AI Recorder App

Upload to your own server

The Android app can send finished recordings to your own HTTPS server. This guide first shows how to set that up in the app, then what your server must accept.

Applies to Android PRO 1.1.17-pro and PLAY 1.1.18 and later. Dated 12 November 2026.

1. What this is

In Settings you will find “Upload to your server”. The description reads: “Finished recordings can be POSTed to your own HTTPS server.”

Please note:

- **You enter the upload URL yourself**. It must start with https://. The app does not pre-fill an address. An invalid address shows “Only https:// URLs are allowed” or “Invalid HTTPS URL”.
- **“Upload recordings automatically” and the other upload switches are off by default.** Turn them on when you need them, and use “Test connection” to check that your server can be reached.
- You may optionally enter an access token (sent as Authorization: Bearer, stored only in the phone's secure storage, never written to logs), an uploader name, and an email (optional). The name and email are used only on your upload server; no app account is required.
- You can also turn on “Upload all-day segments” and “Upload downloaded recordings too”.
- The file that is sent is playable OGG audio (Opus in an OGG container), together with a JSON description (device, start and end time, duration, size, checksum, and so on). While a transfer is in progress you may see “Queued”, “Uploading”, “Uploaded”, or “Upload failed”.

Ask your own technical colleagues to prepare the receiving URL, whether a token is required, and how the server stores the files. Do not send secrets in a chat.

2. How to set it up

1. Open the app and tap *Settings* at the bottom.
2. Scroll down. After “App language” you will see “Upload to your server”.

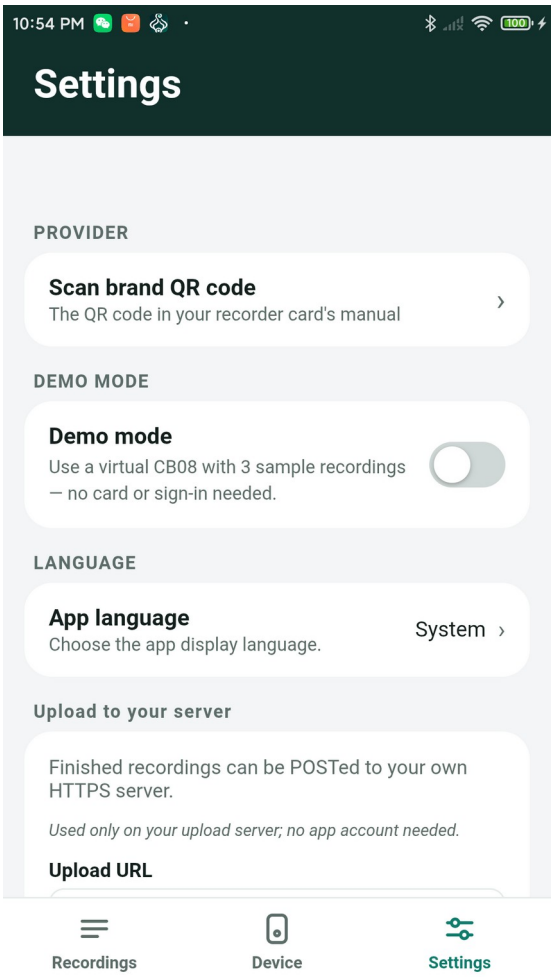


Figure 1 Settings: scroll down and “Upload to your server” appears below the language setting.

3. In “Upload URL”, enter your server address. It must start with `https://`, for example `https://example.com/upload`. Do not use `http://`.
4. If your server requires authentication, enter a token in “Access token”. After it is saved, the field itself stays empty and you will only see “Token saved”. To replace it, tap “Clear” first, then type the new token. The token is stored only in this phone’s secure storage and is never logged. The token is optional; leave it blank if your server does not need one.
5. Optionally fill in “Uploader name” (up to 64 characters, for example Jane or Sales-03) and “Email (optional)”. These values are sent only to your upload server; you do not need to sign in to an app account.
6. Turn on “Upload recordings automatically”. This is the master switch: nothing is uploaded while it is off.

7. Then turn on the following as needed (they can be turned on only when the master switch is on and the URL is valid):

- **“Upload all-day segments”**: also upload all-day recording segments.
- **“Upload downloaded recordings too”**: also upload recordings downloaded from the recorder card, and other recordings that are not all-day segments.

8. Tap “Test connection”. Success shows “Connection OK”; failure shows “Connection failed: …”. The test only checks that your server can be reached; it does not upload a recording.

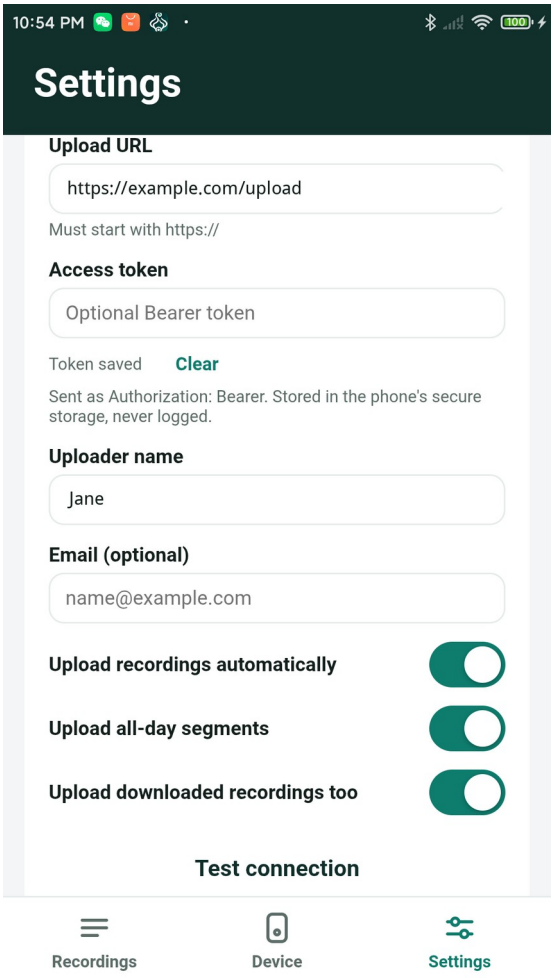


Figure 2 Enter the URL, an optional token and name, turn on the switches you need, then tap Test connection.

Note: the screenshot shows a completed setup — these switches are off by default. The address in the picture has been replaced with the example https://example.com/upload; use your own server URL.

3. What you will see in the list

Each item in the recordings list may show one of these statuses:

- **Queued**

- **Uploading**
- **Uploaded**
- **Upload failed** (tap “Retry” to try again)

If the phone is offline, items stay queued and continue automatically when the network returns. After a failure the app retries with exponential backoff (starting at about 1 second, up to about 5 minutes). Unfinished work is kept if you close and reopen the app.

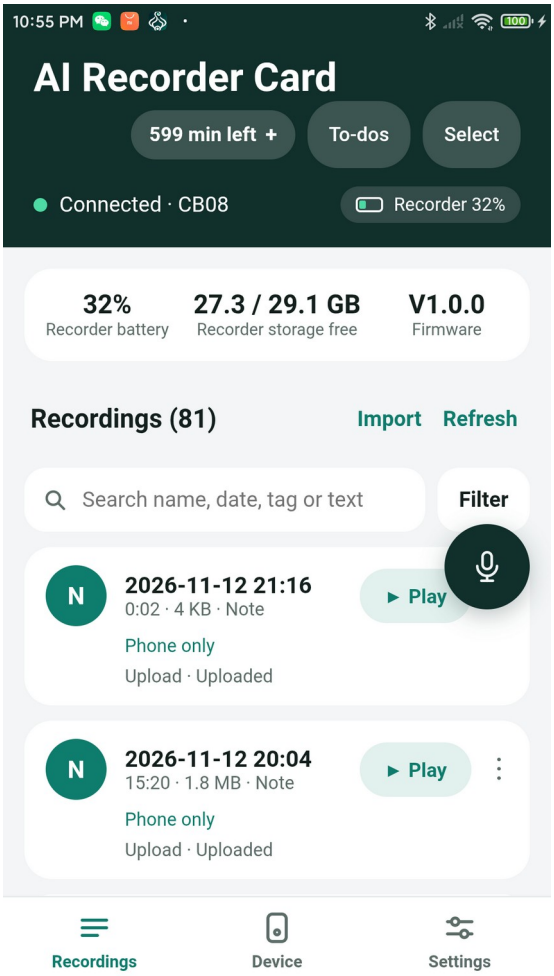


Figure 3 Upload status on the recordings list, for example “Upload · Uploaded”.

4. For your technical colleagues: what the server must accept

The app can send finished recordings to your own HTTPS server. Upload is off by default. Nothing is sent until the user enters an HTTPS URL and turns the feature on. No vendor account or app login is required for this feature.

The official Android and iOS app sends both the “Test connection” probe and file uploads through the native HTTP stack, not the WebView. Receivers do not need Access-Control-

Allow-* headers for the official app. CORS only matters if a browser (for example a page on a different origin) calls the same URL.

Request format

POST to the URL the user configured. HTTPS only (http:// is rejected). File uploads use multipart/form-data.

Headers:

Header	Required	Notes
`Authorization`	No	Bearer <token> when the user saved an access token. Do not log the token.
`X-Device-Id`	Yes	Automatic device identifier (may be empty if no device has been paired yet).
`X-Uploader-Name`	Yes	Uploader name, URL-encoded (for example, a name with spaces or non-Latin characters is percent-encoded). Empty when the name was left blank.
`Content-Type`	Yes	multipart/form-data, with a client-generated boundary.

Multipart parts:

- **`file`** (required) — the audio bytes: Opus audio in an OGG container, the same playable file the app keeps on the phone. Content type audio/ogg. Filename {recId}.ogg. The bytes are directly playable in a browser, ffplay, ffprobe, or any Opus player.
- **`meta`** (required) — a JSON object (Content-Type: application/json), fields below. Some clients may also send this part as metadata; accept either name.

Note: app versions before 1.1.16 sent the recorder card's raw 40-byte Opus frames as {recId}.opus with format "opus". Those frames have no OGG container, so ffprobe reports "Invalid data". If you receive one, wrap it in an OGG container, or ask the sender to re-upload from a current app.

Metadata JSON fields:

Field	Type	Description
`deviceId`	string	Automatic device identifier. May be an empty string.
`uploaderName`	string	User-editable display name, trimmed. May be empty.

Field	Type	Description
`uploaderEmail`	string	Optional email, trimmed. May be empty. If non-empty, the app checks a basic email shape before sending.
`model`	string	Device model (for example cb08). May be empty for phone-only recordings.
`recId`	string	Recording id (also used as the file stem).
`startAt`	string	Start time, ISO-8601 UTC — true UTC: the app converts the recording's device-local wall-clock start. Do not assume the local digits appear unchanged with a trailing Z.
`endAt`	string	End time, ISO-8601 UTC (startAt + duration).
`durationSec`	number	Duration in whole seconds.
`format`	string	ogg (matches the file part). Legacy uploads may say opus.
`bytes`	number	Size of the uploaded file part (the .ogg bytes).
`sha256`	string	Lowercase hex SHA-256 of the exact file-part bytes (the .ogg), not the multipart wrapper. May be empty if the client could not hash; accept that case.
`appVersion`	string	App version string.
`tags`	string[]	App-assigned labels for this recording (optional).

Example (a recording that started at 10:00 local time in Asia/Shanghai; startAt is true UTC, so 02:00Z):

```
{
  "deviceId": "CB08-1234",
  "uploaderName": "Sales-03",
  "uploaderEmail": "alex@example.com",
  "model": "cb08",
  "recId": "note20261110-100000",
  "startAt": "2026-11-10T02:00:00.000Z",
  "endAt": "2026-11-10T03:00:00.000Z",
  "durationSec": 3600,
  "format": "ogg",
  "bytes": 480000,
  "sha256": "b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9",
  "appVersion": "1.1.16",
  "tags": []
}
```

Success, retries, and Test connection

- Treat any HTTP 2xx (200, 201, 204, …) as success.
- Any other status, a redirect, or a network error is a failure. The app keeps the item in a persistent queue and retries with exponential backoff.

The in-app “Test connection” button sends a small JSON POST to the same URL (not multipart):

```
POST /upload
Content-Type: application/json
Authorization: Bearer <token>          (if configured)
X-Device-Id: <deviceId>
X-Uploader-Name: <url-encoded-name>
X-Upload-Test: 1

{"test":true,"ts":1731283200000}
```

Respond with HTTP 2xx to show “Connection OK” in the app.

Suggested folders per card or per person

A typical receiver writes files under recordings/<uploaderName or deviceId>/, sanitizes the folder name so “..”, slashes, and other path characters cannot escape the recordings directory, and keeps the uploaded filename extension as sent (.ogg) — never renaming an .ogg upload to .opus.

- If you file by device ID (deviceId), recordings from different cards land in different folders.
- If the user entered an uploader name, you can file by person (the same person with different cards can share one folder).
- When the client sends a sha256, verify it against the saved bytes; accept an empty sha256. On a mismatch, log it; you need not reject the file.

Terminate TLS in front of the receiver, because the app requires an https:// URL. Add authentication, size limits, and durable storage before using this in production. Do not write the access token to server logs.