

Generate Midjourney Image

Create a new image generation task using the Midjourney AI model.

POST

/api/v1/mj/generate

Try it ▶

Generate Midjourney Image

cURL ↕

🗑

```
curl --request POST \
--url https://api.kie.ai/api/v1/mj/generate \
--header 'Authorization: Bearer <token>' \
--header 'Content-Type: application/json' \
--data '{
  "taskType": "mj_txt2img",
  "speed": "relaxed",
  "prompt": "Help me generate a sci-fi themed fighter jet in a beautiful sky, to be used as a computer wallpaper",
  "fileUrls": [
    "https://example.com/image.jpg"
  ],
  "aspectRatio": "16:9",
  "version": "7",
  "variety": 10,
  "stylization": 1,
  "weirdness": 1,
  "waterMark": "",
  "enableTranslation": false,
  "callbackUrl": "https://api.example.com/callback",
  "ow": 500,
  "videoBatchSize": 1,
  "motion": "high"
}'
```

200 500

🗑

```
{
  "code": 200,
  "msg": "success",
  "data": {
    "taskId": "mj_task_abcdef123456"
  }
}
```

Usage Guide

- This endpoint creates images based on your text prompt
- Multiple variations will be generated
- Supports text-to-image, image-to-image, and image-to-video generation

Developer Notes

- Recommendation for First-Time Users: Start with simple prompts for best results.
- Generated image files are retained for 15 days before being deleted.
- Ensure all required fields are provided for optimal generation.

Authorizations

Authorization

string

header

required

All APIs require authentication via Bearer Token.

Get API Key:

- Visit [API Key Management Page](#) to get your API Key

Usage:

Add to request header:

Authorization: Bearer YOUR_API_KEY

Note:

- Keep your API Key secure and do not share it with others
- If you suspect your API Key has been compromised, reset it immediately in the management page

Body

application/json

taskType

enum<string>

default:mj_txt2img

required

Task type for generation mode.

Image Generation Types:

- mj_txt2img: Text-to-image generation
- mj_img2img: Image-to-image generation
- mj_style_reference: Style reference
- mj_omni_reference: Omni reference

Video Generation Types:

- mj_video: Image-to-standard-definition-video generation (supports videoBatchSize parameters)
- mj_video_hd: Image-to-high-definition-video generation (supports videoBatchSize parameters)

Available options: `mj_txt2img` , `mj_img2img` , `mj_style_reference` , `mj_omni_reference` , `mj_video` , `mj_video_hd`

Example: `"mj_txt2img"`

prompt

string

required

Text prompt describing the desired image content. Required for all generation modes.

- Should be detailed and specific in describing image content
- Can include style, composition, lighting and other visual elements
- Max length: 2000 characters

Example: `"Help me generate a sci-fi themed fighter jet in a beautiful sky, to be used as a computer wallpaper"`

speed

enum<string>

The speed of the API. It can be 'fast', 'relaxed' or 'turbo', which corresponds to different speed of Midjourney.

- This parameter is not required when taskType is mj_video, mj_video_hd or mj_omni_reference

Available options: `relaxed` , `fast` , `turbo`

Example: `"relaxed"`

fileUrl

string

Input image URL (required for image-to-image and image-to-video generation).

- Use either fileUrl or fileUrls field
- Must be a valid image URL
- Image must be accessible to the API server
- Leave empty for text-to-image generation

Example: `"https://example.com/image.jpg"`

fileUrls

string[]

Input image URL array (required for image-to-image and image-to-video generation).

- Use either fileUrl or fileUrls field
- For backward compatibility, fileUrl field is currently retained
- When generating videos, fileUrls can only have one image link
- Recommended to use fileUrls field going forward
- Must be valid image URLs
- Images must be accessible to the API server
- Leave empty for text-to-image generation

Example:

`["https://example.com/image.jpg"]`

aspectRatio

enum<string>

Output image/video aspect ratio.

Supported Aspect Ratios:

Ratio	Format Type	Common Use Cases
2:1	Ultra-wide	Cinematic displays, panoramic views
16:9	Widescreen	HD video, desktop wallpapers
4:3	Standard	Traditional displays, presentations
3:2	Classic	Traditional photography, prints
1:1	Square	Social media posts, profile pictures
3:4	Portrait	Magazine layouts, portrait photos
5:6	Portrait	Mobile photography, stories
9:16	Mobile Portrait	Smartphone wallpapers, stories
2:3	Portrait	Mobile app splash screens
6:5	Landscape	Tablet wallpapers, digital art
1:2	Ultra-tall	Mobile app splash screens, banners

Available options: `1:2` , `9:16` , `2:3` , `3:4` , `5:6` , `6:5` , `4:3` , `3:2` , `1:1` , `16:9` , `2:1`

Example: `"16:9"`

version

enum<string>

Midjourney model version to use.

Midjourney routinely releases new model versions to improve efficiency, coherency, and quality. The latest model is the default, but each model excels at producing different types of images.

Available options: `7` , `6.1` , `6` , `5.2` , `5.1` , `niji6`

Example: `"7"`

variety

integer

Controls the diversity of generated images.

- Increment by 5 each time, such as (0, 5, 10, 15...)
- Higher values create more diverse results
- Lower values create more consistent results

Required range: `0 <= x <= 100`

Example: `10`

stylization

integer

Stylization level (0-1000).

- Controls the artistic style intensity
- Higher values create more stylized results
- Lower values create more realistic results
- Suggested to be a multiple of 50

Required range: `0 <= x <= 1000`

Example: `1`

weirdness

integer

Weirdness level (0-3000).

- Controls the creativity and uniqueness
- Higher values create more unusual results
- Lower values create more conventional results
- Suggested to be a multiple of 100

Required range: `0 <= x <= 3000`

Example: `1`

ow

integer

Omni intensity parameter. Controls the strength of the omni reference effect. Range: 1-1000, increments of 1 (e.g. 1, 2, 3).

- Only used when taskType is 'mj_omni_reference'
- Using an Omni Reference allows you to put characters, objects, vehicles, or non-human creatures from a reference image into your Midjourney creations
- Higher values result in stronger reference influence
- Lower values allow for more creative interpretation

Required range: `1 <= x <= 1000`

Example: `500`

waterMark

string

Watermark identifier.

- Optional parameter
- If provided, a watermark will be added to the generated content

Example: `"my_watermark"`

enableTranslation

boolean

default:false

Whether to enable automatic translation.

- Since prompt only supports English, when this parameter is true, the system will automatically translate non-English prompts to English
- If your prompt is already in English, you can set this to false
- Default: false

Example: `false`

callbackUrl

string<uri>

Callback URL to receive task completion updates.

- Optional but recommended for production use
- System will POST task completion status to this URL
- Alternatively, use the Get Midjourney Task Details endpoint to check status

📄 **Detailed Callback Mechanism:** See [Midjourney Image Generation Callbacks](#) for callback format, status codes, best practices, and troubleshooting.

Example: `"https://api.example.com/callback"`

videoBatchSize

enum<integer>

default:1

Number of videos to generate. Only effective when taskType is 'mj_video' or 'mj_video_hd'.

- 1: Generate 1 video (default)
- 2: Generate 2 videos
- 4: Generate 4 videos

Available options: `1` , `2` , `4`

Example: `1`

motion

enum<string>

default:high

Motion parameter for video generation. Controls the level of motion in generated videos.

- high: High motion level (default)
- low: Low motion level
- Required when taskType is 'mj_video' or 'mj_video_hd'**
- Only effective when taskType is 'mj_video' or 'mj_video_hd'

Available options: `high` , `low`

Example: `"high"`

Response

200 ~

application/json

Request successful

code

enum<integer>

Response status code

- 200:** Success - Request has been processed successfully
- 400:** Bad Request - Invalid request parameters
- 401:** Unauthorized - Authentication credentials are missing or invalid
- 402:** Insufficient Credits - Account does not have enough credits to perform the operation
- 404:** Not Found - The requested resource or endpoint does not exist
- 422:** Validation Error - The request parameters failed validation checks
- 429:** Rate Limited - Request limit has been exceeded for this resource
- 455:** Service Unavailable - System is currently undergoing maintenance
- 500:** Server Error - An unexpected error occurred while processing the request
- 501:** Generation Failed - Image generation task failed
- 505:** Feature Disabled - The requested feature is currently disabled

Available options: `200` , `400` , `401` , `402` , `404` , `422` , `429` , `455` , `500` , `501` , `505`

msg

string

Response message

Example: `"success"`

data

object

Hide child attributes

data.taskId

string

Task ID, can be used with Get Image Details endpoint to query task status

Example: `"mj_task_abcdef123456"`

Midjourney Image Generation Callbacks

When image generation is completed, the system will send a POST request to this URL

When you submit an image generation task to the Midjourney API, you can use the `callbackUrl` parameter to set a callback URL. The system will automatically push the results to your specified address when the task is completed.

Callback Mechanism Overview

-  The callback mechanism eliminates the need to poll the API for task status. The system will proactively push task completion results to your server.

Callback Timing

The system will send callback notifications in the following situations:

- Midjourney image generation task completed successfully
- Midjourney image generation task failed
- Errors occurred during task processing

Callback Method

- HTTP Method:** POST
- Content Type:** application/json
- Timeout Setting:** 15 seconds

Callback Request Format

When the task is completed, the system will send a POST request to your `callbackUrl` in the following format:

Success CallbackFailure Callback

```
{
  "code": 200,
  "msg": "success",
  "data": {
    "taskId": "mj_task_12345",
    "promptJson": "{\"prompt\":\"a beautiful landscape\",\"model\":\"midjourney\"}",
    "resultUrls": [
      "https://example.com/mj_result1.png",
      "https://example.com/mj_result2.png",
      "https://example.com/mj_result3.png",
      "https://example.com/mj_result4.png"
    ]
  }
}
```

Status Code Description

`code` integer required

Callback status code indicating task processing result:

Status Code	Description
200	Success - Image generation completed
500	Server Error - Image generation failed or other internal error

`msg` string required

Status message providing detailed status description

`data.taskId` string required

Task ID, consistent with the taskId returned when you submitted the task

`data.promptJson` string required

JSON string containing the original request parameters, useful for tracking generation request details

`data.resultUrls` array required

Array of result URLs for generated images/videos, contains accessible download links on success

Callback Reception Examples

Here are example codes for receiving callbacks in popular programming languages:

Node.jsPythonPHP

```
const express = require('express');
const app = express();

app.use(express.json());

app.post('/mj-image-callback', (req, res) => {
  const { code, msg, data } = req.body;

  console.log('Received Midjourney image generation callback:', {
    taskId: data.taskId,
    status: code,
    message: msg
  });

  if (code === 200) {
    // Task completed successfully
    console.log('Midjourney image generation completed');

    // Parse original request parameters
    try {
      const promptData = JSON.parse(data.promptJson);
      console.log('Original prompt:', promptData.prompt);
    } catch (e) {
      console.log('Failed to parse promptJson:', e);
    }

    // Process generated images
    const resultUrls = data.resultUrls || [];
    console.log(`Generated ${resultUrls.length} images:`);

    resultUrls.forEach((url, index) => {
      console.log(`Image ${index + 1}: ${url}`);
    });

    // Download and save images
    // Add image download logic here


  } else {
    // Task failed
    console.log('Midjourney image generation failed:', msg);


    // Handle failure cases...
  }

  // Return 200 status code to confirm callback received
  res.status(200).json({ status: 'received' });
});

app.listen(3000, () => {
  console.log('Callback server running on port 3000');
});
```

Best Practices

-  **Callback URL Configuration Recommendations**
- Use HTTPS:** Ensure your callback URL uses HTTPS protocol for secure data transmission
 - Verify Source:** Verify the legitimacy of the request source in callback processing
 - Idempotent Processing:** The same taskId may receive multiple callbacks, ensure processing logic is idempotent
 - Quick Response:** Callback processing should return a 200 status code as quickly as possible to avoid timeout
 - Asynchronous Processing:** Complex business logic should be processed asynchronously to avoid blocking callback response
 - Batch Processing:** Midjourney typically generates multiple images, recommend batch downloading and processing

-  **Important Reminders**
- Callback URL must be a publicly accessible address
 - Server must respond within 15 seconds, otherwise it will be considered a timeout
 - If 3 consecutive retries fail, the system will stop sending callbacks
 - Please ensure the stability of callback processing logic to avoid callback failures due to exceptions
 - Midjourney generated image URLs may have time limits, recommend downloading and saving promptly
 - Pay attention to processing the promptJson field, which contains useful original request information

Troubleshooting

If you do not receive callback notifications, please check the following:

Network Connection Issues

- Confirm that the callback URL is accessible from the public network
- Check firewall settings to ensure inbound requests are not blocked
- Verify that domain name resolution is correct

Server Response Issues

- Ensure the server returns HTTP 200 status code within 15 seconds
- Check server logs for error messages
- Verify that the interface path and HTTP method are correct

Content Format Issues

- Confirm that the received POST request body is in JSON format
- Check that Content-Type is application/json
- Verify that JSON parsing is correct

Image Processing Issues

- Confirm that image URLs are accessible
- Check image download permissions and network connections
- Verify image save paths and permissions
- Note that Midjourney may generate multiple images requiring batch processing

Alternative Solution

If you cannot use the callback mechanism, you can also use polling:



Poll Query Results

Use the get Midjourney task details endpoint to regularly query task status. We recommend querying every 30 seconds.

Get Midjourney Task Details

Retrieve the status and details of an Midjourney generation task.

GET /api/v1/mj/record-info

Try it

Get Midjourney Task Details

cURL

```
curl --request GET \
--url https://api.kie.ai/api/v1/mj/record-info \
--header 'Authorization: Bearer <token>'
```

200 500

```
{
  "code": 200,
  "msg": "success",
  "data": {
    "taskId": "4edb3c5XXXXX75e3f0aa5cc",
    "taskType": "mj_txt2img",
    "paramJson": "{\"aspectRatio\":\"16:9\",\"callbackUrl\":\"https://api.example.com/callback\",
\\fileUrl\":\"\\\",\\\"prompt\\\":\\\"Help me generate a sci-fi themed fighter jet in a beautiful sky, to be used as
a computer wallpaper\\\",\\\"speed\\\":\\\"Relax\\\",\\\"stylization\\\":1,\\\"taskType\\\":\\\"mj_txt2img\\\",\\\"version\\\":\\\"7\\\",
\\\"waterMark\\\":\\\"\\\",\\\"weirdness\\\":1}\",
    \"completeTime\": \"2024-03-20T10:30:00Z\",
    \"resultInfoJson\": {
      \"resultUrls\": [
        {
          \"resultUrl\": \"https://tempfile.aiquickdraw.com/v/42ea4a4250571eb18361bd2e309c8e8a_0_0.jpeg\"
        },
        {
          \"resultUrl\": \"https://tempfile.aiquickdraw.com/v/42ea4a4250571eb18361bd2e309c8e8a_0_1.jpeg\"
        },
        {
          \"resultUrl\": \"https://tempfile.aiquickdraw.com/v/42ea4a4250571eb18361bd2e309c8e8a_0_2.jpeg\"
        },
        {
          \"resultUrl\": \"https://tempfile.aiquickdraw.com/v/42ea4a4250571eb18361bd2e309c8e8a_0_3.jpeg\"
        }
      ]
    },
    \"successFlag\": 1,
    \"createTime\": \"2024-03-20T10:30:00Z\",
    \"errorCode\": null,
    \"errorMessage\": null
  }
}
```

Status Descriptions

- successFlag: 0 (Generating) - Task is currently being processed
- successFlag: 1 (Success) - Task completed successfully
- successFlag: 2 (Failed) - Task generation failed
- successFlag: 3 (Generation Failed) - Task created successfully but generation failed

Authorizations

Authorization string header required

All APIs require authentication via Bearer Token.

Get API Key:

- Visit [API Key Management Page](#) to get your API Key

Usage:

Add to request header:

Authorization: Bearer YOUR_API_KEY

Note:

- Keep your API Key secure and do not share it with others
- If you suspect your API Key has been compromised, reset it immediately in the management page

Query Parameters

taskId string required

Task ID returned from the generation request

Response200 ~ application/json

Request successful

code enum<integer>

Response status code

- 200**: Success - Request has been processed successfully
- 400**: Bad Request - Invalid request parameters
- 401**: Unauthorized - Authentication credentials are missing or invalid
- 402**: Insufficient Credits - Account does not have enough credits to perform the operation
- 404**: Not Found - The requested resource or endpoint does not exist
- 422**: Validation Error - The request parameters failed validation checks
- 429**: Rate Limited - Request limit has been exceeded for this resource
- 455**: Service Unavailable - System is currently undergoing maintenance
- 500**: Server Error - An unexpected error occurred while processing the request
- 501**: Generation Failed - Image generation task failed
- 505**: Feature Disabled - The requested feature is currently disabled

Available options: 200 , 400 , 401 , 402 , 404 , 422 , 429 , 455 , 500 , 501 , 505

msg string

Response message

Example: "success"

data object

Hide child attributes

data.taskId string

Task ID

Example: "4edb3cXXXXX5e3f0aa5cc"

data.taskType string

Task type

Example: "mj_txt2img"

data.paramJson string

Request parameters in JSON format

Example: "{\"prompt\":\"Help me generate a sci-fi themed fighter jet in a beautiful sky, to be used as a computer wallpaper\", \"fileUrl\":\"\\\", \"taskType\":\"mj_txt2img\", \"aspectRatio\":\"16:9\", \"callbackUrl\":\"https://api.example.com/callback\", \"waterMark\":\"\\\", \"stylization\":1, \"weirdness\":1, \"version\":\"7\", \"speed\":\"relaxed\"}"

data.completeTime string

Task completion time

Example: "2024-03-20T10:30:00Z"

data.resultInfoJson object

Result information

Hide child attributes

data.resultInfoJson.resultUrls object[]

Hide child attributes

resultUrl string

Result URL

data.successFlag enum<integer>

Generation status flag

- 0**: Generating
- 1**: Success
- 2**: Failed
- 3**: Generation Failed

Available options: 0 , 1 , 2 , 3

Example: 1

data.createTime string

Task creation time

Example: "2024-03-20T10:30:00Z"

data.errorCode integer | null

Error code when task fails

Exaiple: null

data.errorMessage string | null

Error message when task fails

Example: null

Extend Midjourney Video

Extend existing Midjourney-generated videos to create longer sequences.

POST /api/v1/mj/generateVideoExtend

Try it ▶

Extend Midjourney Video

cURL cURL ↕

```
curl --request POST \
--url https://api.kie.ai/api/v1/mj/generateVideoExtend \
--header 'Authorization: Bearer <token>' \
--header 'Content-Type: application/json' \
--data '{
  "prompt": "Continue the scene with the spacecraft accelerating into a colorful nebula with dynamic light trails",
  "taskType": "mj_video_extend_manual",
  "taskId": "ee603959-debb-48d1-98c4-a6d1c717eba6",
  "index": 0,
  "waterMark": "my_watermark",
  "callbackUrl": "https://api.example.com/callback"
}'
```

200 500

```
{
  "code": 200,
  "msg": "success",
  "data": {
    "taskId": "40d90dd1c6fddsa0a7dssa2a08366149"
  }
}
```

Usage Guide

- Provide the original video taskId and index to continue from the generated result
- Choose between manual extension (with prompt) or automatic extension
- Use prompt to guide how the video should continue while keeping consistency
- Optional callback is supported

Parameter Details

- **prompt** required for manual extension, describes what should happen next
- **taskType** extension mode: `mj_video_extend_manual` or `mj_video_extend_auto`
- **taskId** identifies the original MJ video record to extend
- **index** video index from the original record to extend
- **callbackUrl** callback URL for receiving completion updates
- **waterMark** video watermark, optional

Developer Notes

- Extended videos are stored for 15 days before automatic deletion
- Extension maintains the same aspect ratio and style as the original video
- Manual extension requires a prompt, automatic extension uses AI-generated continuation

Authorizations

Authorization string header **required**

All APIs require authentication via Bearer Token.

Get API Key:

1. Visit [API Key Management Page](#) to get your API Key

Usage:

Add to request header:

Authorization: Bearer YOUR_API_KEY

Note:

- Keep your API Key secure and do not share it with others
- If you suspect your API Key has been compromised, reset it immediately in the management page

Body

application/json

taskType enum<string> **required**

Extension type for video generation mode.

- `mj_video_extend_manual`: Manual extension with custom prompt
- `mj_video_extend_auto`: Automatic extension using AI-generated continuation

Available options: `mj_video_extend_manual` , `mj_video_extend_auto`

Example: `"mj_video_extend_manual"`

taskId string **required**

Task ID of the original MJ video record to extend

Example: `"ee603959-debb-48d1-98c4-a6d1c717eba6"`

index integer **required**

Video index from the original record to extend

Example: `0`

prompt string

Continuation prompt describing what should happen next in the video. Required for manual extension. Max length: 2000 characters

Example: `"Continue the scene with the spacecraft accelerating into a colorful nebula with dynamic light trails"`

waterMark string

Video watermark, optional

Example: `"my_watermark"`

callbackUrl string<uri>

Callback URL to receive extension completion updates. Optional but recommended in production.

- System will POST task status and results to this URL upon completion
- Alternatively, use the Get Midjourney Task Details endpoint to poll status

 Detailed callback format: See [Midjourney Video Extension Callbacks](#)

Example: `"https://api.example.com/callback"`

Response

200 ~ application/json

Request successful

code enum<integer>

Response status code

- **200**: Success - Request has been processed successfully
- **400**: Bad Request - Invalid request parameters
- **401**: Unauthorized - Authentication credentials are missing or invalid
- **402**: Insufficient Credits - Account does not have enough credits to perform the operation
- **404**: Not Found - The requested resource or endpoint does not exist
- **422**: Validation Error - The request parameters failed validation checks
- **429**: Rate Limited - Request limit has been exceeded for this resource
- **455**: Service Unavailable - System is currently undergoing maintenance
- **500**: Server Error - An unexpected error occurred while processing the request
- **501**: Generation Failed - Video extension task failed
- **505**: Feature Disabled - The requested feature is currently disabled

Available options: `200` , `400` , `401` , `402` , `404` , `422` , `429` , `455` , `500` , `501` , `505`

msg string

Response message

Example: `"success"`

data object

▼ Hide child attributes

`data.taskId` string

Task ID, can be used with Get Midjourney Task Details endpoint to query task status

Example: `"40d90dd1c6fddsa0a7dssa2a08366149"`

Upscale

Create an upscale task based on previously generated Midjourney images.

POST

/api/v1/mj/generateUpscale

Try it ▶

Upscale

```
curl --request POST \
  --url https://api.kie.ai/api/v1/mj/generateUpscale \
  --header 'Authorization: Bearer <token>' \
  --header 'Content-Type: application/json' \
  --data '{
    "taskId": "2584469c38e43c173dcae9e60663f645",
    "imageIndex": 0,
    "waterMark": "watermark",
    "callBackUrl": "https://bd86f681ddfb.ngrok-free.app/api/v1/mj/test"
  }'
```

200 500

```
{
  "code": 200,
  "msg": "success",
  "data": {
    "taskId": "mj_upscale_abcdef123456"
  }
}
```

Usage Guide

- This endpoint is used to upscale Midjourney generated images
- Requires providing the taskId from generation task and the image index to upscale
- Supports adding watermarks and callback notifications

Developer Notes

- imageIndex range is 0-3, corresponding to the 4 generated images
- Upscaled images have higher quality, suitable for printing or high-resolution use
- Recommended to use callback URL in production environments for result notifications

Authorizations

Authorization string header required

All APIs require authentication via Bearer Token.

Get API Key:

- Visit [API Key Management Page](#) to get your API Key

Usage:

Add to request header:

Authorization: Bearer YOUR_API_KEY

Note:

- Keep your API Key secure and do not share it with others
- If you suspect your API Key has been compromised, reset it immediately in the management page

Body

application/json

taskId string required

Task ID returned from MJ generation task

Example: "2584469c38e43c173dcae9e60663f645"

imageIndex integer required

Image index, range (0, 1, 2, 3) for the 4 generated images

Required range: 0 <= x <= 3

Example: 0

waterMark string

Watermark identifier

Example: "watermark"

callBackUrl string<uri>

Callback URL to receive task completion updates

Detailed Callback Mechanism: See [Midjourney Image Generation Callbacks](#) for callback format, status codes, best practices, and troubleshooting.

Example: "https://bd86f681ddfb.ngrok-free.app/api/v1/mj/test"

Response

200 application/json

Request successful

code enum<integer>

Response status code

- 200:** Success - Request has been processed successfully
- 400:** Bad Request - Invalid request parameters
- 401:** Unauthorized - Authentication credentials are missing or invalid
- 402:** Insufficient Credits - Account does not have enough credits to perform the operation
- 404:** Not Found - The requested resource or endpoint does not exist
- 422:** Validation Error - The request parameters failed validation checks
- 429:** Rate Limited - Request limit has been exceeded for this resource
- 455:** Service Unavailable - System is currently undergoing maintenance
- 500:** Server Error - An unexpected error occurred while processing the request
- 501:** Generation Failed - Image generation task failed
- 505:** Feature Disabled - The requested feature is currently disabled

Available options: 200 , 400 , 401 , 402 , 404 , 422 , 429 , 455 , 500 , 501 , 505

msg string

Response message

Example: "success"

data object

▼ Hide child attributes

data.taskId string

Task ID, can be used with Get Image Details endpoint to query task status

Example: "mj_upscale_abcdef123456"

Vary

Create a vary task to enhance image clarity and simulate styles based on previously generated Midjourney images.

POST

/api/v1/mj/generateVary

Try it ▶

Vary

cURL

↕

📄

```
curl --request POST \  
  --url https://api.kie.ai/api/v1/mj/generateVary \  
  --header 'Authorization: Bearer <token>' \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "taskId": "96a5***67tr",  
    "imageIndex": 1,  
    "waterMark": "my_watermark",  
    "callBackUrl": "https://example.com/callback"  
  }'
```

200 500

📄

```
{  
  "code": 200,  
  "msg": "success",  
  "data": {  
    "taskId": "2182668588ae82da0bc553c07c48ca38"  
  }  
}
```

Usage Guide

- This endpoint is used to create variations of Midjourney generated images
- Requires providing the taskId from generation task and the image index to vary
- Supports adding watermarks and callback notifications
- Enhances image clarity and provides style simulation

Developer Notes

- imageIndex range is 1-4, corresponding to the 4 generated images
- Varied images have enhanced clarity and style simulation
- Recommended to use callback URL in production environments for result notifications

Authorizations

Authorization string header required

All APIs require authentication via Bearer Token.

Get API Key:

1. Visit [API Key Management Page](#) to get your API Key

Usage:

Add to request header:

Authorization: Bearer YOUR_API_KEY

Note:

- Keep your API Key secure and do not share it with others
- If you suspect your API Key has been compromised, reset it immediately in the management page

Body

application/json

taskId string required

Task ID returned from MJ generation task

Example: "96a5***67tr"

imageIndex integer required

Image index, range (1, 2, 3, 4) for the 4 generated images

Required range: 1 <= x <= 4

Example: 1

waterMark string

Watermark identifier (optional)

Example: "my_watermark"

callBackUrl string<uri>

Callback URL to receive task completion updates (optional)

Detailed Callback Mechanism: See [Midjourney Image Generation Callbacks](#) for callback format, status codes, best practices, and troubleshooting.

Example: "https://example.com/callback"

Response

200 application/json

Request successful

code enum<integer>

Response status code

- **200:** Success - Request has been processed successfully
- **400:** Bad Request - Invalid request parameters
- **401:** Unauthorized - Authentication credentials are missing or invalid
- **402:** Insufficient Credits - Account does not have enough credits to perform the operation
- **404:** Not Found - The requested resource or endpoint does not exist
- **422:** Validation Error - The request parameters failed validation checks
- **429:** Rate Limited - Request limit has been exceeded for this resource
- **455:** Service Unavailable - System is currently undergoing maintenance
- **500:** Server Error - An unexpected error occurred while processing the request
- **501:** Generation Failed - Image generation task failed
- **505:** Feature Disabled - The requested feature is currently disabled

Available options: 200 , 400 , 401 , 402 , 404 , 422 , 429 , 455 , 500 , 501 , 505

msg string

Response message

Example: "success"

data object

Hide child attributes

data.taskId string

Task ID, can be used with Get Image Details endpoint to query task status

Example: "2182668588ae82da0bc553c07c48ca38"