OpenAI Image API: The Lightweight Skill

Model selection, every parameter, pricing, and a prompt library

OpenAI Image API

Field	Value
Text-to-image endpoint	`POST https://api.openai.com/v1/images/generations`
Image edit endpoint	`POST https://api.openai.com/v1/images/edits`
Auth header	`Authorization: Bearer $OPENAI_API_KEY`
Docs	https://platform.openai.com/docs/guides/images

> **Maintenance note:** OpenAI changes parameter behavior without announcement. Run the edge test suite (`/home/patrick/openai-image-test/run-edge-tests.sh`) every six months to catch breaking changes. Last verified: April 2026.

Model Selection - Pick the Right Tool

Use this decision table before every call. Model choice is the biggest lever on cost, quality, and capability.

Scenario	Model	Reason
Text inside image (signs, menus, UI, infographics, posters)	`gpt-image-2`	~99% text accuracy; best available
Production-quality photorealism	`gpt-image-2`	Current quality leader
Complex multi-element layouts (comic panels, storyboards, 8-image batch)	`gpt-image-2` with thinking mode	Reasoning pre-plans layout before rendering

Scenario	Model	Reason
High-volume drafts, thumbnails, iteration	`gpt-image-2` quality=low OR `gpt-image-1-mini` quality=low	Cost: ~$0.01/image; quality sufficient for exploratory work
Transparent background required	`gpt-image-1.5`	gpt-image-2 returns a hard error on `background: "transparent"`. Use gpt-image-1.5 with `output_format: "png"` or `"webp"` - not jpeg (jpeg also errors).
Arbitrary custom dimensions needed	`gpt-image-2`	gpt-image-1.5 only accepts four fixed sizes. gpt-image-2 accepts any valid dimensions.
Tight style preservation in edits	Consider Flux 2 Pro or other specialist models	gpt-image-2 edit reinterprets rather than preserves
Legacy workflow compatibility only	`gpt-image-1.5`	Do not use for new work; gpt-image-2 is strictly better

Key facts: - `gpt-image-2` released April 21, 2026. Default for all new work. - DALL-E 2 and DALL-E 3 are deprecated and retire May 12, 2026. Do not use them. - `gpt-image-2` has two modes: **standard** (fast, $0.04-$0.21/image depending on size/quality) and **thinking** (reasoning + optional web search, variable cost). Use thinking mode only when layout complexity justifies it. - `quality: "auto"` is a valid value on gpt-image-2 in addition to `"low"`, `"medium"`, `"high"`.

Pricing (OpenAI direct, April 2026)

Token-based billing. Rough per-image cost at 1024×1024:

Model	Quality	Est. cost/image
`gpt-image-2`	low	~$0.04
`gpt-image-2`	medium	~$0.10
`gpt-image-2`	high	~$0.21
`gpt-image-1-mini`	low	~$0.005-0.006
`gpt-image-1`	low	~$0.011
`gpt-image-1`	high	~$0.167
`gpt-image-1.5`	medium	~$0.04
`gpt-image-1.5`	high	~$0.133-0.20

Thinking mode adds reasoning token cost on top - variable. Check OpenAI pricing page for exact token rates: https://openai.com/api/pricing/

Text-to-Image

Minimum request

```
curl -s -X POST "https://api.openai.com/v1/images/generations" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"model": "gpt-image-2", "prompt": "A diner chalkboard: TODAY SPECIAL  -
Lobster Roll $24"}'
```

Full request

```
{
  "model": "gpt-image-2",
  "prompt": "...",
  "size": "1536x1024",
  "quality": "medium",
  "n": 1,
  "output_format": "png"
}
```

With transparent background (gpt-image-1.5 only)

```
{
  "model": "gpt-image-1.5",
  "prompt": "...",
  "background": "transparent",
  "output_format": "png"
}
```

Image Edit

Pass one or more reference images as base64 or file upload. Prompt describes the change.

```
curl -s -X POST "https://api.openai.com/v1/images/edits" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -F "model=gpt-image-2" \
  -F "image[]=@reference.png" \
  -F "prompt=Same workers on the beam  -  everyone is on their phone now. One
taking a selfie." \
  -F "size=auto"
```

Edit with mask (surgical region edit)

Mask is a PNG where white = edit this region, black = leave it alone.

```
curl -s -X POST "https://api.openai.com/v1/images/edits" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -F "model=gpt-image-2" \
  -F "image[]=@original.png" \
  -F "mask=@mask.png" \
  -F "prompt=Replace the sky with a dramatic sunset" \
  -F "size=auto"
```

Parameters

Text-to-image

Param	Required	Options	Default	Notes
`model`	Yes	`gpt-image-2`, `gpt-image-1.5`, `gpt-image-1`, `gpt-image-1-mini`	-	
`prompt`	Yes	string, 1-32,000 chars	-	Empty string is a hard error
`size`	No	see Size section per model	`1254x1254` (img2, unverified) / `1024x1024` (img1.5)	See size constraints below
`quality`	No	`"low"`, `"medium"`, `"high"`, `"auto"`	`"medium"`	`"auto"` confirmed valid on gpt-image-2
`n`	No	1-10	`1`	11+ is a hard error on both models

Param	Required	Options	Default	Notes
`output_format`	No	`"png"`, `"jpeg"`, `"webp"`	`"png"`	
`output_compression`	No	0-100	`100`	jpeg and webp only - hard error on png
`background`	No	`"transparent"`, `"opaque"`, `"auto"`	`"auto"`	See background section below

Image edit

Same params plus:

Param	Required	Notes
`image[]`	Yes	File upload(s) - reference image(s)
`mask`	No	PNG mask - white = edit region, black = preserve
`size`	No	`"auto"` infers from input image dimensions

Size Constraints

gpt-image-2 - flexible, arbitrary dimensions

- Both dimensions must be **multiples of 16** - hard error otherwise
- Minimum total pixels: **655,360** (e.g. 1024×640) - hard error below
- Maximum total pixels: **8,294,400** (e.g. 3840×2160) - hard error above
- Maximum aspect ratio: **3:1** - hard error above (e.g. 3840×1264 fails, 3840×1280 passes)
- Max edge: **3840px**
- Default when `size` omitted: **1254×1254** (observed, undocumented - specify explicitly)

Resolution strategy: Default to 2K (1536×1024 landscape or equivalent). Outputs above 2560×1440 are experimental via the direct API - inconsistent results, higher cost. If a destination genuinely requires 4K (large-format print, billboard), generate and validate at 2K first, then upscale with Real-ESRGAN or equivalent. Upscaling a clean 2K image is fast, cheap, and consistent. Do not request experimental API resolution when upscaling is more reliable.

gpt-image-1.5 - fixed sizes only

Only four valid values - any other size string is a hard error:

Value	Dimensions
`"1024x1024"`	Square
`"1536x1024"`	Landscape
`"1024x1536"`	Portrait
`"auto"`	Model chooses - resolves to 1024×1024

Custom dimensions are **not supported** on gpt-image-1.5. This is a major difference from gpt-image-2.

Background Parameter

Confirmed behavior (tested April 2026):

Model	`"transparent"`	`"opaque"`	`"auto"`
`gpt-image-2`	**Hard error** - "Transparent background is not supported for this model"	Works	Works
`gpt-image-1.5`	Works with png/webp. **Hard error with jpeg** - "Transparent background is not supported for JPEG output format"	Works	Works

The skill's previous note that gpt-image-2 "silently produces opaque output" was incorrect. It errors.

Response

```
{
  "created": 1713715200,
  "data": [
    {
      "b64_json": "...",
      "revised_prompt": "..."
    }
  ]
}
```

Default response for GPT Image models is always `b64_json` - the `response_format: "url"` parameter is not supported (that's a DALL-E parameter). Decode and save locally immediately.

Node.js Helper

```
import fs from 'node:fs';
import path from 'node:path';
import OpenAI from 'openai';

const client = new OpenAI(); // uses OPENAI_API_KEY env var

async function generateImage({ prompt, model = 'gpt-image-2', size =
'1536x1024',
  quality = 'medium', n = 1, outputDir = './output', label = 'img' }) {
  fs.mkdirSync(outputDir, { recursive: true });

  const response = await client.images.generate({ model, prompt, size, quality,
n });

  const ts = Date.now();
  return response.data.map((img, i) => {
    const suffix = response.data.length > 1 ? `_${i + 1}` : '';
    const fp = path.join(outputDir, `${label}_${ts}${suffix}.png`);
    fs.writeFileSync(fp, Buffer.from(img.b64_json, 'base64'));
    console.log(`Saved: ${fp}`);
    return fp;
  });
}

await generateImage({ prompt: 'A chalkboard reading "OPEN AT 7"', label:
'chalkboard' });
```

Python Helper

```
import base64, time, pathlib
from openai import OpenAI

client = OpenAI()  # uses OPENAI_API_KEY env var

def generate_image(prompt, model='gpt-image-2', size='1536x1024',
                   quality='medium', n=1, out_dir='./output', label='img'):
    pathlib.Path(out_dir).mkdir(parents=True, exist_ok=True)
    response = client.images.generate(
        model=model, prompt=prompt, size=size, quality=quality, n=n
    )
    ts = int(time.time() * 1000)
    saved = []
    for i, img in enumerate(response.data):
        suffix = f'_{i+1}' if len(response.data) > 1 else ''
        fp = f'{out_dir}/{label}_{ts}{suffix}.png'
        with open(fp, 'wb') as f:
            f.write(base64.b64decode(img.b64_json))
        saved.append(fp)
        print(f'Saved: {fp}')
    return saved

generate_image('A diner chalkboard: TODAY SPECIAL  -  Lobster Roll $24',
label='chalkboard')
```

Text-Heavy Composites

When an image needs more than a few words - body copy, multiple headlines, captions, labels, data - don't fight the model's text rendering limits. Use a two-step composite workflow instead.

Why: AI image models render short text well but degrade on longer copy (wrapping, spacing, multi-line layout). HTML/CSS gives you full typographic control at zero additional AI cost.

The pattern

1. **Generate the background** - prompt for the scene, illustration, or photo without any text.
2. **Composite in HTML** - use the generated image as a CSS background. Layer text with full HTML/CSS control: font, size, leading, weight, color, shadows, positioning.
3. **Export to PNG** - render the HTML via headless Chromium (Puppeteer or Playwright). Any server running Node.js can do this.

Headless export - Puppeteer

```
import puppeteer from 'puppeteer';

async function compositeToPNG({ html, outputPath, width = 1200, height = 630 })
{
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.setViewport({ width, height, deviceScaleFactor: 2 }); // 2x =
retina-quality output
  await page.setContent(html, { waitUntil: 'networkidle0' });
  await page.screenshot({ path: outputPath, type: 'png', clip: { x: 0, y: 0,
width, height } });
  await browser.close();
}

const imageUrl = 'https://...'; // or base64 data URI from OpenAI response
const html = `
  <html><body style="margin:0;padding:0;width:1200px;height:
630px;position:relative;
    background:url('${imageUrl}') center/cover no-repeat;">
    <div style="position:absolute;bottom:60px;left:80px;right:80px;
      font-family:'Georgia',serif;color:#fff;text-shadow:0 2px 8px
rgba(0,0,0,.7);">
      <h1 style="font-size:52px;line-height:1.2;margin:0 0 16px">Headline</h1>
      <p style="font-size:24px;line-height:1.5;margin:0">Body copy here. Full
wrapping, no limits.</p>
    </div>
  </body></html>
`;

await compositeToPNG({ html, outputPath: './output/card.png' });
```

Headless export - Playwright

```
import { chromium } from 'playwright';

const browser = await chromium.launch();
const page = await browser.newPage();
await page.setViewportSize({ width: 1200, height: 630 });
await page.setContent(html, { waitUntil: 'networkidle0' });
await page.screenshot({ path: './output/card.png', type: 'png' });
await browser.close();
```

When to use this pattern

- Social cards, thumbnails, ad creative with headline + body copy

- Infographics where data labels or annotations are extensive
- Posters or flyers where layout is text-dominant
- Any output where text must be editable, templated, or localized

When NOT to use

- Short text baked into the visual (signs, chalkboards, menus) - keep it in the prompt
- Text that is stylistically part of the art (hand-lettered, painted, engraved)
- Text that must interact spatially with scene elements (speech bubbles, arrows pointing at objects)

Prompting

For text in images: - Quote the exact text: `hand-lettered text reads "OPEN AT 7"` - Specify lettering style: `chalk lettering`, `serif sign painter`, `bold sans-serif neon`, `embossed metal type` - Anchor the surface: chalkboard, brick wall, neon storefront, paper menu, vinyl sleeve - Spell out multi-line breaks: `Line 1: "ESPRESSO". Line 2: "$4.50".` - For complex layouts (posters, infographics, app mockups): structure prompts as JSON with explicit keys (`type`, `header`, `layout`, `footer`). The model responds better to explicit slots than prose.

For edits: - Describe the change, not the full scene - Call out what stays the same to anchor preservation - Use a mask for surgical edits - without it, the model may reinterpret the whole image - Chain edits by passing the previous output back in as a reference image

What gpt-image-2 handles well: - Text in images (current best available) - Photorealism, product photography, editorial - Complex multi-element layouts and infographics - Style transfer (pixel art, watercolor, anime, oil painting, film stills) - UI and app screenshot mockups

What it handles poorly: - Long paragraphs of body text - use the Text-Heavy Composites workflow above - Strict style preservation in edits - it reinterprets more than it preserves - Transparent backgrounds - use gpt-image-1.5

Prompt Library

700+ community-curated prompts across styles and use cases: https://github.com/YouMind-OpenLab/awesome-gpt-image-2

Local cache pattern

Fetch once, reference locally. Don't re-download on every use.

```
PROMPT_LIB="/opt/gpt2-prompts/awesome-gpt-image-2.md"
if [ ! -f "$PROMPT_LIB" ]; then
  mkdir -p "$(dirname $PROMPT_LIB)"
  curl -s https://raw.githubusercontent.com/YouMind-OpenLab/awesome-gpt-
image-2/main/README.md \
    -o "$PROMPT_LIB"
fi

# Search by category or keyword
grep -n "^### No\." "$PROMPT_LIB" | grep -i "poster"

# Read a specific prompt block by line range
sed -n '2722,2830p' "$PROMPT_LIB"
```

Each entry: `### No. N: Category - Title` → `#### Description` → `#### Prompt` → `#### Details`.

Use for style and category inspiration. Adapt to your use case - treat as starting points.

Gotchas

- Auth header is `Authorization: Bearer $OPENAI_API_KEY`
- `gpt-image-2` transparent background is a **hard error**, not silent - "Transparent background is not supported for this model"
- `output_compression` on PNG is a **hard error** - only valid for jpeg and webp
- Default size when `size` is omitted on gpt-image-2 is `1254x1254` (undocumented - always specify explicitly)
- gpt-image-1.5 does **not** support custom dimensions - only `1024x1024`, `1536x1024`, `1024x1536`, `auto`
- Transparent + jpeg on gpt-image-1.5 is a **hard error**
- Max `n` is 10 on both models - 11 errors
- DALL-E 2 and DALL-E 3 retire May 12, 2026 - migrate any existing code before then
- GPT Image models always return `b64_json` - `response_format: "url"` is a DALL-E parameter, not supported here
- Thinking mode costs vary with reasoning token usage - monitor if cost-sensitive