Using files

The Gemini API supports uploading media files separately from the prompt input, allowing your media to be reused across multiple requests and multiple prompts. For more details, check out the Prompting with media (https://ai.google.dev/gemini-api/docs/prompting_with_media) guide.

Method: media.upload

Creates a `File`.

Endpoint

- Upload URI, for media upload requests:

```
POST https://generativelanguage.googleapis.com/upload/v1beta/files
```

- Metadata URI, for metadata-only requests:

```
POST https://generativelanguage.googleapis.com/v1beta/files
```

The URL uses gRPC Transcoding (https://google.aip.dev/127) syntax.

Request body

Example request

ImageAudio (#audio)Text (#text)Video (#video)PDF (#pdf) (#image)

Python (#python)Node.jsGo (#go)Shell (#shell) (#node.js)

```
// Make sure to include these imports:
// import { GoogleAIFileManager } from "@google/ge
// import { GoogleGenerativeAI } from "@google/gen
const fileManager = new GoogleAIFileManager(proces

const uploadResult = await fileManager.uploadFile(
  `${mediaPath}/jetpack.jpg`,
  {
```

The request body contains data with the following structure:

Fields

`file` `object` (`File` (/api/files#File))

Optional. Metadata for the file to create.

```
    mimeType: "image/jpeg",
    displayName: "Jetpack drawing",
  },
);
// View the response.
console.log(
  `Uploaded file ${uploadResult.file.displayName} a
);

const genAI = new GoogleGenerativeAI(process.env.AI
const model = genAI.getGenerativeModel({ model: "g
const result = await model.generateContent([
  "Tell me about this image.",
  {
    fileData: {
      fileUri: uploadResult.file.uri,
      mimeType: uploadResult.file.mimeType,
    },
  },
]);
```

Response body

Response for **`media.upload`**.

If successful, the response body contains data with the following structure:

Fields

JSON representation

```
{
  "file": {
    object (File (/api/files#File))
  }
}
```

file object (File (/api/files#File))

Metadata for the created file.

Method: files.get

Gets the metadata for the given **File**.

Endpoint

```
GET https://generativelanguage.googleapis.com/v1beta/{name=files/*}
```

The URL uses gRPC Transcoding (https://google.aip.dev/127) syntax.

Path parameters

name string

Required. The name of the **File** to get. Example: **files/abc-123** It takes the form **files/{file}**.

Request body

The request body must be empty.

Example request

Python (#python) Node.js (#node.js) Go (#go) Shell (#shell)

```
// Make sure to include these imports:
// import { GoogleAIFileManager } from "@google/gene
const fileManager = new GoogleAIFileManager(process.

const uploadResponse = await fileManager.uploadFile(
  `${mediaPath}/jetpack.jpg`,
  {
    mimeType: "image/jpeg",
    displayName: "Jetpack drawing",
  },
);

// Get the previously uploaded file's metadata.
const getResponse = await fileManager.getFile(uploac
```

```
// View the response.
console.log(
  `Retrieved file ${getResponse.displayName} as ${ge
);
```

:da23b07af1a7135a8b461ae64e/samples/files.js#L248-L266)

Response body

If successful, the response body contains an instance of **`File`** (/api/files#File).

Method: files.list

Lists the metadata for **`File`**s owned by the requesting project.

Endpoint

```
GET https://generativelanguage.googleapis.com/v1beta/files
```

The URL uses gRPC Transcoding (https://google.aip.dev/127) syntax.

Query parameters

Example request

Python (#python) Node.js (#node.js) Go (#go) Shell (#shell)

```
// Make sure to include these imports:
// import { GoogleAIFileManager } from "@google/gene
const fileManager = new GoogleAIFileManager(process.

const listFilesResponse = await fileManager.listFile
```

`pageSize` `integer`

Optional. Maximum number of `File`s to return per page. If unspecified, defaults to 10. Maximum `pageSize` is 100.

`pageToken` `string`

Optional. A page token from a previous `files.list` call.

Request body

The request body must be empty.

Response body

Response for `files.list`.

If successful, the response body contains data with the following structure:

Fields

`files[]` `object` (`File` (/api/files#File))

The list of `File`s.

`nextPageToken` `string`

A token that can be sent as a `pageToken` into a subsequent `files.list` call.

```
// View the response.
for (const file of listFilesResponse.files) {
  console.log(`name: ${file.name} | display name: ${
}
```

:da23b07af1a7135a8b461ae64e/samples/files.js#L233-L242)

JSON representation

```
{
  "files": [
    {
      object (File (/api/files#File))
    }
  ],
  "nextPageToken": string
}
```

Method: files.delete

Deletes the `File`.

Endpoint

```
DELETE https://generativelanguage.googleapis.com/v1beta/{name=files/*}
```

The URL uses gRPC Transcoding (https://google.aip.dev/127) syntax.

Path parameters

`name` `string`

Required. The name of the `File` to delete. Example: `files/abc-123` It takes the form `files/{file}`.

Request body

The request body must be empty.

Example request

Python (#python) Node.js (#node.js) Go (#go) Shell (#shell)

```
// Make sure to include these imports:
// import { GoogleAIFileManager } from "@google/gene
const fileManager = new GoogleAIFileManager(process.

const uploadResult = await fileManager.uploadFile(
  `${mediaPath}/jetpack.jpg`,
  {
    mimeType: "image/jpeg",
    displayName: "Jetpack drawing",
  },
);

// Delete the file.
await fileManager.deleteFile(uploadResult.file.name)

console.log(`Deleted ${uploadResult.file.displayName
```

:da23b07af1a7135a8b461ae64e/samples/files.js#L272-L287)

Response body

If successful, the response body is empty.

REST Resource: files

Resource: File

A file uploaded to the API. Next ID: 15

Fields

`name` `string`

Immutable. Identifier. The `File` resource name. The ID (name excluding the "files/" prefix) can contain up to 40 characters that are lowercase alphanumeric or dashes (-). The ID cannot start or end with a dash. If the name is empty on create, a unique name will be generated. Example: `files/123-456`

`displayName` `string`

Optional. The human-readable display name for the `File`. The display name must be no more than 512 characters in length, including spaces. Example: "Welcome Image"

JSON representation

```
{
  "name": string,
  "displayName": string,
  "mimeType": string,
  "sizeBytes": string,
  "createTime": string,
  "updateTime": string,
  "expirationTime": string,
  "sha256Hash": string,
  "uri": string,
  "state": enum (State (/api/files#State)),
  "error": {
    object (Status (/api/files#v1beta.Status))
  },
```

`mimeType` `string`

Output only. MIME type of the file.

`sizeBytes`

`string (int64` (https://developers.google.com/discovery/v1/type-format) `format)`

Output only. Size of the file in bytes.

`createTime`

`string (Timestamp` (https://protobuf.dev/reference/protobuf/google.protobuf/#timestamp) `format)`

Output only. The timestamp of when the `File` was created.
A timestamp in RFC3339 UTC "Zulu" format, with nanosecond resolution and up to nine fractional digits. Examples: `"2014-10-02T15:01:23Z"` and `"2014-10-02T15:01:23.045123456Z"`.

`updateTime`

`string (Timestamp` (https://protobuf.dev/reference/protobuf/google.protobuf/#timestamp) `format)`

Output only. The timestamp of when the `File` was last updated.
A timestamp in RFC3339 UTC "Zulu" format, with nanosecond resolution and up to nine fractional digits. Examples: `"2014-10-02T15:01:23Z"` and `"2014-10-02T15:01:23.045123456Z"`.

JSON representation

```
  // metadata
  "videoMetadata": {
    object (VideoMetadata (/api/files#VideoMetadata))
  }
  // Union type
}
```

expirationTime

`string (Timestamp` (https://protobuf.dev/reference/protobuf/google.protobuf/#timestamp) `format)`

Output only. The timestamp of when the `File` will be deleted. Only set if the `File` is scheduled to expire.
A timestamp in RFC3339 UTC "Zulu" format, with nanosecond resolution and up to nine fractional digits. Examples: `"2014-10-02T15:01:23Z"` and `"2014-10-02T15:01:23.045123456Z"`.

sha256Hash

`string (bytes` (https://developers.google.com/discovery/v1/type-format) `format)`

Output only. SHA-256 hash of the uploaded bytes.
A base64-encoded string.

uri `string`

Output only. The uri of the `File`.

state `enum (State` (/api/files#State))

Output only. Processing state of the File.

error `object (Status` (/api/files#v1beta.Status))

Output only. Error status if File processing failed.

metadata `Union type`

Metadata for the File. `metadata` can be only one of the following:

> **`videoMetadata`**
>
> `object (VideoMetadata (/api/files#VideoMetadata))`
>
> Output only. Metadata for a video.

VideoMetadata

Metadata for a video `File`.

Fields

> **`videoDuration`**
>
> `string (Duration (https://protobuf.dev/reference/protobuf/google.protobuf/#duration) format)`
>
> Duration of the video.
> A duration in seconds with up to nine fractional digits, ending with 's'. Example: `"3.5s"`.

JSON representation

```
{
  "videoDuration": string
}
```

State

States for the lifecycle of a File.

Enums	
`STATE_UNSPECIFIED`	The default value. This value is used if the state is omitted.
`PROCESSING`	File is being processed and cannot be used for inference yet.
`ACTIVE`	File is processed and available for inference.
`FAILED`	File failed processing.

Status

The `Status` type defines a logical error model that is suitable for different programming environments, including REST APIs and RPC APIs. It is used by gRPC (https://github.com/grpc). Each `Status` message contains three pieces of data: error code, error message, and error details.

You can find out more about this error model and how to work with it in the API Design Guide (https://cloud.google.com/apis/design/errors).

Fields

JSON representation

```
{
  "code": integer,
  "message": string,
  "details": [
    {
      "@type": string,
      field1: ...,
      ...
    }
  ]
}
```

`code` `integer`

The status code, which should be an enum value of `google.rpc.Code`.

`message` `string`

A developer-facing error message, which should be in English. Any user-facing error message should be localized and sent in the `google.rpc.Status.details` (/api/files#FIELDS.details) field, or localized by the client.

`details[]` `object`

A list of messages that carry the error details. There is a common set of message types for APIs to use.
An object containing fields of an arbitrary type. An additional field `"@type"` contains a URI identifying the type. Example: `{ "id": 1234, "@type": "types.example.com/standard/id" }`.

Last updated 2025-01-13 UTC.