REST API: Attaching Files

Setup and Reference · https://mriacrm.com/docs/rest-api-attaching-files/

Attach documents to Leads, Deals, Contacts and Companies from your own code — signed contracts from an e-signature tool, or a one-off load of every customer's paperwork from the system you are moving away from. A file attached this way is the same as one uploaded by hand: it shows in the record's **Files** tab, where anyone who can open the record can download it.

1. Endpoints

The same five endpoints exist for each of `leads`, `deals`, `contacts` and `companies`. Products have no files, in the API or in the app. Paths are relative to the API base, `https://api.atlassian.com/svc/jira/{cloudId}/apps/{appId}_{envId}/v1` — see REST API: Example Node.js.

Method	Path	Does
GET	`/{records}/{id}/files`	Lists the record's files: name, type, size, who added it and when. Paged with startAt and maxResults (up to 200).
POST	`/{records}/{id}/files`	Starts an upload and returns an upload URL. Step 1 of 3.
POST	`/{records}/{id}/files/{fileId}/confirm`	Attaches the uploaded file to the record. Step 3 of 3.
GET	`/{records}/{id}/files/{fileId}`	One file, with a short-lived downloadUrl.
DELETE	`/{records}/{id}/files/{fileId}`	Removes the file from the record.

Scopes and permissions

Calls	OAuth scope	CRM permission
List, get	`read:crm:custom`	View on that record type

Calls	OAuth scope	CRM permission
Start, confirm, delete	`write:crm:custom`	Edit on that record type

Writing also needs an active Mria CRM license. The integration from REST API: Setup must have `write:crm:custom` enabled and the token must ask for it — REST API: Example Node.js asks only for read:crm:custom, so add it to its scope list before uploading.

2. Uploading a file

An upload is three calls. The file itself does not travel through the API: you get a one-time upload URL, send the bytes there, then confirm.

1. **Start** — POST to `/{records}/{id}/files` with the file's `name`, `type`, `size` and `checksum`. The answer carries the file's `id` and an `uploadUrl`.
2. **Send the bytes** — PUT the file to `uploadUrl` exactly as it is, with `Content-Type: application/octet-stream` and **no Authorization header**. A form-encoded body is rejected with 415.
3. **Confirm** — POST `name` and `type` to `/{records}/{id}/files/{fileId}/confirm`. Only now does the file appear on the record.

Both the upload URL and the confirmation expire after **one hour**, and the confirmation must come from the same user who started the upload.

What the start call checks

Field	Rule
`name`	1–255 characters, ending in .pdf, .doc, .docx, .xlsx, .txt, .csv, .png, .jpg, .jpeg, .gif, .json or .zip. It is the name shown in the Files tab.
`type`	The matching MIME type, for example `application/pdf` or `image/png`. The spec lists all eleven.
`size`	The exact size in bytes, at most 10 MB (10,485,760) — the same limit as the Files tab.
`checksum`	SHA-256 of the file, **Base64-encoded** — 44 characters, not the 64-character hex form. The upload URL accepts only bytes that match it and the size.

From the command line

```
# One file onto one Contact. TOKEN is a 3LO access token with write:crm:custom.
API="https://api.atlassian.com/svc/jira/$CLOUD_ID/apps/f3bb4fc8-7911-405b-959b-
002b3292a3d0_$ENV_ID/v1"
FILE="NDA-signed.pdf"
SIZE=$(wc -c < "$FILE" | tr -d ' ')
SUM=$(openssl dgst -sha256 -binary "$FILE" | base64)    # 44 characters, not hex

# 1. Start the upload -> {"id": "...", "uploadUrl": "..."}
curl -s -X POST "$API/contacts/$CONTACT_ID/files" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d "{\"name\":\"$FILE\",\"type\":\"application/pdf\",\"size\":$SIZE,\"checksum\":\"$SUM\"}"

# 2. Send the bytes - to uploadUrl, with NO Authorization header
curl -s -X PUT "$UPLOAD_URL" \
  -H "Content-Type: application/octet-stream" --data-binary @"$FILE"

# 3. Confirm - the file now shows in the Contact's Files tab
curl -s -X POST "$API/contacts/$CONTACT_ID/files/$FILE_ID/confirm" \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d "{\"name\":\"$FILE\",\"type\":\"application/pdf\"}"
```

From Node.js

A function to drop into REST API: Example Node.js, where `API_BASE` and the access token already exist:

```
// Attach one local file to a record. Node 18+, no dependencies.
// API_BASE and the token come from the Example Node.js page; the token
// needs write:crm:custom as well as read:crm:custom.
import { readFile } from "node:fs/promises";
import { createHash } from "node:crypto";
import { basename, extname } from "node:path";

const TYPES = {
  ".pdf": "application/pdf", ".doc": "application/msword",
  ".docx": "application/vnd.openxmlformats-officedocument.wordprocessingml.document",
  ".xlsx": "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
  ".txt": "text/plain", ".csv": "text/csv", ".png": "image/png",
```

```
  ".jpg": "image/jpeg", ".jpeg": "image/jpeg", ".gif": "image/gif",
  ".json": "application/json", ".zip": "application/zip",
};

export async function attachFile(records, recordId, path, token) {
  const bytes = await readFile(path);
  const name = basename(path);
  const type = TYPES[extname(name).toLowerCase()];
  if (!type) throw new Error(`${name}: file type not accepted`);
  const auth = { Authorization: `Bearer ${token}`, "Content-Type": "application/json" };
  const base = `${API_BASE}/${records}/${recordId}/files`;

  // 1. start
  const start = await fetch(base, { method: "POST", headers: auth, body: JSON.stringify({
    name, type, size: bytes.length,
    checksum: createHash("sha256").update(bytes).digest("base64"),
  }) });
  if (start.status !== 201) throw new Error(`${name}: start ${start.status} ${await
start.text()}`);
  const { id, uploadUrl } = await start.json();

  // 2. bytes - no Authorization header on the presigned URL
  const put = await fetch(uploadUrl, { method: "PUT",
    headers: { "Content-Type": "application/octet-stream" }, body: bytes });
  if (!put.ok) throw new Error(`${name}: upload ${put.status}`);

  // 3. confirm
  const done = await fetch(`${base}/${id}/confirm`, { method: "POST", headers: auth,
    body: JSON.stringify({ name, type }) });
  if (done.status !== 201) throw new Error(`${name}: confirm ${done.status} ${await
done.text()}`);
  return id;
}

// Bulk: one call per file, one file at a time.
// for (const { contactId, path } of rows) await attachFile("contacts", contactId, path, token);
```

Bulk uploads are a loop

There is no endpoint that takes several files at once: each file is its own start, upload and

confirm. To load a folder of documents, match each file to its record first — by the Contact's email or the Company's name, through the search endpoints — and upload one at a time.

3. Listing, downloading and deleting

- **List** — `GET /{records}/{id}/files` returns each file's `id`, `name`, `type`, `size`, `createdBy` and `createdAt`. Page through with `startAt` and `maxResults` until `isLast` is true.
- **Download** — `GET /{records}/{id}/files/{fileId}` adds a `downloadUrl`. It is short-lived: fetch it straight away, again without an Authorization header.
- **Delete** — `DELETE /{records}/{id}/files/{fileId}` answers 204. The file is gone from the Files tab.

Related Pages

- REST API: Reference — the request and response schemas for these endpoints, under Lead files, Deal files, Contact files and Company files.
- REST API: Setup — the integration, scopes and API base.
- Activities Breakdown — the Files tab these files appear in.

This API is supported and actively developed, so it can change without notice. It builds on Atlassian's Forge App REST APIs (Preview). Send feedback to the Mria CRM team.