OpenAI Platform

Using realtime models

Copy page

Use realtime models and prompting effectively.

Realtime models are post-trained for specific customer use cases. In response to your feedback, the latest speech-to-speech model works differently from previous models. Use this guide to understand and get the most out of it.

Meet the models

Our most advanced speech-to-speech model is gpt-realtime.

This model shows improvements in following complex instructions, calling tools, and producing speech that sounds natural and expressive. For more information, see the announcement blog post.

Update your session to use a prompt

After you initiate a session over WebRTC, WebSocket, or SIP, the client and model are connected. The server will send a session.created event to confirm. Now it's a matter of prompting.

Basic prompt update

1. Create a basic audio prompt in the dashboard.

   If you don't know where to start, experiment with the prompt fields until you find something interesting. You can always manage, iterate on, and version your prompts later.

2. Update your realtime session to use the prompt you created. Provide its prompt ID in a `session.update` client event:

Update the system instruct… javascript

```javascript
const event = {
  type: "session.update",
  session: {
      type: "realtime",
      model: "gpt-realtime",
      // Lock the output to audio (add
      output_modalities: ["audio"],
      audio: {
        input: {
          format: "pcm16",
          turn_detection: { type: "sema
        },
        output: {
          format: "g711_ulaw",
          voice: "alloy",
          speed: 1.0
        }
      },
      // Use a server-stored prompt by
      prompt: {
        id: "pmpt_123",          // you
        // version: "89",        // opt
        variables: {
          city: "Paris"          // exa
        }
      },
      // You can still set direct sessi
      instructions: "Speak clearly and
  },
};

// WebRTC data channel and WebSocket bo
dataChannel.send(JSON.stringify(event))
```

When the session's updated, the server emits a session.updated event with the new state of the session. You can update the session any time.

Changing prompt mid-call

To update the session mid-call (to swap prompt version or variables, or override instructions), send the update over the same data channel you're using:

```
// Example: switch to a specific prompt
dc.send(JSON.stringify({
  type: "session.update",
  session: {
    prompt: {
      id: "pmpt_123",
      version: "89",
      variables: {
        city: "Berlin"
      }
    }
  }
}));

// Example: override instructions (note
dc.send(JSON.stringify({
  type: "session.update",
  session: {
    instructions: "Speak faster and kee
  }
}));
```

Prompting gpt-realtime

Here are top tips for prompting the realtime speech-to-speech model. For a more in-depth guide to prompting, see the realtime prompting cookbook
.

General usage tips

Iterate relentlessly. Small wording changes can make or break behavior.

Example: Swapping “inaudible” → “unintelligible” improved noisy input handling.

Use bullets over paragraphs. Clear, short bullets outperform long paragraphs.

Guide with examples. The model strongly follows onto sample phrases.

Be precise. Ambiguity and conflicting instructions degrade performance, similar to GPT-5.

Control language. Pin output to a target language if you see drift.

Reduce repetition. Add a variety rule to reduce robotic phrasing.

Use all caps for emphasis: Capitalize key rules to makes them stand out to the model.

Convert non-text rules to text: The model responds better to clearly written text.

Example: Instead of writing, "IF x > 3 THEN ESCALATE", write, "IF MORE THAN THREE FAILURES THEN ESCALATE."

Structure your prompt

Organize your prompt to help the model understand context and stay consistent across turns.

Use clear, labeled sections in your system prompt so the model can find and follow them. Keep each section focused on one thing.

```
# Role & Objective        — who you are
# Personality & Tone      — the voice ar
# Context                 — retrieved co
# Reference Pronunciations — phonetic gu
# Tools                   — names, usage
# Instructions / Rules    — do’s, don’ts
# Conversation Flow       — states, goal
# Safety & Escalation     — fallback and
```

This format also makes it easier for you to iterate and modify problematic sections.

To make this system prompt your own, add domain-specific sections (e.g., Compliance, Brand Policy) and remove sections you don’t need. In each section, provide instructions and other information for the model to respond correctly. See specifics below.

Practical tips for prompting realtime models

Here are 10 tips for creating effective, consistently performing prompts with gpt-realtime. These are just an overview. For more details and full system prompt examples, see the realtime prompting cookbook.

1. Be precise. Kill conflicts.

The new realtime model is very good at instruction following. However, that also means small wording changes or unclear instructions can shift behavior in meaningful ways. Inspect and iterate on your system prompt to try different phrasing and fix instruction contradictions.

In one experiment we ran, changing the word "inaudible" to "unintelligble" in instructions for handling noisy inputs significantly improved the model's performance.

After your first attempt at a system prompt, have an LLM review it for ambiguity or conflicts.

2. Bullets > paragraphs.

Realtime models follow short bullet points better than long paragraphs.

Before (harder to follow):

```
When you can’t clearly hear the user, don’
```

After (easier to follow):

```
Only respond to clear audio or text.

If audio is unclear/partial/noisy/silent

Continue in the same language as the use
```

3. Handle unclear audio.

The realtime model is good at following instructions on how to handle unclear audio. Spell out what to do when audio isn’t usable.

```
## Unclear audio
- Always respond in the same language t
- Only respond to clear audio or text.
- If the user's audio is not clear (e.g

Sample clarification phrases (parameter

- “Sorry, I didn’t catch that—could you
- “There’s some background noise. Pleas
- “I only heard part of that. What did
```

4. Constrain the model to one language.

If you see the model switching languages in an unhelpful way, add a dedicated "Language" section in your prompt. Make sure it doesn’t conflict with other rules. By default, mirroring the user’s language works well.

Here's a simple way to mirror the user's language:

```
## Language
Language matching: Respond in the same l
For non-English, start with the same sta
```

Here's an example of an English-only constraint:

```
## Language
- The conversation will be only in Engli
- Do not respond in any other language,
- If the user speaks another language, p
```

In a language teaching application, your language and conversation sections might look like this:

```
## Language
### Explanations
Use English when explaining grammar, voc

### Conversation
Speak in French when conducting practice
```

You can also control dialect for a more consistent personality:

```
## Language
Response only in argentine spanish.
```

5. Provide sample phrases and flow snippets.

The model learns style from examples. Give short, varied samples for common conversation moments.

For example, you might give this high-level shape of conversation flow to the model:

```
Greeting → Discover → Verify → Diagnose →
```

And then provide prompt guidance for each section. For example, here's how you might instruct for the greeting section:

```
## Conversation flow — Greeting
Goal: Set tone and invite the reason fo

How to respond:
- Identify as ACME Internet Support.
- Keep it brief; invite the caller's go

Sample phrases (vary, don't always reus
- "Thanks for calling ACME Internet—how
- "You've reached ACME Support. What's
- "Hi there—tell me what you'd like hel

Exit when: Caller states an initial goa
```

6. Avoid robotic repetition.

If responses sound repetitive or robotic, include an explicit variety instruction. This can sometimes happen when using sample phrases.

```
## Variety
- Do not repeat the same sentence twice. V
```

7. Use capitalized text to emphasize instructions.

Like many LLMs, using capitalization for important rules can help the model to understand and follow those rules. It's also helpful to convert non-text rules (such as numerical conditions) into text before capitalization.

Instead of:

```
## Rules
- If [func.return_value] > 0, respond 1 to
```

Use:

```
## Rules
- IF [func.return_value] IS BIGGER THAN 0,
```

8. Help the model use tools.

The model's use of tools can alter the experience—how much they rely on user confirmation vs. taking action, what they say while they make the tool call, which rules they follow for each specific tool, etc.

One way to prompt for tool usage is to use preambles. Good preambles instruct the model to give the user some feedback about what it's doing before it makes the tool call, so the user always knows what's going on.

Here's an example:

```
# Tools
- Before any tool call, say one short line
```

You can include sample phrases for preambles to add variety and better tailor to your use case.

There are several other ways to improve the model's behavior when performing tool calls and keeping the conversation going with the user. Ideally, the model is calling the right tools proactively, checking for confirmation for any important write actions, and keeping the user informed along the way. For more specifics, see the realtime prompting cookbook.

9. Use LLMs to improve your prompt.

LLMs are great at finding what's going wrong in your prompt. Use ChatGPT or the API to get a model's review of your current realtime prompt and get help improving it.

Whether your prompt is working well or not, here's a prompt you can run to get a model's review:

```
## Role & Objective
You are a **Prompt-Critique Expert**.
Examine a user-supplied LLM prompt and

## Instructions
Review the prompt that is meant for an
- Ambiguity: Could any wording be inter
- Lacking Definitions: Are there any cl
- Conflicting, missing, or vague instru
- Unstated assumptions: Does the prompt

## Do **NOT** list issues of the follow
- Invent new instructions, tool calls,
- Issues that you are not sure about.

## Output Format

# Issues
- Numbered list; include brief quote sn

# Improvements
- Numbered list; provide the revised li

# Revised Prompt
- Revised prompt where you have applied
```

Use this template as a starting point for troubleshooting a recurring issue:

```
Here's my current prompt to an LLM:
[BEGIN OF CURRENT PROMPT]
{CURRENT_PROMPT}
[END OF CURRENT PROMPT]

But I see this issue happening from the
[BEGIN OF ISSUE]
{ISSUE}
[END OF ISSUE]
Can you provide some variants of the pr
```

10. Help users resolve issues faster.

Two frustrating user experiences are slow, mechanical voice agents and the inability to escalate. Help users faster by providing instructions in your system prompt for speed and escalation.

In the personality and tone section of your system prompt, add pacing instructions to get the model to quicken its support:

```
# Personality & Tone
## Personality
Friendly, calm and approachable expert

## Tone
Tone: Warm, concise, confident, never f

## Length
2–3 sentences per turn.

## Pacing
Deliver your audio response fast, but d
```

Often with realtime voice agents, having a reliable way to escalate to a human is important. In a safety and escalation section, modify the instructions on WHEN to escalate depending on your use case. Here's an example:

```
# Safety & Escalation
When to escalate (no extra troubleshoot
- Safety risk (self-harm, threats, hara
- User explicitly asks for a human
- Severe dissatisfaction (e.g., “extrem
- **2** failed tool attempts on the sam
- Out-of-scope or restricted (e.g., rea

What to say at the same time of calling
- “Thanks for your patience—**I’m conne
- Then call the tool: `escalate_to_huma

Examples that would require escalation:
- “This is the third time the reset dic
- “I am extremely frustrated!”
```

Further reading

This guide is long but not exhaustive! For more in a specific area, see the following resources:

- Realtime prompting cookbook: Full prompt examples and a deep dive into when and how to use them
- Inputs and outputs: Text and audio input requirements and output options
- Managing conversations: Learn to manage a conversation for the duration of a realtime session
- Webhooks and server-side controls: Create a sideband channel to separate sensitive server-side logic from an untrusted client

Function calling: How to call functions in your realtime app

MCP servers: How to use MCP servers to access additional tools in realtime apps

Realtime transcription: How to transcribe audio with the Realtime API

Voice agents: A quickstart for building a voice agent with the Agents SDK