Sending and returning files with GPT Actions

Copy page

Sending files

POST requests can include up to ten files (including DALL-E generated images) from the conversation. They will be sent as URLs which are valid for five minutes.

For files to be part of your POST request, the parameter must be named `openaiFileIdRefs` and the description should explain to the model the type and quantity of the files which your API is expecting.

The `openaiFileIdRefs` parameter will be populated with an array of JSON objects. Each object contains:

`name`	The name of the file. This will be an auto generated name when created by DALL-E.
`id`	A stable identifier for the file.
`mime_type`	The mime type of the file. For user uploaded files this is based on file extension.
`download_link`	The URL to fetch the file which is valid for five minutes.

Here's an example of an `openaiFileIdRefs` array with two elements:

```
[
  {
    "name": "dalle-Lh2tg7WuosbyR9hk",
    "id": "file-XFlOqJYTPBPwMZE3IopCBv1Z",
    "mime_type": "image/webp",
    "download_link": "https://files.oaiusercontent.com/file-XFlOqJYTPBPwMZE3IopCBv1Z?se=2024
  },
  {
    "name": "2023 Benefits Booklet.pdf",
    "id": "file-s5nX7o4junn2ig0J84r8Q0Ew",
    "mime_type": "application/pdf",
    "download_link": "https://files.oaiusercontent.com/file-s5nX7o4junn2ig0J84r8Q0Ew?se=2024
  }
]
```

Actions can include files uploaded by the user, images generated by DALL-E, and files created by Code Interpreter.

OpenAPI Example

```
/createWidget:
    post:
      operationId: createWidget
      summary: Creates a widget based on an image.
      description: Uploads a file reference using its file id. This file should be an image
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                openaiFileIdRefs:
                  type: array
                  items:
                    type: string
```

While this schema shows `openaiFileIdRefs` as being an array of type `string`, at runtime this will be populated with an array of JSON objects as previously shown.

Returning files

Requests may return up to 10 files. Each file may be up to 10 MB and cannot be an image or video.

These files will become part of the conversation similarly to if a user uploaded them, meaning they may be made available to code interpreter, file search, and sent as part of subsequent action invocations. In the web app users will see that the files have been returned and can download them.

To return files, the body of the response must contain an `openaiFileResponse` parameter. This parameter must always be an array and must be populated in one of two ways.

Inline option

Each element of the array is a JSON object which contains:

- `name` The name of the file. This will be visible to the user.
- `mime_type` The MIME type of the file. This is used to determine eligibility and which features have access to the file.
- `content` The base64 encoded contents of the file.

Here's an example of an openaiFileResponse array with two elements:

```
[
  {
    "name": "example_document.pdf",
    "mime_type": "application/pdf",
    "content": "JVBERi0xLjQKJcfsj6IKNSAwIG9iago8PC9MZW5ndGggNiAwIFIvRmlsdGVyIC9GbGF0ZURlY29k
  },
  {
    "name": "sample_spreadsheet.csv",
    "mime_type": "text/csv",
    "content": "iVBORw0KGgoAAAANSUhEUgAAAAUAAAAFCAYAAACNbyblAAAAHElEQVQI12P4//8/w38GIAXDIBKE
  }
]
```

OpenAPI example

```
/papers:
  get:
    operationId: findPapers
    summary: Retrieve PDFs of relevant academic papers.
    description: Provided an academic topic, up to five relevant papers will be returned as
    parameters:
      - in: query
        name: topic
        required: true
        schema:
          type: string
        description: The topic the papers should be about.
    responses:
      '200':
        description: Zero to five academic paper PDFs
        content:
            application/json:
              schema:
                type: object
                properties:
                  openaiFileResponse:
                    type: array
                    items:
                      type: object
                      properties:
                        name:
                          type: string
                          description: The name of the file.
                        mime_type:
                          type: string
                          description: The MIME type of the file.
                        content:
                          type: string
                          format: byte
                          description: The content of the file in base64 encoding.
```

URL option

Each element of the array is a URL referencing a file to be downloaded. The headers `Content-Disposition` and `Content-Type` must be set such that a file name and MIME type can be determined. The name of the file will be visible to the user. The MIME type of the file determines eligibility and which features have access to the file.

There is a 10 second timeout for fetching each file.

Here's an example of an `openaiFileResponse` array with two elements:

```
[
  "https://example.com/f/dca89f18-16d4-4a65-8ea2-ededced01646",
  "https://example.com/f/01fad6b0-635b-4803-a583-0f678b2e6153"
]
```

Here's an example of the required headers for each URL:

```
Content-Type: application/pdf
Content-Disposition: attachment; filename="example_document.pdf"
```

OpenAPI example

```
get:
    operationId: findPapers
    summary: Retrieve PDFs of relevant academic papers.
    description: Provided an academic topic, up to five relevant papers will be returned as
    parameters:
      - in: query
        name: topic
        required: true
        schema:
          type: string
        description: The topic the papers should be about.
    responses:
      '200':
        description: Zero to five academic paper PDFs
        content:
            application/json:
              schema:
                type: object
                properties:
                  openaiFileResponse:
                    type: array
                    items:
                    type: string
                    format: uri
                    description: URLs to fetch the files.
```