Speech

- Setup
- Speak from the terminal
  - Save audio to a file
- Generate speech with the API
  - Audio formats
  - Limits

Speech models read text aloud. Ollama streams the audio as it is generated, so playback can start before the whole text has been spoken.

Setup

Start Ollama v0.40.9 or later, then download Phonon:

```
ollama pull phonon
```

Phonon uses Gradium's 31M English checkpoint and comes with Gradium's English voices, such as `Zoey`, `Easton`, and `Freya`. Without a voice, it speaks with the checkpoint's own voice, `default`. `ollama show phonon` lists every voice. Voice names work in any case and with or without accents.

Speak from the terminal

`ollama run` plays the audio through your computer's speakers:

```
# Speak some text
ollama run phonon "Hello from Ollama."

# Choose a voice
ollama run phonon --voice Zoey "Hi, this is Zoey."
```

```
# Speak text from another program
echo "Text from another program works too." | ollama run phonon

# Speak a chat model's answer
ollama run gemma4 --hidethinking "Tell me a fun fact about owls in two sentences." | ollama
run phonon
```

Piped text is spoken as it arrives, so a chat model's answer starts playing after its first sentence or two.

Run `ollama run phonon` without any text to type lines at a `>>>` prompt. Each line is spoken when you press Enter. Press Ctrl+C to stop speaking, and type `/bye` or press Ctrl+D to exit.

Audio plays with your system's built-in player: `afplay` on macOS, the first of `paplay`, `pw-play`, `aplay`, or `ffplay` that is installed on Linux, and PowerShell on Windows.

Save audio to a file

When the output is redirected, `ollama run` writes a WAV file instead of playing it. This also works on servers without speakers:

```
ollama run phonon "Save this to a file." > speech.wav
```

To hear a remote server, run `ollama` on your own computer and point it at the server. The audio plays where the command runs:

```
OLLAMA_HOST=my-server:11434 ollama run phonon "Hello from the server."
```

Generate speech with the API

Use the OpenAI-compatible `/v1/audio/speech` endpoint:

```shell curl http://localhost:11434/v1/audio/speech \ -H "Content-Type: application/json" \ -d '{ "model": "phonon", "input": "Hello from Ollama.", "voice": "default" }' --output speech.wav ```
```python from openai import OpenAI
```
client = OpenAI(base_url="http://localhost:11434/v1", api_key="ollama")

with client.audio.speech.with_streaming_response.create(
    model="phonon",
```

```
    voice="default",
    input="Hello from Ollama.",
    response_format="wav",
) as response:
    response.stream_to_file("speech.wav")
```

```javascript import fs from 'node:fs' import OpenAI from 'openai'
const openai = new OpenAI({ baseURL: 'http://localhost:11434/v1', apiKey: 'ollama' })

const response = await openai.audio.speech.create({
  model: 'phonon',
  voice: 'default',
  input: 'Hello from Ollama.',
  response_format: 'wav',
})
fs.writeFileSync('speech.wav', Buffer.from(await response.arrayBuffer()))
```

The response streams while audio is generated, so you can start playing it right away:

```
curl -sN http://localhost:11434/v1/audio/speech \
  -H "Content-Type: application/json" \
  -d '{"model": "phonon", "input": "Playback starts while the rest is still being generated."}' \
  | ffplay -nodisp -autoexit -loglevel error -
```

Audio formats

`response_format`	Output
`wav` (default)	A WAV file: 16-bit, mono, 24 kHz. The header is written before the length is known, so its size fields hold the maximum value.
`pcm`	Raw 16-bit little-endian samples, mono, 24 kHz, with no header.

Limits

- `input` can be up to 4,096 characters per request. `ollama run` splits longer text at paragraph and sentence breaks for you.
- Prices, numbers, symbols, email addresses, and URLs are spelled out before speaking, so `$4.59` is read as "4 dollars and 59 cents". Times and numeric dates are read as written, so write dates out (`October 23`, not `10/23`) for a natural reading. Emoji, bullet characters, and markdown markers such as `*` and `#` are skipped.
- Options the model doesn't support, such as other formats, speeds, or `instructions`, return a `400` error instead of being ignored. An unknown voice returns a `400` error that lists the available voices.
- Speech models only generate speech: chat, completion, and embedding requests to them return a `400` error.

`ollama show phonon` and `/api/show` list the model's capabilities as `speech`, and its voices (`voices` in `/api/show`). See OpenAI compatibility for every supported field.

□□□ #2
□ □□□ □□□ 2026-10-31 13:16:44 UTC
□ □□□ □□□ 2026-10-31 13:16:45 UTC