OpenAI Platform

Image generation

Copy page

Allow models to generate or edit images.

The image generation tool allows you to generate images using a text prompt, and optionally image inputs. It leverages the GPT Image model, and automatically optimizes text inputs for improved performance.

To learn more about image generation, refer to our dedicated image generation guide.

Usage

When you include the `image_generation` tool in your request, the model can decide when and how to generate images as part of the conversation, using your prompt and any provided image inputs.

The `image_generation_call` tool call result will include a base64-encoded image.

Generate an image python

```python
from openai import OpenAI
import base64

client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input="Generate an image of gray ta
    tools=[{"type": "image_generation"}
)

# Save the image to a file
image_data = [
    output.result
    for output in response.output
    if output.type == "image_generatior
]

if image_data:
    image_base64 = image_data[0]
    with open("otter.png", "wb") as f:
        f.write(base64.b64decode(image_
```

You can provide input images using file IDs or base64 data.

To force the image generation tool call, you can set the parameter `tool_choice` to `{"type": "image_generation"}`.

Tool options

You can configure the following output options as parameters for the image generation tool:

Size: Image dimensions (e.g., 1024×1024, 1024×1536)

Quality: Rendering quality (e.g. low, medium, high)

Format: File output format

Compression: Compression level (0-100%) for JPEG and WebP formats

Background: Transparent or opaque

`size`, `quality`, and `background` support the `auto` option, where the model will automatically select the best option based on the prompt.

For more details on available options, refer to the image generation guide.

Revised prompt

When using the image generation tool, the mainline model (e.g. `gpt-4.1`) will automatically revise your prompt for improved performance.

You can access the revised prompt in the `revised_prompt` field of the image generation call:

```
{
  "id": "ig_123",
  "type": "image_generation_call",
  "status": "completed",
  "revised_prompt": "A gray tabby cat hu
  "result": "..."
}
```

Prompting tips

Image generation works best when you use terms like "draw" or "edit" in your prompt.

For example, if you want to combine images, instead of saying "combine" or "merge", you can say something like "edit the first image by adding this element from the second image".

Multi-turn editing

You can iteratively edit images by referencing previous response or image IDs. This allows you to refine images across multiple turns in a conversation.

Using previous response ID Using image ID

Multi-turn image generation python

```python
from openai import OpenAI
import base64

client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input="Generate an image of gray ta
    tools=[{"type": "image_generation"}
)

image_data = [
    output.result
    for output in response.output
    if output.type == "image_generatior
]

if image_data:
    image_base64 = image_data[0]

    with open("cat_and_otter.png", "wb"
        f.write(base64.b64decode(image_


# Follow up

response_fwup = client.responses.create
    model="gpt-5",
    previous_response_id=response.id,
    input="Now make it look realistic",
    tools=[{"type": "image_generation"}
)

image_data_fwup = [
    output.result
    for output in response_fwup.output
    if output.type == "image_generatior
]

if image_data_fwup:
    image_base64 = image_data_fwup[0]
    with open("cat_and_otter_realistic.
        f.write(base64.b64decode(image_
```

Streaming

The image generation tool supports streaming partial images as the final result is being generated. This provides faster visual feedback for users and improves perceived latency.

You can set the number of partial images (1-3) with the `partial_images` parameter.

Stream an image python

```python
from openai import OpenAI
import base64

client = OpenAI()

stream = client.images.generate(
    prompt="Draw a gorgeous image of a
    model="gpt-image-1",
    stream=True,
    partial_images=2,
)

for event in stream:
    if event.type == "image_generation.
        idx = event.partial_image_index
        image_base64 = event.b64_json
        image_bytes = base64.b64decode(
        with open(f"river{idx}.png", "w
            f.write(image_bytes)
```

Supported models

The image generation tool is supported for the following models:

- `gpt-4o`
- `gpt-4o-mini`
- `gpt-4.1`
- `gpt-4.1-mini`
- `gpt-4.1-nano`
- `o3`

The model used for the image generation process is always `gpt-image-1`, but these models can be used as the mainline model in the Responses API as they can reliably call the image generation tool when needed.