"exception_on_overflow": False} if __debug__ else {}
    while True:
        data = await asyncio.to_thread(audio_stream.read, CHUNK_SIZE, **
        await audio_queue_mic.put({"data": data, "mime_type": "audio/pcm

async def send_realtime(session):
    """Sends audio from the mic audio queue to the GenAI session."""
    while True:
        msg = await audio_queue_mic.get()
        await session.send_realtime_input(audio=msg)

async def receive_audio(session):
    """Receives responses from GenAI and puts audio data into the speaker
    while True:
        turn = session.receive()
        async for response in turn:
            if (response.server_content and response.server_content.model
                for part in response.server_content.model_turn.parts:
                    if part.inline_data and isinstance(part.inline_data.
                        audio_queue_output.put_nowait(part.inline_data.c

        # Empty the queue on interruption to stop playback
        while not audio_queue_output.empty():
            audio_queue_output.get_nowait()

async def play_audio():
    """Plays audio from the speaker audio queue."""
    stream = await asyncio.to_thread(
        pya.open,
        format=FORMAT,
        channels=CHANNELS,
        rate=RECEIVE_SAMPLE_RATE,
        output=True,

```

```

    )
    while True:
        bytestream = await audio_queue_output.get()
        await asyncio.to_thread(stream.write, bytestream)

async def run():
    """Main function to run the audio loop."""
    try:
        async with client.aio.live.connect(
            model=MODEL, config=CONFIG
        ) as live_session:
            print("Connected to Gemini. Start speaking!")
            async with asyncio.TaskGroup() as tg:
                tg.create_task(send_realtime(live_session))
                tg.create_task(listen_audio())
                tg.create_task(receive_audio(live_session))
                tg.create_task(play_audio())
    except asyncio.CancelledError:
        pass
    finally:
        if audio_stream:
            audio_stream.close()
        pya.terminate()
        print("\nConnection closed.")

if __name__ == "__main__":
    try:
        asyncio.run(run())
    except KeyboardInterrupt:
        print("Interrupted by user.")

```

## Example applications

Check out the following example applications that illustrate how to use Live API for end-to-end use cases:

- [Live audio starter app](https://aistudio.google.com/apps/bundled/live_audio?showPreview=true&showCode=true&showAssistant=false)  
([https://aistudio.google.com/apps/bundled/live\\_audio?showPreview=true&showCode=true&showAssistant=false](https://aistudio.google.com/apps/bundled/live_audio?showPreview=true&showCode=true&showAssistant=false))  
on AI Studio, using JavaScript libraries to connect to Live API and stream bidirectional audio through your microphone and speakers.
- See the [Partner integrations](#) (#partner-integrations) for additional examples and getting started guides.

## What's next

- Read the full Live API [Capabilities](/gemini-api/docs/live-guide) (/gemini-api/docs/live-guide) guide for key capabilities and configurations; including Voice Activity Detection and native audio features.
- Read the [Tool use](/gemini-api/docs/live-tools) (/gemini-api/docs/live-tools) guide to learn how to integrate Live API with tools and function calling.
- Read the [Session management](/gemini-api/docs/live-session) (/gemini-api/docs/live-session) guide for managing long running conversations.
- Read the [Ephemeral tokens](/gemini-api/docs/ephemeral-tokens) (/gemini-api/docs/ephemeral-tokens) guide for secure authentication in [client-to-server](#) (#implementation-approach) applications.
- For more information about the underlying WebSockets API, see the [WebSockets API reference](/api/live) (/api/live).

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (https://www.apache.org/licenses/LICENSE-2.0). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-12-22 UTC.