



DALL-E ❤️ SPFx

Using DALL-E OpenAI Image generator API from SPFx

About Me



Luis Mañez
Chief Architect @ ClearPeople

M365 Development MVP. More than 20 years of experience working with Microsoft Technologies. MS Certified specialist in Developing Azure and SharePoint MCPD. Passionate about technology, blogger, technical writer and MS community active member.



@luismanez



DALL-E 2

DALL-E 2 is an AI system that can create realistic images and art from a description in natural language.

Image generation

Outpainting

Inpainting

Variations

DALL·E 2 can create original, realistic images and art from a text description. It can combine concepts, attributes, and styles.

Input

An astronaut riding a horse in photorealistic style.

Output



Image generation

Outpainting

Inpainting

Variations

DALL·E 2 can expand images beyond what's in the original canvas, creating expansive new compositions.

Input



Output



Image generation

Outpainting

Inpainting

Variations

DALL·E 2 can make realistic edits to existing images from a natural language caption. It can add and remove elements while taking shadows, reflections, and textures into account.

Input

Add a flamingo beside the pool.



Output



Image generation

Outpainting

Inpainting

Variations

DALL·E 2 can take an image and create different variations of it inspired by the original.

Input



Output



Billing

Image models

Build DALL-E directly into your apps to generate and edit novel images and art. Our image models offer three tiers of resolution for flexibility.

[Learn more ↗](#)

Resolution	Price
1024×1024	\$0.020 / image
512×512	\$0.018 / image
256×256	\$0.016 / image

Create an account

Overview Documentation Examples Playground

Help Personal

ORGANIZATION

- Personal
- Settings
- Usage**
- Members
- Billing

USER

- API Keys

Usage

Below you'll find a summary of API usage for your organization. All dates and times are UTC-based, and data may be delayed up to 5 minutes.

< March >

DAILY CUMULATIVE

Daily usage (USD)

A bar chart titled "Daily usage (USD)" showing a single teal bar for March 7th reaching approximately \$0.03. The y-axis ranges from \$0.00 to \$0.04 in increments of \$0.01. The x-axis shows dates from 01 Mar to 07 Mar.

Date	Usage (USD)
07 Mar	\$0.03

Usage this month

\$0.03 / \$10.00

Daily usage breakdown (UTC)

7 March 2023 All org members

Model usage 0 requests

Fine-tune training 0 requests

DALL-E API usage 1 request

Generate an API Key

The screenshot shows the OpenAI API Keys management interface. On the left, a sidebar menu includes 'Organization' (Personal, Settings, Usage, Members, Billing) and 'User' (API Keys). The 'API Keys' option is selected and highlighted in green. The main content area is titled 'API keys' and contains a note about displaying secret keys. It lists a single API key entry:

SECRET KEY	CREATED	LAST USED
sk-	7 Mar 2023	7 Mar 2023

A button labeled '+ Create new secret key' is present. Below this, a 'Default organization' section allows selecting 'Personal' or other organizations. A note at the bottom states: 'Note: You can also specify which organization to use for each API request. See [Authentication](#) to learn more.'



Image generation

Fine-tuning

Embeddings

Speech to text

Moderation

Rate limits

Error codes

Safety best practices

Production best practices

API REFERENCE

Introduction

Authentication

Making requests

Models

Completions

Chat

Edits

Images

Create image

Create image edit

Create image variation

Embeddings

Audio

Files

Fine-tunes

Modifications

Engines

Create image Beta

POST <https://api.openai.com/v1/images/generations>

Creates an image given a prompt.

Request body

prompt string Required

A text description of the desired image(s). The maximum length is 1000 characters.

n integer Optional Defaults to 1

The number of images to generate. Must be between 1 and 10.

size string Optional Defaults to 1024x1024

The size of the generated images. Must be one of `256x256`, `512x512`, or `1024x1024`.

response_format string Optional Defaults to url

The format in which the generated images are returned. Must be one of `url` or `b64_json`.

user string Optional

A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. [Learn more](#).

Example requestcurl ▾ [Copy](#)

```
1 curl https://api.openai.com/v1/images/generations
2 -H 'Content-Type: application/json' \
3 -H 'Authorization: Bearer YOUR_API_KEY' \
4 -d '{
5   "prompt": "A cute baby sea otter",
6   "n": 2,
7   "size": "1024x1024"
8 }'
```

Parameters[Copy](#)

```
1 {
2   "prompt": "A cute baby sea otter",
3   "n": 2,
4   "size": "1024x1024"
5 }
```

Response[Copy](#)

```
1 {
2   "created": 1589478378,
3   "data": [
4     {
5       "url": "https://..."
6     },
7     {
8       "url": "https://..."
9     }
10  ]
11 }
```

A black and white photograph of a woman wearing a full-face motocross helmet and goggles, riding a dirt bike through a deep puddle of mud. She is leaning forward, gripping the handlebars. The bike is covered in mud, and the background is a blurred, muddy landscape.

Let's See it in action!

Luis Mañez @luismanez · 5 nov. 2022

#SPFx spoiler alert! ... what about a webpart that generates images using DALL-E api? ... available soon. stay tune! #officedevnpn @m365pnp

The screenshot shows a SharePoint page with a web part titled "Image description (be creative!)". The input field contains the text: "a home built with a 'Parisian chateau' theme with fountains and extensive landscaping on the lakeshore". Below it, a slider is set to "2". Under "Size", the "256x256" option is selected. Two generated images are displayed: one showing a large, ornate house with a fountain on a lake, and another showing a different angle of the same or a similar house.

DISCLAIMER

Sébastien Levert @sebastienlevert · 5 nov. 2022
En respuesta a @luismanez y @m365pnp
That first one has a weird driveway 🚗 Or they have a floating car!

Luis Mañez @luismanez · 5 nov. 2022
En respuesta a @sebastienlevert y @m365pnp
hahaha! gonna add a disclaimer message to the webpart "I'm not responsible for the generated content" XD

1

5

26

...

↑

A black and white photograph of a person in a dark wetsuit performing a headfirst dive into a body of water. The diver is positioned centrally, creating a large splash upon entry. The background consists of dark, textured water with visible ripples and reflections.

Let's Dive In!



POST Generate images base64 POST Generate images

DALL-E Image generator / Generate images

Save

Send ...

POST https://api.openai.com/v1/images/generations

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {  
2   "prompt": "a pencil and watercolor drawing of a bright city in the future with flying cars",  
3   "n": 2,  
4   "size": "256x256",  
5   "response_format": "url",  
6   "user": "luisman"  
7 }
```

Body Cookies Headers (10) Test Results

200 OK 4.33 s 1.4 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "created": 1678268028,  
3   "data": [  
4     {  
5       "url": "https://oaidalleapiprodscus.blob.core.windows.net/private/org-j1v16Gdh4Y72ygxNBW1QpUKE/  
6         user-GX9JbKJYVKg19oLyDzLFJRz/img-QxMc4Uu8a8yxtPkgxmQvY90F.png?st=2023-03-08T08%3A33%3A48Z&  
7           se=2023-03-08T10%3A33%3A48Z&sp=r&sv=2021-08-06&sr=b&rscd=inline&rscct=image/png&  
8             skoid=6aaadede-4fb3-4698-a8f6-684d7786b067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&  
9               skt=2023-03-07T21%3A43%3A41Z&ske=2023-03-08T21%3A43%3A41Z&skb=b&skv=2021-08-06&  
10              sig=T2CGqVAndHz9tiKvmzDSqfRgB1Bz261xE/0cmleHF6s%3D"  
11     },  
12     {  
13       "url": "https://oaidalleapiprodscus.blob.core.windows.net/private/org-j1v16Gdh4Y72ygxNBW1QpUKE/  
14         user-GX9JbKJYVKg19oLyDzLFJRz/img-SAmWtLGNjkPHsHg8wHDW4rVC.png?st=2023-03-08T08%3A33%3A48Z&  
15           se=2023-03-08T10%3A33%3A48Z&sp=r&sv=2021-08-06&sr=b&rscd=inline&rscct=image/png&  
16             skoid=6aaadede-4fb3-4698-a8f6-684d7786b067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&  
17               skt=2023-03-07T21%3A43%3A41Z&ske=2023-03-08T21%3A43%3A41Z&skb=b&skv=2021-08-06&  
18                 sig=T2CGqVAndHz9tiKvmzDSqfRgB1Bz261xE/0cmleHF6s%3D"  
19   ]  
20 }
```

Cookies Capture requests Runner Trash

EXPLORER

OPEN EDITORS

REACT-DALL-E-IMAGE-GENERATI... [+]

- src
- webparts/dalleImageGenerator
 - assets
 - components
 - DalleImageGenerator.module.scss
 - DalleImageGenerator.module.scss.ts
 - DalleImageGenerator.tsx
 - IDalleImageGeneratorProps.ts
 - IDalleImageGeneratorState.ts
 - loc
 - models
 - IGeneratedImageInfo.ts
 - IGeneratedImagesRequest.ts M
 - IGeneratedImagesResponse.ts
 - services
 - DalleImageGeneratorService.ts
 - Constants.ts M
 - DalleImageGeneratorWebPart.manifes...
 - DalleImageGeneratorWebPart.ts
 - pnpjsConfig.ts
 - index.ts
 - teams
 - temp
 - outline
- OUTLINE

IGeneratedImagesRequest.ts M X

src > webparts > dalleImageGenerator > models > IGeneratedImagesRequest.ts > GeneratedImagesRequest

```
10
11 export default class GeneratedImagesRequest implements IGeneratedImagesRequest {
12   readonly prompt: string;
13   readonly n: number;
14   readonly size: imageGenerationAllowedSizes;
15   readonly response_format: string;
16
17   constructor(
18     imageDescription: string,
19     numberOfImagesToGenerate: number,
20     imageSize: imageGenerationAllowedSizes) {
21
22     if (numberOfImagesToGenerate > 10)
23       throw Error("Cannot generate more than 10 images per request");
24
25     this.prompt = imageDescription;
26     this.n = numberOfImagesToGenerate;
27     this.size = imageSize;
28     this.response_format = 'b64_json';
29   }
30
31   public toJson(): string {
32     return JSON.stringify(this);
33   }
34 }
```

EXPLORER

... TS IGeneratedImagesResponse.ts ×

src > webparts > dalleImageGenerator > models > TS IGeneratedImagesResponse.ts > ...

OPEN EDITORS

× TS IGeneratedImagesResponse.ts src/web...

REACT-DALL-E-IMAGE-GENERATI... D F U S

src

webparts/dalleImageGenerator ●

- > assets
- components
 - DalleImageGenerator.module.scss
 - TS DalleImageGenerator.module.scss.ts
 - TS DalleImageGenerator.tsx
 - TS IDalleImageGeneratorProps.ts
 - TS IDalleImageGeneratorState.ts
- > loc
- models
 - TS IGeneratedImageInfo.ts
 - TS IGeneratedImagesRequest.ts M
 - TS IGeneratedImagesResponse.ts SELECTED
- services
 - TS DalleImageGeneratorService.ts
 - TS Constants.ts M
 - {...} DalleImageGeneratorWebPart.manifes...
 - TS DalleImageGeneratorWebPart.ts
 - TS pnpjsConfig.ts
 - TS index.ts
- > teams
- > temp
- > colintreis

OUTLINE

```
1 import { IGeneratedImageInfo } from "./IGeneratedImageInfo";
2
3
4 export interface IGeneratedImagesResponse {
5   created: number;
6   data: IGeneratedImageInfo[];
7 }
8
9 export interface IGeneratedImageInfo {
10   b64_json: string;
11 }
```

EXPLORER

... DalleImageGeneratorService.ts

OPEN EDITORS

X DalleImageGeneratorService.ts src/web...

REACT-DALL-E-IMAGE-GENERATION

src

webparts/dalleImageGenerator

> assets

components

DalleImageGenerator.module.scss

TS DalleImageGenerator.module.scss.ts

TS DalleImageGenerator.tsx

TS IDalleImageGeneratorProps.ts

TS IDalleImageGeneratorState.ts

> loc

models

IGeneratedImageInfo.ts

IGeneratedImagesRequest.ts M

IGeneratedImagesResponse.ts

services

DalleImageGeneratorService.ts

TS Constants.ts M

{...} DalleImageGeneratorWebPart.manifes...

TS DalleImageGeneratorWebPart.ts

TS pnpjsConfig.ts

TS index.ts

> teams

> temp

> colintreis

OUTLINE

src > webparts > dalleImageGenerator > services > DalleImageGeneratorService.ts > ...

```
1 import GeneratedImagesRequest, { IGeneratedImagesRequest, imageGenerationAllowedSizes } from '@pnp/sp';
2 import { IGeneratedImagesResponse } from '../models/IGeneratedImagesResponse';
3 import Constants from '../Constants';
4
5 import { ServiceKey, ServiceScope } from "@microsoft/sp-core-library";
6 import { HttpClient, IHttpClientOptions, HttpClientResponse } from "@microsoft/sp-http";
7
8 import { getSP } from "../pnpjsConfig";
9 import { SPFI } from "@pnp/sp";
10 import "@pnp/sp/lists";
11 import "@pnp/sp/folders/list";
12 import "@pnp/sp/files/folder";
13
14 export interface IDalleImageGeneratorService {
15     generateImages(
16         imageDescription: string,
17         numberOfImagesToGenerate: number,
18         imageSize: imageGenerationAllowedSizes): Promise<IGeneratedImagesResponse>;
19
20     saveImageToSiteAssetsLibrary(imageName: string, image: Blob): Promise<string>;
21 }
22
23 export default class DalleImageGeneratorService implements IDalleImageGeneratorService {
24
25     private _httpClient: HttpClient;
26     private _sp: SPFI;
27
28     constructor(serviceScope: ServiceScope) {
29
30     }
```



EXPLORER

OPEN EDITORS

REACT-DALL-E-IMAGE-GENERATION

- src
- webparts/dalleImageGenerator
 - assets
 - components
 - DalleImageGenerator.module.scss
 - DalleImageGenerator.module.scss.ts
 - DalleImageGenerator.tsx
 - IDalleImageGeneratorProps.ts
 - IDalleImageGeneratorState.ts
 - loc
 - models
 - IGeneratedImageInfo.ts
 - IGeneratedImagesRequest.ts
 - IGeneratedImagesResponse.ts
 - services
 - DalleImageGeneratorService.ts
 - Constants.ts
 - DalleImageGeneratorWebPart.manifest.json
 - DalleImageGeneratorWebPart.ts
 - pnpjsConfig.ts
 - index.ts
- teams
- temp
- outline

DalleImageGeneratorService.ts M X

> webparts > dalleImageGenerator > services > DalleImageGeneratorService.ts > IDalleImageGeneratorService

```
21 }  
22  
23 export default class DalleImageGeneratorService  
24     implements IDalleImageGeneratorService {  
25  
26     private _httpClient: HttpClient;  
27     private _sp: SPFI;  
28  
29     constructor(serviceScope: ServiceScope) {  
30         serviceScope.whenFinished(async () => {  
31             this._httpClient = serviceScope.consume(HttpClient.serviceKey)  
32         });  
33         this._sp = getSP();  
34     }  
35  
36     public async saveImageToSiteAssetsLibrary(imageName: string, image: Blob) {  
37         const imageFileToSharePoint =  
38             await this._sp.web.lists  
39                 .getByTitle("Documents")  
40                 .rootFolder  
41                 .files  
42                 .addChunked(imageName, image, undefined, true);  
43         const item = await imageFileToSharePoint.file  
44             .getItem<{ Id: number, Title: string, FileRef: string }>  
45                 ("Id", "Title", "FileRef");  
46         return item.FileRef;  
47     }  
48 }
```



EXPLORER

OPEN EDITORS

- × [DalleImageGeneratorService.ts](#) sr... M
- ...
- REACT-DALL-E-IMAGE-GENERATION
- src
- webparts/dalleImageGenerator
 - > assets
 - components
 - DalleImageGenerator.module.scss
 - TS DalleImageGenerator.module.scss.ts
 - TS DalleImageGenerator.tsx
 - TS IDalleImageGeneratorProps.ts
 - TS IDalleImageGeneratorState.ts
 - > loc
 - models
 - TS IGeneratedImageInfo.ts
 - TS IGeneratedImagesRequest.ts M
 - TS IGeneratedImagesResponse.ts
 - services
 - DalleImageGeneratorService.ts M
 - TS Constants.ts M
 - {...} DalleImageGeneratorWebPart.manifes...
 - TS DalleImageGeneratorWebPart.ts
 - TS pnpjsConfig.ts
 - TS index.ts
 - > teams
 - > temp
 - > colintro.js
- > OUTLINE

DalleImageGeneratorService.ts M X

> webparts > dalleImageGenerator > services > [DalleImageGeneratorService.ts](#) > [DalleImageGeneratorService](#)

```
public async generateImages(  
    imageDescription: string,  
    numberOfImagesToGenerate: number,  
    imageSize: imageGenerationAllowedSizes):  
    Promise<IGeneratedImagesResponse> {  
  
    const generateImagesRequest: IGeneratedImagesRequest =  
        new GeneratedImagesRequest(  
            imageDescription, numberOfImagesToGenerate, imageSize);  
  
    const requestHeaders: Headers = new Headers();  
    requestHeaders.append('Content-type', 'application/json');  
    requestHeaders.append('Authorization', `Bearer ${Constants.DalleApiKey}`);  
  
    const httpClientOptions: IHttpClientOptions = {  
        body: generateImagesRequest.toJson(),  
        headers: requestHeaders  
    };  
  
    const response: HttpClientResponse =  
        await this._httpClient.post(  
            "https://api.openai.com/v1/images/generations",  
            HttpClient.configurations.v1,  
            httpClientOptions);  
  
    const generateImagesResponse: IGeneratedImagesResponse =  
        await response.json();  
  
    return generateImagesResponse;  
}
```

EXPLORER

OPEN EDITORS 1 unsaved

- DalleImageGeneratorService.ts sr... M
- DalleImageGeneratorWebPart.ts src/we...

REACT-DALL-E-IMAGE-GENERATION

- src
 - webparts/dalleImageGenerator
 - assets
 - components
 - DalleImageGenerator.module.scss
 - DalleImageGenerator.module.scss.ts
 - DalleImageGenerator.tsx
 - IDalleImageGeneratorProps.ts
 - IDalleImageGeneratorState.ts
 - loc
 - models
 - IGeneratedImageInfo.ts
 - IGeneratedImagesRequest.ts M
 - IGeneratedImagesResponse.ts
 - services
 - DalleImageGeneratorService.ts M
 - Constants.ts M
 - DalleImageGeneratorWebPart.manifest.json
 - DalleImageGeneratorWebPart.ts
 - pnpjsConfig.ts
 - index.ts
 - teams
- OUTLINE

DalleImageGeneratorService.ts M DalleImageGeneratorWebPart.ts ●

webparts > dalleImageGenerator > DalleImageGeneratorWebPart.ts > DalleImageGeneratorWebPart > render

```
20
21 export default class DalleImageGeneratorWebPart extends BaseClientSideWebPart<IDalleImageGeneratorProps, IDalleImageGeneratorState> {
22
23   private _isDarkTheme: boolean = false;
24   private _environmentMessage: string = '';
25   private _dalleImageGeneratorService: IDalleImageGeneratorService;
26
27   protected async _onInit(): Promise<void> {
28     this._environmentMessage = this._getEnvironmentMessage();
29     return super
29       .onInit()
30       .then(() => {
31         getSP(this.context);
32         const webpartScope = this.context.serviceScope.startNewChild();
33         webpartScope.finish();
34         return webpartScope;
35       })
36       .then(serviceScope => {
37         this._dalleImageGeneratorService = serviceScope.consume(DalleImageGeneratorService);
38       });
39     }
40
41
42   public render(): void {
43     const element: React.ReactElement<IDalleImageGeneratorProps> = React.createElement(
44       DalleImageGenerator,
45       {
46         description: this.properties.description,
47         isDarkTheme: this._isDarkTheme,
48         environmentMessage: this._environmentMessage,
49         hasTeamsContext: !!this.context.sdks.microsoftTeams,
50         siteAbsoluteUrl: this.context.pageContext.site.absoluteUrl,
51         dalleImageGeneratorService: this._dalleImageGeneratorService
52       }
53     );
54     this.domElement.innerHTML = ReactDOM.render(element, this.domElement);
55   }
56
57   protected _onDispose(): void {
58     this._dalleImageGeneratorService?.dispose();
59   }
60
61   private _getEnvironmentMessage(): string {
62     return `Welcome to ${this.context.pageContext.webTitle}!`;
63   }
64 }
```

EXPLORER

OPEN EDITORS 1 unsaved

- DalleImageGeneratorService.ts M
- DalleImageGeneratorWebPart.ts src/we...
- DalleImageGenerator.tsx src/webparts/d...

REACT-DALL-E-IMAGE-GENERATI... [+] 🎨 ⚙️ 🗑️

- src
- webparts/dalleImageGenerator
- assets
- components
 - DalleImageGenerator.module.scss
 - DalleImageGenerator.module.scss.ts
 - DalleImageGenerator.tsx**
 - IDalleImageGeneratorProps.ts
 - IDalleImageGeneratorState.ts
- loc
- models
 - IGeneratedImageInfo.ts
 - IGeneratedImagesRequest.ts M
 - IGeneratedImagesResponse.ts
- services
 - DalleImageGeneratorService.ts M
 - Constants.ts M
 - DalleImageGeneratorWebPart.manifes...
 - DalleImageGeneratorWebPart.ts
 - pnpjsConfig.ts
 - index.ts

OUTLINE

src > webparts > dalleImageGenerator > components > **DalleImageGenerator.tsx** > ...

```

68  <Stack>
69    <TextField
70      label="Image description (be creative!)"
71      multiline
72      rows={3}
73
74  Image description (be creative!)
75  A green dog eating a big plate of pasta
76
77
78  Number of images to generate?
79  ━━━━ 2
80
81  Size
82  ● 256x256
83  ○ 512x512
84  ○ 1024x1024
85
86
87
88  Generate Images with DALL-E
89
90  ⓘ Click on the image to save it into the site documents library
91  text="Generate images with DALL-E"
92  onClick={this._sendImagesGenerationRequest} />
93
94  <Stack tokens={{ padding: 10, childrenGap: 10 }}>
95    <MessageBar>{this.state.actionMessageInfo}</MessageBar>
96    {imageSavedInfo}
97    {images}
98  </Stack>
99

```



```
... generatorService.ts M DalleImageGeneratorWebPart.ts DalleImageGenerator.tsx M ...  
src > webparts > dalleImageGenerator > components > DalleImageGenerator.tsx > DalleImageGenerator  
100 );  
101 }  
102  
103 private async _sendImagesGenerationRequest(): Promise<void> {  
104   console.log(this.state);  
105  
106   this.setState({  
  
let if import { base64StringToBlob } from 'blob-util';  
images = this.state.images.map(imageAsBase64, index) =>  
  const blob = base64StringToBlob(imageAsBase64);  
  const url = URL.createObjectURL(blob);  
  return <Image key={index} src={url} alt={this.state.imageDescription} onClick=  
    const imageUrl: string =  
      await this.props.dalleImageGeneratorService.saveImageToSiteAssetsLibrary(  
        `${this.state.imageDescription}_${index}.png`, blob);  
    this.setState({  
      imageSavedInfo: this.props.siteAbsoluteUrl + imageUrl  
    });  
  };  
});  
};
```



<https://github.com/pnp/sp-dev-fx-webparts/tree/main/samples/react-dall-e-image-generation>





Thanks!
#sharingiscaring



Microsoft®
Most Valuable
Professional