Upload files

The Drive API allows you to upload file data when you create (/drive/api/v3/reference/files/create) or update (/drive/api/v3/reference/files/update) a File (/drive/api/v3/reference/files/) resource.

In this guide and reference, *media* refers to all available files with MIME types that we support for upload to Google Drive. For a list of some of the MIME types that begin with 'application/vnd.google-apps', see create (/drive/api/v3/mime-types). The user can upload any file they want to Drive and some MIME type will be set for the item. If the format of the content can't be detected, then the MIME type will be set to 'application/octet-stream'.

When you upload media, you use a special URI. Methods that support media uploads have two URI endpoints:

- The `/upload` URI, for the media. The format of the `/upload` endpoint is the standard resource URI with an `/upload` prefix. Use this URI when you transfer the media data itself. Example: `POST /upload/drive/v3/files`.
- The standard resource URI, for the metadata. If the resource contains any data fields, those fields are used to store metadata that describes the uploaded file. You can use this URI when you create or update metadata values. Example: `POST /drive/v3/files`.

Upload types

There are three types of uploads you can perform:

- Simple upload: `uploadType=media`. For quick transfer of a small file (5 MB or less). To perform a simple upload, refer to Perform a Simple Upload (#simple).
- Multipart upload: `uploadType=multipart`. For quick transfer of a small file (5 MB or less) and metadata that describes the file, all in a single request. To perform a multipart upload, refer to Perform a Multipart Upload (#multipart).
- Resumable upload: `uploadType=resumable`. For more reliable transfer, especially important with large files. Resumable uploads are a good choice for most applications, since they also work for small files at the cost of one additional HTTP request per upload. To perform a resumable upload, refer to Perform a Resumable Upload (#resumable).

Most Google API client libraries implement at least one of the methods. Refer to the client library documentation for additional details on how to use each of the methods.

Perform a simple upload

A simple upload is the most straightforward way to upload a file. Use this option if:

- the file is small enough to upload again in its entirety if the connection fails.
- there is no metadata to send or if you plan to send metadata in a separate request.

If you need to provide metadata for the file, you can use a multipart upload (#multipart) or resumable upload (#resumable) instead.

For larger files (more than 5 MB) or less reliable network connections, use the resumable upload.

These examples show how to upload an image using the client libraries:

Java (#java)Python (#python)PHP (#php).NETRuby (#ruby)Node.js (#node.js)Objective-C (#objective-c)

```
var fileMetadata = new File()
{
    Name = "photo.jpg"
};
FilesResource.CreateMediaUpload request;
using (var stream = new System.IO.FileStream("files/photo.jpg",
                        System.IO.FileMode.Open))
{
    request = driveService.Files.Create(
        fileMetadata, stream, "image/jpeg");
    request.Fields = "id";
    request.Upload();
}
var file = request.ResponseBody;
Console.WriteLine("File ID: " + file.Id);
```

Send a simple upload request

To use simple upload:

1. Create a **POST** request to the method's **/upload** URI. To update an existing file, use **PUT**.
2. Add the query parameter **uploadType=media**.

   For example:

   ```
   POST https://www.googleapis.com/upload/drive/v3/files?uploadType=media
   ```

3. Add the file's data to the request body.
4. Add these HTTP headers:
   - **Content-Type**. Set to the MIME media type of the object being uploaded.
   - **Content-Length**. Set to the number of bytes you upload. This heading is not required if you use chunked transfer encoding (https://tools.ietf.org/html/rfc7230#section-4.1).
5. Send the request.

Example: Send a simple upload request

This example shows a simple upload request:

```
 https://www.googleapis.com/upload/drive/v3/files?uploadType=media HTTP/1.1
nt-Type: image/jpeg
nt-Length: [NUMBER_OF_BYTES_IN_FILE]
rization: Bearer [YOUR_AUTH_TOKEN]

_DATA]
```

If the request succeeds, the server returns the HTTP **200 OK** status code along with the file's metadata:

```
1.1 200
nt-Type: application/json
```

```
ıme": "Untitled"
```

A blob uploaded to Drive gets "Untitled" as the default title. For information about how to handle errors, refer to Handle errors (#errors).

Perform a multipart upload

A multipart upload request allows you to send metadata along with the data to upload. Use this option if the data you send is small enough to upload again in its entirety if the connection fails.

If your file does not have any metadata, use simple upload (#simple) instead. For larger files (more than 5 MB) or less reliable network connections, use resumable upload (#resumable) instead.

Send a multipart upload request

To use multipart upload:

1. Create a `POST` request to the method's `/upload` URI. To update an existing file, use `PUT`.
2. Add the query parameter `uploadType=multipart`.

   For example:

   ```
   POST https://www.googleapis.com/upload/drive/v3/files?uploadType=multipart
   ```

3. Create the body of the request. Format the body according to the `multipart/related` content type [RFC 2387 (http://tools.ietf.org/html/rfc2387)], which contains two parts:

   a. **Metadata part**. Must come first, and must have a `Content-Type` header set to `application/json; charset=UTF-8`. Add the file's metadata to this part in JSON format.

   b. **Media part**. Must come second, and must have a `Content-Type` header, which may have any MIME type. Add the file's data to this part.

   Identify each part with a boundary string, preceded by two hyphens. In addition, add two hyphens after the final boundary string.

4. Add these top-level HTTP headers:

   - `Content-Type`. Set to `multipart/related`, and include the boundary string you're using to identify the different parts of the request. For example: `Content-Type: multipart/related; boundary=foo_bar_baz`
   - `Content-Length`. Set to the total number of bytes in the request body.

5. Send the request.

To create or update the metadata portion only, without the associated data, send a **`POST`** or **`PUT`** request to the ırd resource endpoint: **`https://www.googleapis.com/drive/v3/files`**

Example: Send a multipart upload request

This example shows a multipart upload request:

```
 https://www.googleapis.com/upload/drive/v3/files?uploadType=multipart HTTP/1.1
rization: Bearer [YOUR_AUTH_TOKEN]
nt-Type: multipart/related; boundary=foo_bar_baz
nt-Length: [NUMBER_OF_BYTES_IN_ENTIRE_REQUEST_BODY]

_bar_baz
nt-Type: application/json; charset=UTF-8


me": "myObject"


_bar_baz
nt-Type: image/jpeg

_DATA]
_bar_baz--
```

If the request succeeds, the server returns the HTTP **200 OK** status code along with the file's metadata:

```
'1.1 200
:nt-Type: application/json


ıme": "myObject"
```

To handle errors, refer to Handle errors (#errors).

Perform a resumable upload

This protocol allows you to resume an upload operation after a communication failure interrupts the flow of data. Use this option if:

- You transfer large files.
- The likelihood of a network interruption or some other transmission failure is high (for example, if you upload a file from a mobile app).

Resumable uploads can also reduce your bandwidth usage when there is a network failure, because you don't have to restart large file uploads from the start.

If you send small files over a reliable network connection, you can use simple upload (#simple) or multipart upload (#multipart) instead.

Learn about request URIs

When you upload media, you use a special URI. In fact, methods that support media uploads have two URI endpoints:

- The `/upload` URI, for the media. The format of the `/upload` endpoint is the standard resource URI with an `/upload` prefix. Use this URI when you transfer the media data itself. Example: `POST /upload/drive/v3/files`.

- The standard resource URI, for the metadata. If the resource contains any data fields, those fields store metadata that describes the uploaded file. You can use this URI to create or update metadata values. Example: `POST /drive/v3/files`.

Initiate a resumable upload session

To initiate a resumable upload session:

1. Create a request to the method's `/upload` URI. To create a new file, use `POST`. To update an existing file, use `PUT`.
2. Add the query parameter `uploadType=resumable`.

   For example:

   ```
   POST https://www.googleapis.com/upload/drive/v3/files?uploadType=resumable
   ```

   or:

   ```
   PUT https://www.googleapis.com/upload/drive/v3/files/[FILE_ID]?uploadType=r
   ```

3. If you have metadata for the file, add the metadata to the request body in JSON format. Otherwise, leave the request body empty.
4. Add these HTTP headers:
   - `X-Upload-Content-Type`. Optional. Set to the MIME type of the file data, which will be transferred in subsequent requests. If the MIME type of the data is not specified in metadata or through this header, the object will be served as `application/octet-stream` (https://tools.ietf.org/html/rfc2046#section-4.5.1).
   - `X-Upload-Content-Length`. Optional. Set to the number of bytes of file data, which will be transferred in subsequent requests.
   - `Content-Type`. Required if you have metadata for the file. Set to `application/json; charset=UTF-8`.

- `Content-Length`. Required unless you use chunked transfer encoding (https://tools.ietf.org/html/rfc7230#section-4.1). Set to the number of bytes in the body of this initial request.

5. Send the request.

Example: Initiate a resumable upload session

This example shows how to initiate a resumable session to upload a new file:

```
 https://www.googleapis.com/upload/drive/v3/files?uploadType=resumable HTTP/1.1
rization: Bearer [YOUR_AUTH_TOKEN]
nt-Length: 38
nt-Type: application/json; charset=UTF-8
oad-Content-Type: image/jpeg
oad-Content-Length: 2000000


me": "myObject"
```

The next section describes how to handle the response.

Save the resumable session URI

If the session initiation request succeeds, the response includes a `200 OK` HTTP status code. In addition, it includes a `Location` header that specifies the resumable session URI. Use the resumable session URI to upload the file data and query the upload status.

A resumable session URI expires after one week.

Copy and save the resumable session URI so you can use it for subsequent requests.

Example: Save the resumable session URI

This example shows a response that includes a resumable session URI:

```
'1.1 200 OK
:ion: https://www.googleapis.com/upload/drive/v3/files?uploadType=resumable&uploa
:nt-Length: 0
```

Upload the file

There are two ways to upload a file with a resumable session:

1. **In a single request.** This approach is usually best, since it requires fewer requests and thus has better performance.
2. **In multiple chunks.** Use this approach if:
   - You need to reduce the amount of data transferred in any single request. You might need to do this when there is a fixed time limit for individual requests, as is true for certain classes of Google App Engine requests.
   - You need to provide a customized indicator to show the upload progress.

Single requestMultiple chunks (#multiple-ch...

To upload the file in a single request:

1. Create a `PUT` request to the resumable session URI.
2. Add the file's data to the request body.
3. Add a `Content-Length` HTTP header, set to the number of bytes in the file.
4. Send the request.

If the upload request is interrupted, or if you receive a `5xx` response, follow the procedure in Resume an interrupted upload (#resume-upload).

Example: Upload the file

Single requestMultiple chunks (#multiple-ch...

This example shows a resumable request to upload an entire 2,000,000-byte JPEG file, and uses the resumable session URI obtained in the previous step:

```
PUT https://www.googleapis.com/upload/drive/v3/files?uploadType=resumable&upload
Content-Length: 2000000
Content-Type: image/jpeg

[BYTES 0-1999999]
```

If the request succeeds, you receive a `200 OK` or `201 Created` response, along with any metadata associated with the resource.

Resume an interrupted upload

If an upload request is terminated before a response, or if you receive a `503 Service Unavailable` response, then you need to resume the interrupted upload. To do this:

1. To request the upload status, create an empty `PUT` request to the resumable session URI.
2. Add a `Content-Range` header to indicate that the current position in the file is unknown.

   For example, set the `Content-Range` to `*/2000000` if your total file length is 2,000,000 bytes.

   If you don't know the full size of the file, set the `Content-Range` to `*/*`.
3. Send the request.
4. Process the response.
   - A `200 OK` or `201 Created` response indicates that the upload was completed, and no further action is necessary.
   - A `308 Resume Incomplete` response indicates that you need to continue to upload the file.
   - A `404 Not Found` response indicates the upload session has expired and the upload needs to be restarted from the start.
5. If you received a `308 Resume Incomplete` response, process the response's `Range` header, which specifies which bytes the server has received so far. The response will not

have a **Range** header if no bytes have been received yet.

For example, a **Range** header of **bytes=0-42** indicates that the first 43 bytes of the file have been received.

6. Now that you know where to resume the upload, continue to upload the file, either send the data that remains or send the next chunk. Include a **Content-Range** header to indicate which portion of the file you send.

   For example, **Content-Range: bytes 43-1999999/2000000** indicates that you send bytes 43 through 1,999,999.

Example: Resume an interrupted upload

This example shows a request for the upload status:

```
ıttps://www.googleapis.com/upload/drive/v3/files?uploadType=resumable&upload_id=x
ınt-Length: 0
ınt-Range: bytes */2000000
```

The server's response uses the **Range** header to indicate that it has received the first 43 bytes of the file so far:

```
'1.1 308 Resume Incomplete
ınt-Length: 0
ː: bytes=0-42
```

You can then send a request to resume the upload. Send the bytes that remain in the file. Start at byte 43:

```
ıttps://www.googleapis.com/upload/drive/v3/files?uploadType=resumable&upload_id=x
ınt-Length: 1999957
ınt-Range: bytes 43-1999999/2000000

:S 43-1999999]
```

Handle errors

When you upload media, be sure to follow these best practices related to handle errors:

- Resume or retry uploads that fail due to connection interruptions or any 5xx errors, for example:
    - 500 Internal Server Error
    - 502 Bad Gateway
    - 503 Service Unavailable
    - 504 Gateway Timeout
- Resume or retry uploads that fail due to 403 rate limit errors.
- Use an exponential backoff strategy if any 403 or 5xx server error is returned when upload requests retry or resend. These errors can occur if a server gets overloaded. Exponential backoff can help alleviate these kinds of problems when there is a high volume of requests or heavy network traffic.
- (Resumable uploads only) Restart uploads if a `404 Not Found` error is received after it resumes or uploads a chunk. This indicates the upload session has expired and must be restarted from the start. Upload sessions expire after 1 week of inactivity.

For additional details, see Handling API Errors (/drive/api/v3/handle-errors).

Import to Google Docs types

When you create a file in Google Drive, you can convert some types file into a Google Docs, Sheets or Slides document. Specify the `mimeType` file property. This sample shows how to upload a CSV file as a spreadsheet:

Java (#java)Python (#python)PHP (#php).NETRuby (#ruby)Node.js (#node.js)Objective-C (#objective-c)

```
var fileMetadata = new File()
{
    Name = "My Report",
    MimeType = "application/vnd.google-apps.spreadsheet"
};
FilesResource.CreateMediaUpload request;
using (var stream = new System.IO.FileStream("files/report.csv",
                        System.IO.FileMode.Open))
{
    request = driveService.Files.Create(
        fileMetadata, stream, "text/csv");
    request.Fields = "id";
    request.Upload();
}
var file = request.ResponseBody;
Console.WriteLine("File ID: " + file.Id);
```

The supported conversions are available dynamically in the About resource's (/drive/api/v3/reference/about#resource) `importFormats` array and include:

From	To
Microsoft Word, OpenDocument Text, HTML, RTF, plain text	Google Docs
Microsoft Excel, OpenDocument Spreadsheet, CSV, TSV, plain text	Google Sheets
Microsoft Powerpoint, OpenDocument Presentation	Google Slides
JPEG, PNG, GIF, BMP, PDF	Google Docs (embeds the image in a Doc)
plain text (special MIME type), JSON	Google Apps Script

When you upload and convert media during an `update` request to a Google Doc, Sheet, or Slide the full contents of the document will be replaced.

When you convert images you can improve the quality of the OCR algorithm. Specify the applicable BCP 47 (https://tools.ietf.org/html/bcp47) language code in the `ocrLanguage` (/drive/api/v3/reference/files/create#ocrLanguage) parameter. The extracted text will appear in the Google Docs document alongside the embedded image.

Use a pregenerated ID to upload files

The Drive API allows you to retrieve a list of pregenerated file IDs that can be used to upload and create resources. Upload and file creation requests can then include these pregenerated

IDs. Set the `id` fields in the file metadata.

You can safely retry uploads with pregenerated IDs in the case of an indeterminate server error or timeout. If the file was successfully created, subsequent retries return a `HTTP 409` error, it does not create duplicate files.

Note: Pregenerated IDs are not supported for native Google Document creation, or uploads where conversion to native Google Document format is requested.