Gemini File‑Upload Migration Guide

=================================

Everything your team needs to switch from hand‑built `file\_data` blobs to the SDK helper `types.Part.fromFile`. Extracted from the May 2025 Google Gemini docs.

------------------------------------------------------------

1  Import the helper

------------------------------------------------------------

```ts

import { GoogleGenAI, types } from "@google/genai";

```

`types.Part.fromFile()` automatically converts camel‑case to the snake‑case proto.

------------------------------------------------------------

2  Upload once, get a URI

------------------------------------------------------------

```ts

const client = new GoogleGenAI({ apiKey: process.env.GEMINI\_API\_KEY });

const uploaded = await client.files.upload({

buffer: fs.readFileSync("/path/or/buffer.pdf"),

mimeType: "application/pdf"

});

console.log(uploaded.uri); // "files/abc123"

console.log(uploaded.mimeType); // "application/pdf"

```

Facts – 2 GB max, 48 h retention, 20 GB per project, upload is free.

------------------------------------------------------------

3  Build a Part with the SDK

------------------------------------------------------------

```ts

const filePart = types.Part.fromFile({

fileUri: uploaded.uri,

mimeType: uploaded.mimeType

});

const res = await client.models.generateContent({

model: "gemini-2.5-flash",

contents: [

{ role: "user", parts: [ filePart, { text: "Give feedback." } ] }

],

config: {

responseMimeType: "application/json",

responseSchema: MY\_SCHEMA

}

});

```

------------------------------------------------------------

4  What the helper outputs

------------------------------------------------------------

```json

{ "file\_data": { "file\_uri": "files/abc123", "mime\_type": "application/pdf" } }

```

No manual snake‑case needed.

------------------------------------------------------------

5  Recommended upload helper

------------------------------------------------------------

```ts

export interface SDKFileData {

fileUri: string;

mimeType: string;

}

export async function uploadOrGetFile(

client: GoogleGenAI,

src: Buffer | string,

mimeType: string

): Promise<SDKFileData> {

const buf = Buffer.isBuffer(src) ? src : await fetchToBuffer(src);

const hash = sha256(buf);

const cached = await redis.get(`gem:file:${hash}`);

if (cached) return JSON.parse(cached);

const f = await client.files.upload({ buffer: buf, mimeType });

const data = { fileUri: f.uri, mimeType: f.mimeType };

await redis.setex(`gem:file:${hash}`, 47\*60\*60, JSON.stringify(data));

return data;

}

```

------------------------------------------------------------

6  Replace manual code

------------------------------------------------------------

```diff

- apiParts.push({ file\_data: fileData });

+ apiParts.push(types.Part.fromFile(fileData));

```

------------------------------------------------------------

Reference snippets

------------------------------------------------------------

• \*Node Libraries page\*   `types.Part.fromFile({...})` → `{ file\_data: ... }`

• \*Files API guide\*       Files stored 48 h, URI format `files/ID`.

• \*Image understanding\*   Upload → `fromFile` example.

• \*Document processing\*   Same pattern for PDFs.