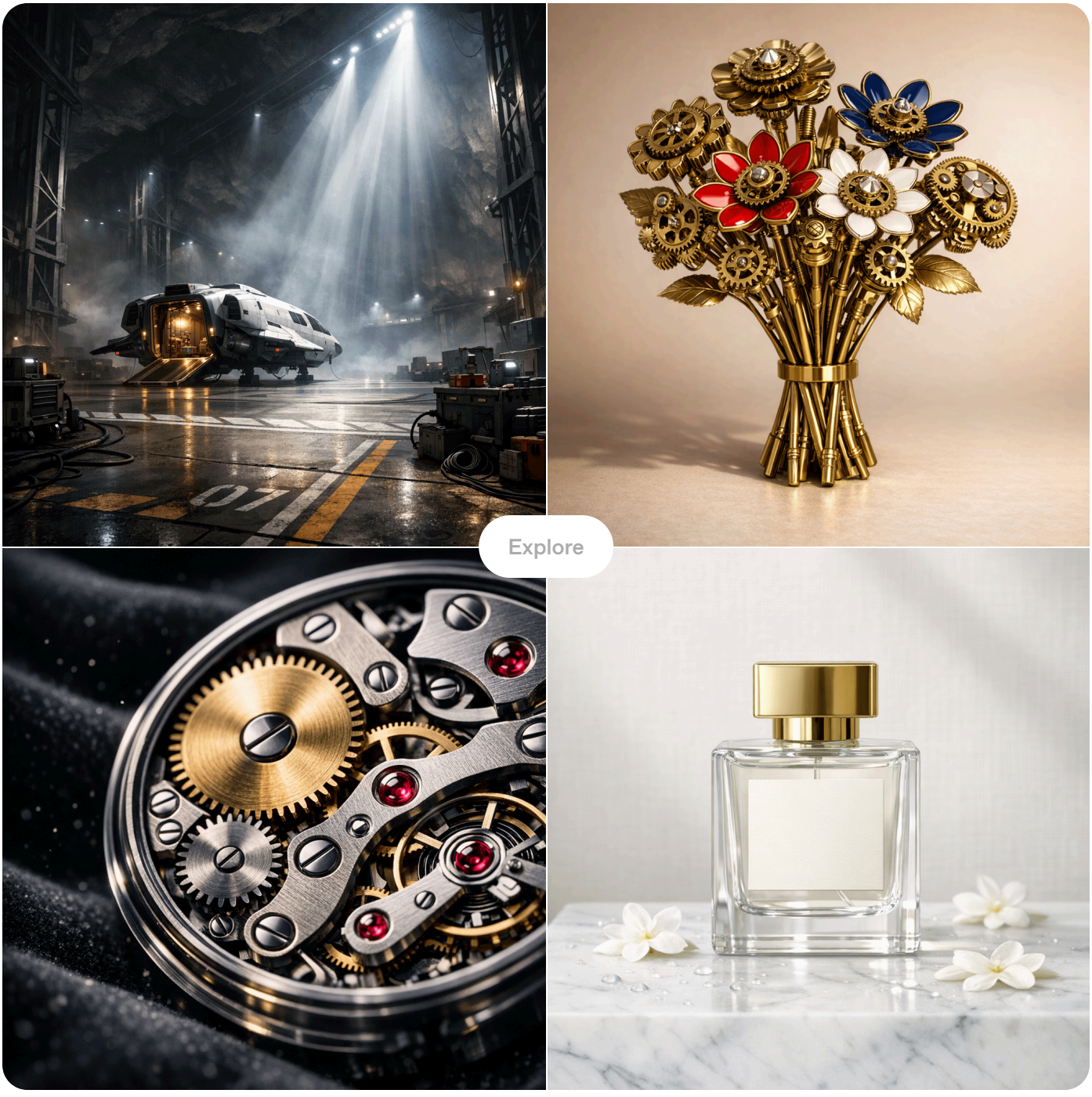
OpenAI Platform

Copy page

GPT-Image

Image generation

Learn how to generate or edit images.



Overview

The OpenAI API lets you generate and edit images from text prompts, using GPT Image or DALL·E models. You can access image generation capabilities through two APIs:

Image API

The Image API provides three endpoints, each with distinct capabilities:

Generations: Generate images from scratch based on a text prompt

Edits: Modify existing images using a new prompt, either partially or entirely

Variations: Generate variations of an existing image (available with DALL·E 2 only)

This API supports GPT Image models (`gpt-image-1.5` , `gpt-image-1` , and `gpt-image-1-mini`) as well as `dall-e-2` and `dall-e-3` .

Responses API

The Responses API allows you to generate images as part of conversations or multi-step flows. It supports image generation as a built-in tool, and accepts image inputs and outputs within context.

Compared to the Image API, it adds:

Multi-turn editing: Iteratively make high fidelity edits to images with prompting

Flexible inputs: Accept image File IDs as input images, not just bytes

The image generation tool in responses uses GPT Image models (`gpt-image-1.5` , `gpt-image-1` , and `gpt-image-1-mini`). When using `gpt-image-1.5` and `chatgpt-image-latest` with the Responses API, you can optionally set the `action` parameter, detailed below. For a list of mainline models that support calling this tool, refer to the supported models below.

Choosing the right API

If you only need to generate or edit a single image from one prompt, the Image API is your best choice.

If you want to build conversational, editable image experiences with GPT Image, go with the Responses API.

Both APIs let you customize output — adjust quality, size, format, compression, and enable transparent backgrounds.

Model comparison

Our latest and most advanced model for image generation is `gpt-image-1.5`, a natively multimodal language model, part of the GPT Image family.

GPT Image models include `gpt-image-1.5` (state of the art), `gpt-image-1`, and `gpt-image-1-mini`. They share the same API surface, with `gpt-image-1.5` offering the best overall quality.

We recommend using `gpt-image-1.5` for the best experience, but if you are looking for a more cost-effective option and image quality isn't a priority, you can use `gpt-image-1-mini`.

You can also use specialized image generation models—DALL·E 2 and DALL·E 3—with the Image API, but please note these models are now deprecated and we will stop supporting them on 05/12, 2026.

MODEL	ENDPOINTS	USE CASE
DALL·E 2	Image API: Generations, Edits, Variations	Lower cost, concurrent requests, inpainting (image editing with a mask)
DALL·E 3	Image API: Generations only	Higher image quality than DALL·E 2, support for larger resolutions
GPT Image	Image API: Generations, Edits – Responses API (as part of the image generation tool)	Superior instruction following, text rendering, detailed editing, real-world knowledge

This guide focuses on GPT Image, but you can also switch to the docs for DALL·E 2 and DALL·E 3.

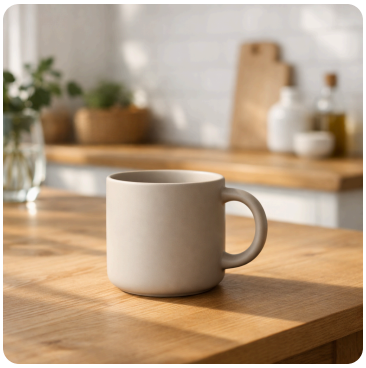
> To ensure this model is used responsibly, you may need to complete the API Organization Verification from your developer console before using GPT Image models, including `gpt-image-1.5`, `gpt-image-1`, and `gpt-image-1-mini`.

Generate Images

You can use the image generation endpoint to create images based on text prompts, or the image generation tool in the Responses API to generate images as part of a conversation.

To learn more about customizing the output (size, quality, format, transparency), refer to the customize image output section below.

You can set the `n` parameter to generate multiple images at once in a single request (by default, the API returns a single image).



Responses API | Image API

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 tabby cat hugging an otter with an orange
    tools=[{"type": "image_generation"}],
)

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

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

Multi-turn image generation

With the Responses API, you can build multi-turn conversations involving image generation either by providing image generation calls outputs within context (you can also just use the image ID), or by using the `previous_response_id` parameter. This makes it easy to iterate on images across multiple turns—refining prompts, applying new instructions, and evolving the visual output as the conversation progresses.

Generate vs Edit

With the Responses API you can choose whether to generate a new image or edit one already in the conversation. The optional `action` parameter (supported on `gpt-image-1.5` and `chatgpt-image-latest`) controls this behavior: keep `action: "auto"` to let the model decide (recommended), set `action: "generate"` to always create a new image, or set `action: "edit"` to force editing (requires an image in context).

Force image creation with action — python

```python
from openai import OpenAI
import base64

client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input="Generate an image of gray tabby cat hugging an otter with an orange
    tools=[{"type": "image_generation", "action": "generate"}],
)

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

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

If you force `edit` without providing an image in context, the call will return an error. Leave `action` at `auto` to have the model decide when to generate or edit.

When `action` is set to `auto`, the `image_generation_call` result includes an `action` field so you can see whether the model generated a new image or edited one already in context:

```
{
    "id": "ig_123...",
    "type": "image_generation_call",
    "status": "completed",
```

```
    "background": "opaque",
    "output_format": "jpeg",
    "quality": "medium",
    "result": "/9j/4...",
    "revised_prompt": "...",
    "size": "1024x1024",
    "action": "generate"
}
```

Using previous response ID | Using image ID

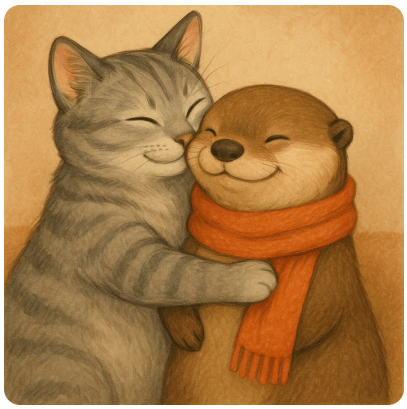
Multi-turn image generation — python

```python
from openai import OpenAI
import base64

client = OpenAI()

response = client.responses.create(
    model="gpt-5",
    input="Generate an image of gray tabby cat hugging an otter with an orange
    tools=[{"type": "image_generation"}],
)

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

if image_data:
    image_base64 = image_data[0]

    with open("cat_and_otter.png", "wb") as f:
        f.write(base64.b64decode(image_base64))


# 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_generation_call"
]

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

Result

"Generate an image of gray tabby cat hugging an otter with an orange scarf"



"Now make it look realistic"

Streaming

The Responses API and Image API support streaming image generation. This allows you to stream partial images as they are generated, providing a more interactive experience.

You can adjust the `partial_images` parameter to receive 0-3 partial images.

If you set `partial_images` to 0, you will only receive the final image.

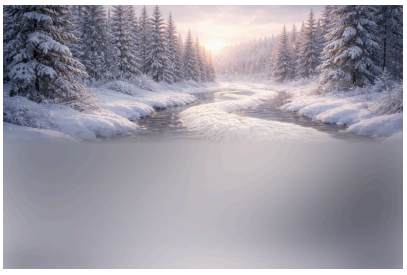
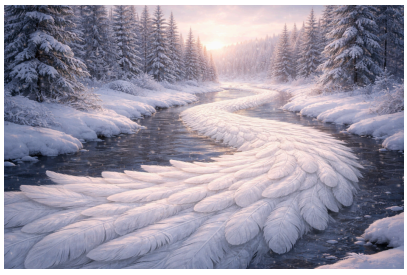
For values larger than zero, you may not receive the full number of partial images you requested if the full image is generated more quickly.

Responses API | Image API

Stream an image — python

```python
from openai import OpenAI
import base64

client = OpenAI()

stream = client.responses.create(
    model="gpt-4.1",
    input="Draw a gorgeous image of a river made of white owl feathers, snaking
    stream=True,
    tools=[{"type": "image_generation", "partial_images": 2}],
)

for event in stream:
    if event.type == "response.image_generation_call.partial_image":
        idx = event.partial_image_index
        image_base64 = event.partial_image_b64
        image_bytes = base64.b64decode(image_base64)
        with open(f"river{idx}.png", "wb") as f:
            f.write(image_bytes)
```

Result

PARTIAL 1	PARTIAL 2	FINAL IMAGE
		

Prompt: Draw a gorgeous image of a river made of white owl feathers, snaking its way through a serene winter landscape

Revised prompt

When using the image generation tool in the Responses API, 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 hugging an otter. The otter is wearing a
    "result": "..."
}
```

Edit Images

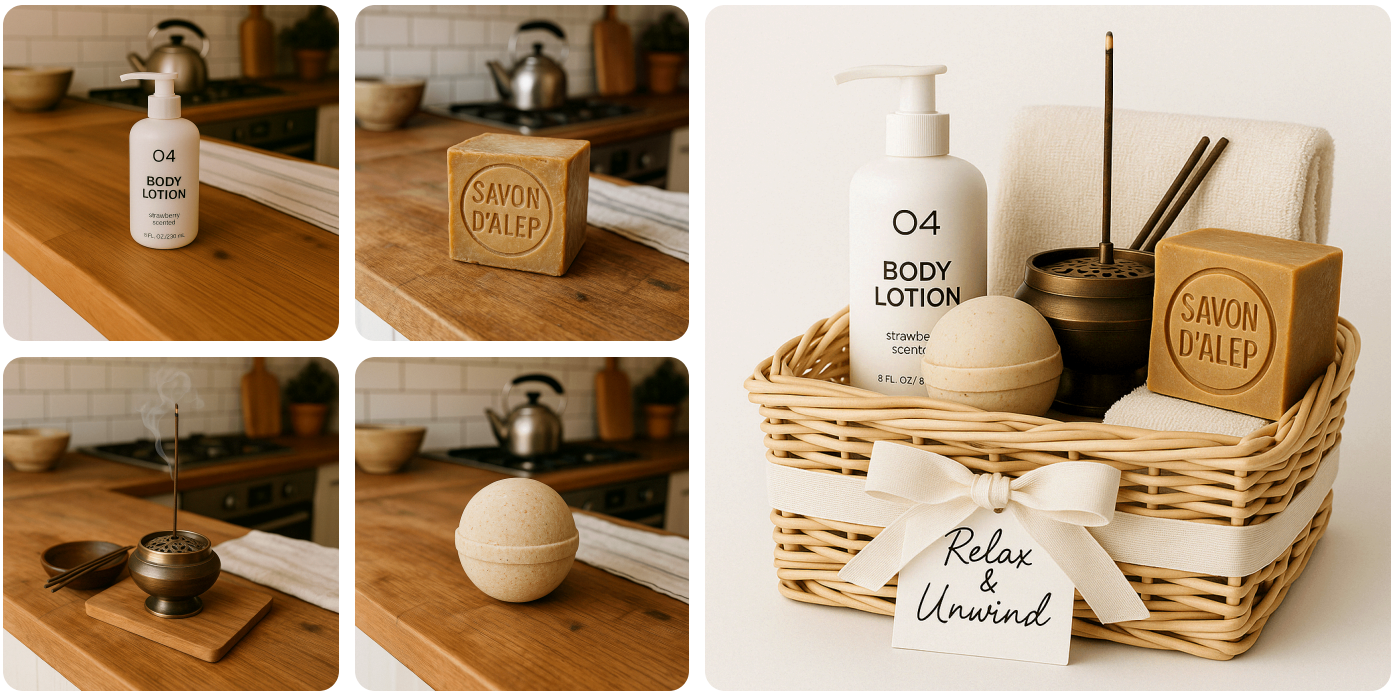
The image edits endpoint lets you:

- Edit existing images
- Generate new images using other images as a reference
- Edit parts of an image by uploading an image and mask indicating which areas should be replaced (a process known as **inpainting**)

Create a new image using image references

You can use one or more images as a reference to generate a new image.

In this example, we'll use 4 input images to generate a new image of a gift basket containing the items in the reference images.



Responses API | Image API

With the Responses API, you can provide input images in 3 different ways:

- By providing a fully qualified URL
- By providing an image as a Base64-encoded data URL
- By providing a file ID (created with the Files API)

Create a File

Create a base64 encoded image

Edit an image — python

```python
from openai import OpenAI
import base64

client = OpenAI()

prompt = """Generate a photorealistic image of a gift basket on a white backgrc
labeled 'Relax & Unwind' with a ribbon and handwriting-like font,
containing all the items in the reference pictures."""

base64_image1 = encode_image("body-lotion.png")
base64_image2 = encode_image("soap.png")
```

```python
file_id1 = create_file("body-lotion.png")
file_id2 = create_file("incense-kit.png")

response = client.responses.create(
    model="gpt-4.1",
    input=[
        {
            "role": "user",
            "content": [
                {"type": "input_text", "text": prompt},
                {
                    "type": "input_image",
                    "image_url": f"{base64_image1}",
                },
                {
                    "type": "input_image",
                    "image_url": f"{base64_image2}",
                },
                {
                    "type": "input_image",
                    "file_id": file_id1,
                },
                {
                    "type": "input_image",
                    "file_id": file_id2,
                }
            ],
        }
    ],
    tools=[{"type": "image_generation"}],
)

image_generation_calls = [
    output
    for output in response.output
    if output.type == "image_generation_call"
]

image_data = [output.result for output in image_generation_calls]

if image_data:
    image_base64 = image_data[0]
    with open("gift-basket.png", "wb") as f:
        f.write(base64.b64decode(image_base64))
else:
    print(response.output.content)
```

Edit an image using a mask (inpainting)

You can provide a mask to indicate which part of the image should be edited.

When using a mask with GPT Image, additional instructions are sent to the model to help guide the editing process accordingly.

> Unlike with DALL·E 2, masking with GPT Image is entirely prompt-based. This means the model uses the mask as guidance, but may not follow its exact shape with complete precision.

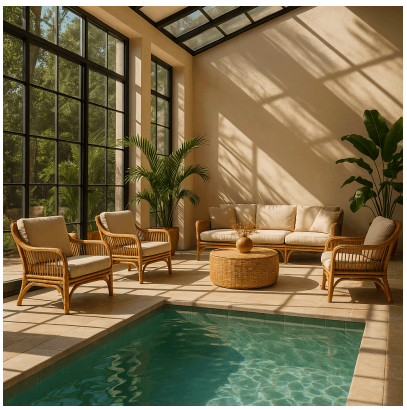
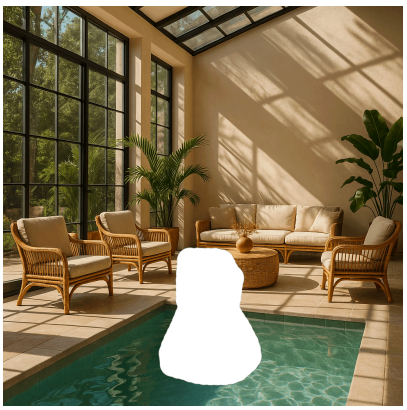
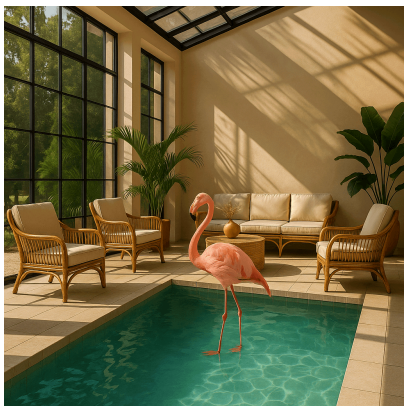
If you provide multiple input images, the mask will be applied to the first image.

Responses API | Image API

Edit an image with a mask — python

```python
from openai import OpenAI
client = OpenAI()

fileId = create_file("sunlit_lounge.png")
maskId = create_file("mask.png")

response = client.responses.create(
    model="gpt-4o",
    input=[
        {
            "role": "user",
            "content": [
                {
                    "type": "input_text",
                    "text": "generate an image of the same sunlit indoor lounge
                },
                {
                    "type": "input_image",
                    "file_id": fileId,
                }
            ],
        },
    ],
    tools=[
        {
            "type": "image_generation",
            "quality": "high",
            "input_image_mask": {
```

```
                "file_id": maskId,
            }
        },
    ],
)

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

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

IMAGE	MASK	OUTPUT
		

Prompt: a sunlit indoor lounge area with a pool containing a flamingo

Mask requirements

The image to edit and mask must be of the same format and size (less than 50MB in size).

The mask image must also contain an alpha channel. If you're using an image editing tool to create the mask, make sure to save the mask with an alpha channel.

> **Add an alpha channel to a black and white mask**

Input fidelity

GPT Image models (`gpt-image-1.5` , `gpt-image-1` , and `gpt-image-1-mini`) support high input fidelity, which allows you to better preserve details from the input images in the output. This is especially useful when using images that contain elements like faces or logos that require accurate preservation in the generated image.

You can provide multiple input images that will all be preserved with high fidelity, but keep in mind that if using `gpt-image-1` or `gpt-image-1-mini` , the first image will be preserved with richer textures and finer details, so if you include elements such as faces, consider placing them in the first image.

If you are using `gpt-image-1.5` , the first **5** input images will be preserved with higher fidelity.

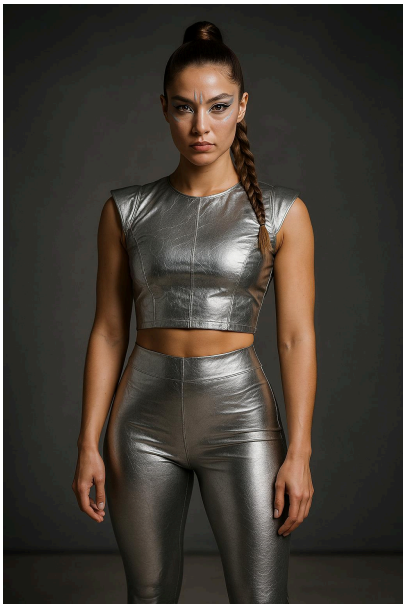
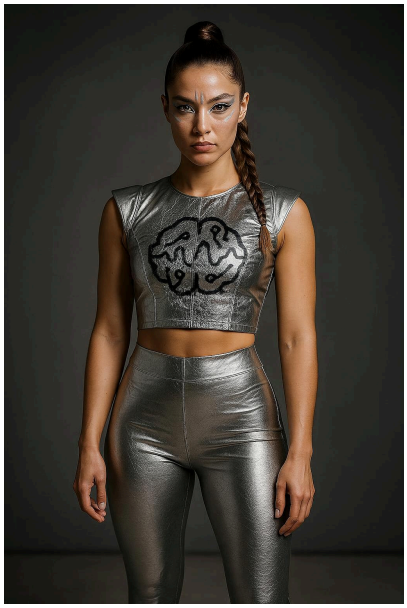
To enable high input fidelity, set the `input_fidelity` parameter to `high` . The default value is `low` .

Responses API Image API

Generate an image with high input fidelity — python

```python
from openai import OpenAI
import base64

client = OpenAI()

response = client.responses.create(
    model="gpt-4.1",
    input=[
        {
            "role": "user",
            "content": [
                {"type": "input_text", "text": "Add the logo to the woman's top
                {
                    "type": "input_image",
                    "image_url": "https://cdn.openai.com/API/docs/images/woman_
                },
                {
                    "type": "input_image",
                    "image_url": "https://cdn.openai.com/API/docs/images/brain_
                },
            ],
        }
    ],
    tools=[{"type": "image_generation", "input_fidelity": "high", "action": "ed
)

```

```python
# Extract the edited image
image_data = [
    output.result
    for output in response.output
    if output.type == "image_generation_call"
]

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

INPUT 1	INPUT 2	OUTPUT
		

Prompt: Add the logo to the woman's top, as if stamped into the fabric.

Keep in mind that when using high input fidelity, more image input tokens will be used per request. To understand the costs implications, refer to our vision costs section.

Customize Image Output

You can configure the following output options:

Size: Image dimensions (e.g., `1024x1024`, `1024x1536`)

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.

Size and quality options

Square images with standard quality are the fastest to generate. The default size is 1024x1024 pixels.

Available sizes	`1024x1024` (square) - `1536x1024` (landscape) - `1024x1536` (portrait) `auto` (default)
Quality options	- `low` - `medium` - `high` - `auto` (default)

Output format

The Image API returns base64-encoded image data. The default format is `png`, but you can also request `jpeg` or `webp`.

If using `jpeg` or `webp`, you can also specify the `output_compression` parameter to control the compression level (0-100%). For example, `output_compression=50` will compress the image by 50%.

Using `jpeg` is faster than `png`, so you should prioritize this format if latency is a concern.

Transparency

GPT Image models (`gpt-image-1.5`, `gpt-image-1`, and `gpt-image-1-mini`) support transparent backgrounds. To enable transparency, set the `background` parameter to `transparent`.

It is only supported with the `png` and `webp` output formats.

Transparency works best when setting the quality to `medium` or `high`.

Responses API | Image API

Generate an image with a transparent background

python

```python
import openai
import base64

response = openai.responses.create(
    model="gpt-5",
    input="Draw a 2D pixel art style sprite sheet of a tabby gray cat",
    tools=[
        {
            "type": "image_generation",
            "background": "transparent",
            "quality": "high",
        }
    ],
)

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

if image_data:
    image_base64 = image_data[0]

    with open("sprite.png", "wb") as f:
        f.write(base64.b64decode(image_base64))
```

Limitations

GPT Image models (`gpt-image-1.5`, `gpt-image-1`, and `gpt-image-1-mini`) are powerful and versatile image generation models, but they still have some limitations to be aware of:

Latency: Complex prompts may take up to 2 minutes to process.

Text Rendering: Although significantly improved over the DALL·E series, the model can still struggle with precise text placement and clarity.

Consistency: While capable of producing consistent imagery, the model may occasionally struggle to maintain visual consistency for recurring characters or brand elements across multiple generations.

Composition Control: Despite improved instruction following, the model may have difficulty placing elements precisely in structured or layout-sensitive compositions.

Content Moderation

All prompts and generated images are filtered in accordance with our content policy.

For image generation using GPT Image models (`gpt-image-1.5` , `gpt-image-1` , and `gpt-image-1-mini`), you can control moderation strictness with the `moderation` parameter. This parameter supports two values:

`auto` (default): Standard filtering that seeks to limit creating certain categories of potentially age-inappropriate content.

`low` : Less restrictive filtering.

Supported models

When using image generation in the Responses API, most modern models starting with `gpt-4o` and newer should support the image generation tool. Check the model detail page for your model to confirm if your desired model can use the image generation tool.

Cost and latency

This model generates images by first producing specialized image tokens. Both latency and eventual cost are proportional to the number of tokens required to render an image—larger image sizes and higher quality settings result in more tokens.

The number of tokens generated depends on image dimensions and quality:

QUALITY	SQUARE (1024×1024)	PORTRAIT (1024×1536)	LANDSCAPE (1536×1024)
Low	272 tokens	408 tokens	400 tokens
Medium	1056 tokens	1584 tokens	1568 tokens
High	4160 tokens	6240 tokens	6208 tokens

Note that you will also need to account for input tokens: text tokens for the prompt and image tokens for the input images if editing images. If you are using high input fidelity, the number of input tokens will be higher.

Refer to our pricing page for more information about price per text and image tokens.

So the final cost is the sum of:

input text tokens

input image tokens if using the edits endpoint

image output tokens

Partial images cost

If you want to stream image generation using the `partial_images` parameter, each partial image will incur an additional 100 image output tokens.