Part 1 — What the function does

What it does. The REST API lets another program read and write the maintenance records of the CMMS database ReliaEn has open: a plant system that raises work orders, a report server that reads them, the technicians' phones. ReliaEn serves it from your computer once you start the service in the window Technician App and API. Every call signs in as a ReliaEn user and acts with that user's role, exactly as the same user would in the desktop windows.

What a program does	What the API answers
Signs in with a user name and password	An access token valid 8 hours, a refresh token valid 30 days, the user's role
Calls any route with the access token	The answer as JSON; the same records and the same rules as the desktop
Calls with no token, a wrong password or an unknown user	One refusal for all three, Invalid user or password. - nothing tells which part was wrong
Calls after the access token has ended	Session expired at …; sign in again.
Asks in Thai (Accept-Language: th)	The same answer with every message in Thai; numbers and codes unchanged

The full list of routes is published by the service itself at /openapi.json, grouped under ten tags: auth, assets, workorders, pm, fracas, spares, feedback, exchange, kpi and sync.

From which data. The CMMS database ReliaEn has open - a *.reliaen* file, or a PostgreSQL database whose connection ReliaEn has saved - and its user accounts.

When an engineer uses it. When records should flow between the CMMS and another program without typing them twice: work orders raised by a plant system, results read by a reporting tool, a phone app of your own.

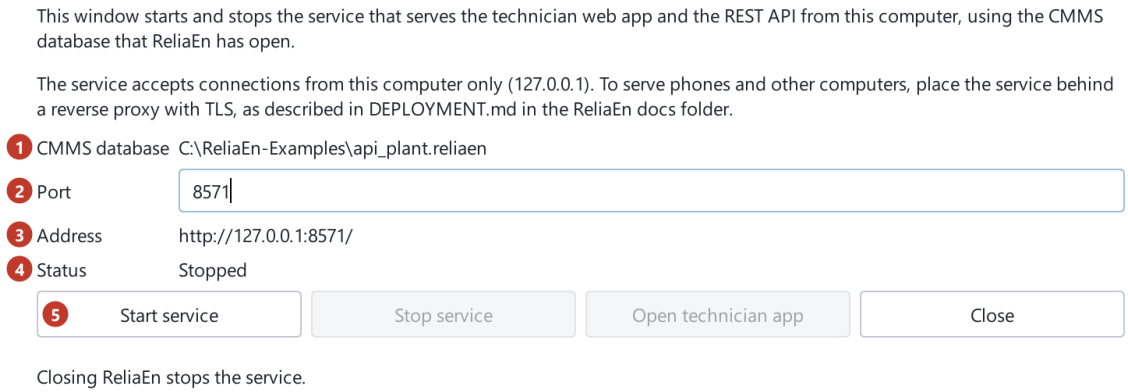
When not to use it.

Situation	Use instead
Technicians record their work on a phone	The technician web app the same service serves - see the chapter *The technician web app*
A file of records is to be loaded once	CMMS ▸ Import Wizard… - see the chapter *Import wizard*
Phones or other computers must reach the service	The service accepts connections from this computer only (127.0.0.1); place it behind a reverse proxy with TLS, as described in DEPLOYMENT.md in the ReliaEn docs folder

Part 2 — The example's input data

Setting. The plant's IT engineer connects a reporting tool to the CMMS. Before writing the connection, the engineer signs in as the maintenance supervisor, reads the session the API returns, and checks how the API refuses a wrong password, a call with no token and a token that has ended.

Input	Value
CMMS database	A CMMS file with five user accounts: `u-sup` (supervisor), `u-tech` (technician), `u-eng`, `u-store` and `u-admin`
User name and password	`u-sup` with the password `longenough-pw`
Site	`T1`
Clock	Sign-in at 2026-09-01 08:00 UTC

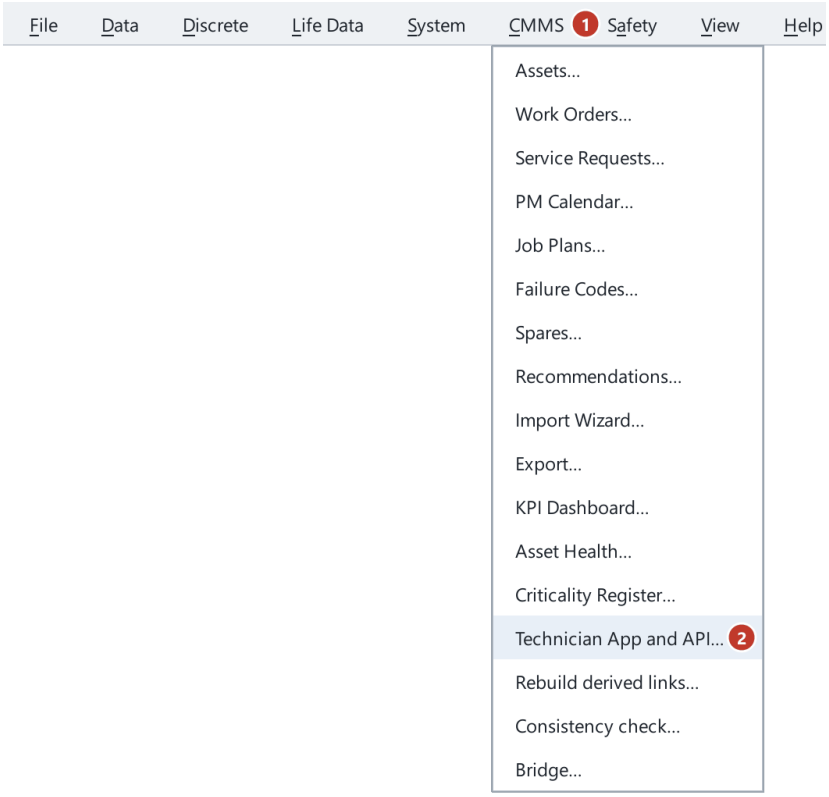


The example's database, as the service window names it

- ① `CMMS database`: `C:\ReliaEn-Examples\api_plant.reliaen`, the example's file with the five user accounts.
- ② `Port`: 8571.
- ③ `Address`: `http://127.0.0.1:8571/`.
- ④ `Status`: `Stopped`.
- ⑤ `Start service`.

Part 3 — How to use it, step by step

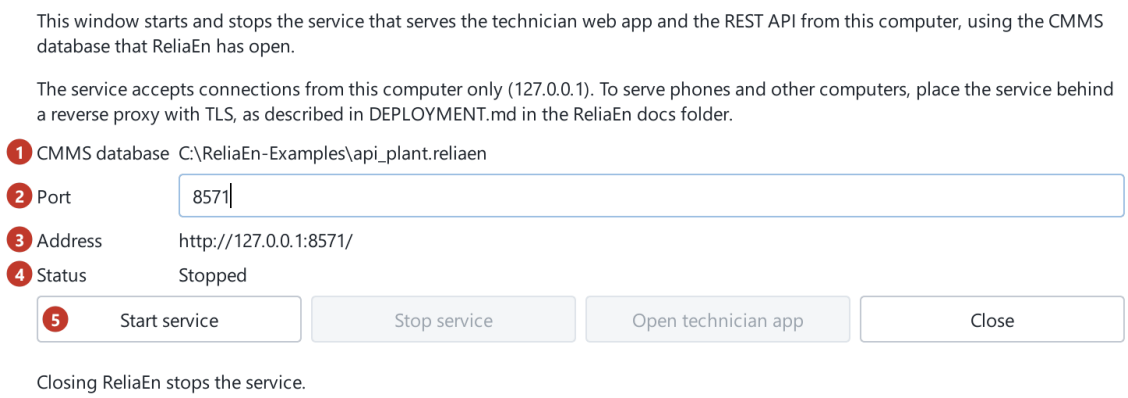
1. **Open the CMMS database** in ReliaEn, then **choose** **CMMS** ▸ **Technician App and API…** in the menu bar.



Step 1 — the menu path

- ① CMMS: the menu.
- ② Technician App and API…: the entry to click.

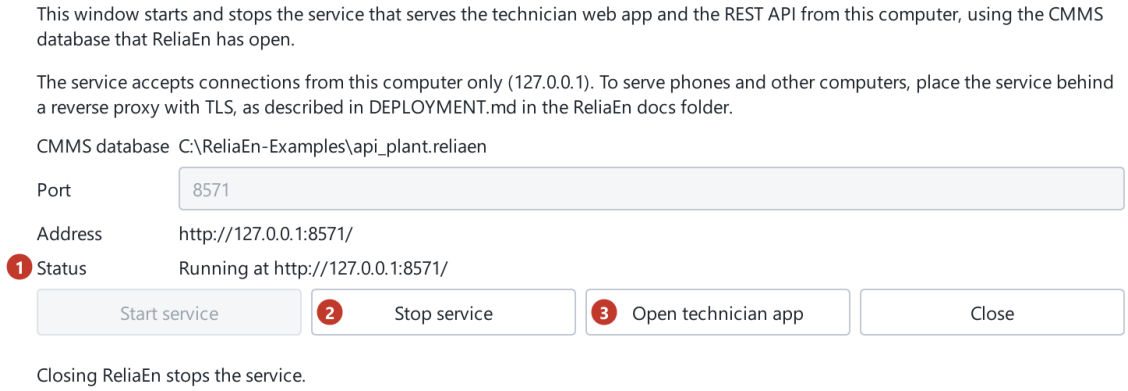
2. **Check the window's fields, then click Start service.**



Step 2 — the window as it opens

- ① CMMS database: the database ReliaEn has open - the one the service will use.
- ② Port: 8571, the default; any whole number from 1024 to 65535 that no other program uses.
- ③ Address: http://127.0.0.1:8571/ - where programs on this computer reach the service.
- ④ Status: Stopped.
- ⑤ Start service: starts it.

3. **Wait for Running at http://127.0.0.1:8571/.** The service now answers; it keeps running after you close the window, until you click Stop service, open another CMMS database or close ReliaEn.



Step 3 — the service running

- ① Status: Running at http://127.0.0.1:8571/.
- ② Stop service: stops it.
- ③ Open technician app: opens the technician web app in your browser.

4. ① **Sign in from your program** with a POST to /api/auth/login at the address the status line shows (① above):

```
POST http://127.0.0.1:8571/api/auth/login
Content-Type: application/json

{"username": "u-sup", "password": "longenough-pw", "tenant_id": "T1"}
```

5. ① **Send the access token with every other call** to the same address, in the header Authorization: Bearer <access_token> - here, to read your own session:

```
GET http://127.0.0.1:8571/api/auth/me
Authorization: Bearer <access_token>
```

Part 4 — The result, call by call

The sign-in (step 4) answers 200 with the session:


```
{"access_token": "eyJhbGciOiJIUzI1NiIs…",
 "refresh_token": "eyJhbGciOiJIUzI1NiIs…",
 "user_id": "79c167ee59744cfabeda903e92f62370",
 "role": "supervisor",
 "exp": "2026-09-01T16:00:00Z",
 "refresh_exp": "2026-10-01T08:00:00Z",
 "api": "API 1.0.2 · schema 17."}
```

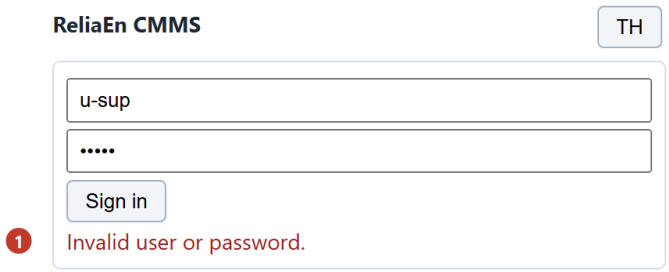

- access_token: the key for every other call. It ends at exp, 2026-09-01T16:00:00Z - 8 hours after the sign-in at 08:00.
- refresh_token: gets a new access token from /api/auth/refresh without the password, until refresh_exp, 30 days on.
- role: supervisor - what this user may do; the API reads the role from the user's account at every call.
- api: the service's version and the database layout it runs on.

Your session (step 5) answers 200: {"user_id": "79c167ee59744cfabeda903e92f62370", "role": "supervisor", "tenant_id": "T1", "hourly_rate": 0.0, "api": "API 1.0.2 · schema 17."}.

The refusals. Every refusal is a JSON body with a code and a message:

Call	Answer
Sign-in with the password wrong	401 {"code": "E-18", "message": "Invalid user or password."}
Sign-in as nobody, a user that does not exist	401 {"code": "E-18", "message": "Invalid user or password."} - the same answer
POST /api/workorders/list_open with no token	401 {"code": "E-18", "message": "Invalid user or password."}
The same, with Accept-Language: th	401 {"code": "E-18", "message": "ผู้ใช้หรือรหัสผ่านไม่ถูกต้อง"}
GET /api/auth/me with the morning's token at 16:00 UTC	401 {"code": "E-19", "message": "Session expired at 2026-09-01 23:00; sign in again."}

The technician web app (chapter *The technician web app*) is one program that calls this API; this is how it shows the E-18 answer to its user:



The E-18 answer, as the technician web app shows it

- ① Invalid user or password. - the API's message, word for word.

The published list of routes. GET /openapi.json answers the OpenAPI 3.1.0 document of ReliaEn CMMS API version 1.0.2: 101 routes under the ten tags of Part 1.

Part 5 — How to interpret the results

What the sign-in gives you. A session of 8 hours for one user. Your program keeps the access token, sends it on every call, and before `exp` asks `/api/auth/refresh` for a new one with the refresh token. After `refresh_exp`, or after the user signs out, it signs in again with the password.

Why one refusal for three mistakes. A wrong password, an unknown user and a missing token all answer `Invalid user or password.`: a caller cannot learn from the API which user names exist. When your program receives E-18, check the user name, the password and that the header `Authorization` is sent.

Reading E-19. The token has ended; the time in the message is the computer's local time - here 23:00 at UTC+7 is the token's `exp` of 16:00 UTC. Sign in again, or refresh before the end.

Roles decide, not the program. A call the user's role may not make is refused with `Role … may not …`, naming the role and the action; the same user gets the same answer in the desktop windows. Give each program its own user with the smallest role it needs.

Who can reach the service. Programs on the computer that runs ReliaEn. For phones and other computers, the service goes behind a reverse proxy with TLS, as DEPLOYMENT.md describes - never opened directly to the network.