AI-03995 OpenEMR BKM Upload

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	July 5, 2026 5:50 PM

What is the bkm to upload to OpenEMR patients information? Does OpenEMR support csv upload via API?

Does OpenEMR Support CSV Upload via API?

No. OpenEMRâ€™s standard REST/FHIR APIs do not natively support direct CSV file uploads for processing patient data.

The API layer is built around standard **JSON or XML payloads** (primarily complying with HL7 FHIR standards) designed for handling individual or structured array resources (e.g., POST /v1/patient). If you send a raw CSV file to the API, it won't be able to parse it natively.

Best Known Method (BKM) for Uploading Patient Information

Because OpenEMR lacks a native "Import CSV" button in its core UI, the community relies on two primary methodologies depending on your technical constraints and whether this is a one-time migration or a recurring workflow.

Approach 1: Database-Level ETL (Best for One-Time Bulk Migration)

The fastest, most common method for a clean or initial setup is mapping your spreadsheet directly to OpenEMR's backend patient_data table using a database

management tool like **phpMyAdmin** or direct MySQL CLI.

1. **Extract the target schema***Database Prep*
   Create 2â€“3 dummy patients manually inside OpenEMR's UI with full demographic details. Open phpMyAdmin, locate the patient_data table, and export it as a CSV file with column headers. This provides your exact structural template.
2. **Transform and map your data***Data Staging*
   Open the exported template alongside your source patient list in a spreadsheet editor (like LibreOffice Calc or Excel). Map your source columns to match the OpenEMR layout precisely. Add blank columns for fields you don't use. Ensure date formats match the database requirements (YYYY-MM-DD). Crucial: Clear the id field values so MySQL auto-increments them on import, and ensure your pid and pubpid (external/public IDs) are unique integers to prevent duplicate key errors.
3. **Backup and clear dummy data***Risk Mitigation*
   Take a full backup of your existing database. If you are doing a clean slate import, empty (truncate) the dummy records from the patient_data tableâ€”but do not DROP the table.
4. **Execute the Import***Load Data*
   In phpMyAdmin's Import tab, upload your newly prepared CSV file. Ensure you specify to skip the first row if your file contains column headers.
5. **Generate system UUIDs***Post-Processing*
   OpenEMR relies on automated UUIDs for API routing. After a direct DB import, run the sql_upgrade.php script from your web browser or server command line to let OpenEMR systematically generate the missing UUID fields for your new entries.

Approach 2: Custom API Scripting (Best for Automated or Recurring Syncs)

If you need to continually upload patient CSVs from a third-party application or a legacy registration desk, the BKM is to build an intermediary script (written in Python, PHP, or JavaScript) or use a workflow automation tool (like Node-RED or n8n).

1. **The Parsing Script:** Write a script that watches a directory or accepts a file upload, parses the CSV line-by-line, and converts each row into a valid FHIR-compliant JSON object.
2. **The API Loop:** Iterate through those objects and systematically send standard POST HTTP requests directly to the OpenEMR Patient endpoint (/apis/default/api/patient or the FHIR equivalents).

âš ï¸ **Warning on Relational Tables:** While direct database imports (Approach 1) work beautifully for basic demographics in the patient_data table, remember that clinical notes, insurance details, and history reside in separate relational tables. If your CSV includes complex data beyond basic demographics, the API scripting approach (Approach 2) is significantly safer because OpenEMRâ€™s backend code automatically handles distributing data to the correct relational tables.