Manually Importing a Patient History

Open Clinical History provides a browser-based workflow for manually importing an existing patient history from a plain-text document.

The manual import page is:

```
/history_import.php
```

It is designed for importing raw, unstructured clinical history without requiring the source information to have already been converted into clinical events, SNOMED CT concepts or anatomical mappings.

The workflow has four stages:

```
1 · Upload
      |
      v
2 · Extract
      |
      v
3 · Audit & Import
      |
      v
4 · Complete
```

The original document is preserved while Open Clinical History builds a structured longitudinal clinical history from it.

Before Importing a Patient

The clinical processing environment should already have been prepared.

A normal installation sequence is:

```
Configure Open Clinical History
        |
        v
Import SNOMED CT
        |
        v
Build Image/SNOMED Database
        |
        v
Configure Gemini / LLM
        |
        v
Import Patient History
```

The manual upload itself can be performed even if the clinical-processing environment is not ready.

However, **extraction cannot start** until the required runtime data and LLM configuration are available.

Runtime Requirements

Before extraction begins, Open Clinical History verifies that the required clinical lookup data exists.

This includes:

```
snomed_health_history_lookup
snomed_health_history_term_lookup
body_layer
body_layer_lookup
body_layer_term
```

and the tables required by the patient-history processing pipeline.

It also verifies that the SNOMED and image lookup databases were built from the same SNOMED release.

If these checks fail, extraction is disabled and the page provides links to:

SNOMED Import

and:

Image/SNOMED Summary Build

This prevents LLM requests being consumed before deterministic SNOMED and anatomical mapping resources are available.

LLM Requirements

The configured LLM must also be available before extraction can start.

The current implementation uses Gemini.

The page verifies:

- LLM processing is enabled
- a Gemini API key is configured
- the configured model is available to the application
- the daily request/token budget has not prevented processing

The screen also shows current LLM usage information including:

```
model
requests today
tokens today
```

Configuration is managed under:

Admin → Configuration

Step 1: Upload

Open:

Import Patient History

or navigate directly to:

```
/history_import.php
```

The first screen asks for patient information and a history file.

Patient Record Number

Required

For example:

```
MRN-123456
```

or:

```
UR-84721
```

The record number is the primary identifier used by the import system to locate the patient.

Open Clinical History searches for:

```
patient_record_number = supplied record number

and

source_system = history_import
```

If a matching patient already exists, that patient is reused.

If no matching patient exists, a new patient is created.

Use a stable identifier

The same patient should always be imported using the same record number.

For example:

```
First import:
MRN-123456

Later import:
MRN-123456
```

Changing the identifier may create another patient instead of locating the existing one.

Important Limitation: Existing Patient Timelines

The current implementation can locate and reuse an existing patient.

However, it does **not yet perform full cross-document deduplication against events that have already been committed to that patient's longitudinal history**.

For example:

```
Document A
2019 - Myocardial infarction
        |
        v
Committed patient event



Document B
Past history: myocardial infarction in 2019
```

The extraction pipeline performs extensive duplicate and summary reconciliation **within the document being imported**, but it does not yet comprehensively compare the new document against every previously committed clinical event for that patient.

Therefore, care should currently be taken when importing overlapping histories for an existing patient.

This is particularly relevant when importing repeated copies of:

- complete medical histories
- problem lists
- discharge summaries
- specialist letters containing copied past history

Name

Optional

The patient's display name may be entered when creating a new patient.

For example:

```
Jane Example
```

The display name is stored as a patient attribute.

It is **not** used for patient matching.

If a matching patient record number already exists, entering a different name does not currently update that patient's stored details.

Date of Birth

Optional

Use the date selector provided by the browser.

Internally the value is submitted as:

```
YYYY-MM-DD
```

For example:

```
1965-11-24
```

Open Clinical History validates that:

- it is a real calendar date
- it is not in the future
- the year is not earlier than 1880

If the patient already exists, entering a different date of birth does not currently update that patient.

Sex at Birth

Available options are:

```
Female
Male
Intersex
Unknown
```

The value is also used when selecting the patient's anatomical image asset set.

Currently:

```
male   → male asset set


other values → female asset set
```

The anatomical catalogue available in the installation determines which artwork is ultimately available for visualisation.

History File

The manual importer currently accepts:

plain-text files only

Recommended extensions are:

```
.txt
.text
```

The browser file selector accepts:

```
text/plain
```

The document may contain completely unstructured clinical text.

It does **not** need to contain structured JSON, SNOMED codes or predefined event records.

For example, a source document may contain:

```
Patient: Example Patient

1998
Appendicectomy.

2007
Diagnosed with hypertension.

March 2016
Admitted following sudden onset left-sided weakness and dysarthria.
CT confirmed an acute cerebral infarction.

Current problems:
Hypertension
Residual left-sided weakness
```

Open Clinical History is responsible for interpreting the structure.

Maximum File Size

The maximum manual-upload size uses the configured:

```
api_max_upload_mb
```

setting.

The default is:

```
2 MB
```

Although the setting is named for the API, it is currently also used by the manual history importer.

The value can be changed under:

Admin → Configuration

A file larger than the configured limit is rejected before processing.

Character Encoding

Uploaded documents should preferably use:

```
UTF-8
```

If the uploaded text is not valid UTF-8, Open Clinical History currently attempts to convert it from:

```
ISO-8859-1
```

before storing the document.

What Happens to the Uploaded File?

Open Clinical History reads the uploaded text and stores its normalised contents in the database.

The document record includes information such as:

- patient
- source filename
- MIME type

- byte count
- character count
- SHA-256 document hash
- original text
- processing status

The import code does not deliberately copy the original uploaded file into a permanent web-accessible upload directory.

The clinical text itself is persisted in the `history_document` database record.

Exact Duplicate Documents

When a document is uploaded, Open Clinical History calculates a SHA-256 hash of its normalised text.

The document hash is associated with the patient.

If **exactly the same text** has already been uploaded for the same patient, Open Clinical History reuses the existing `history_document` rather than storing another copy.

For example:

```
Patient 42
   +
identical clinical text
        |
        v
existing history_document reused
```

This prevents accidental duplicate uploads of an unchanged source document.

This should not be confused with clinical-event deduplication.

A modified document produces a different document hash and is treated as a new source document.

Upload the Document

After entering the patient information and selecting the text file, click:

Upload and create patient

Despite the button wording, an existing patient with the same record number is reused rather than duplicated.

After upload, the page moves to:

```
2 · extract
```

The stored document is assigned an internal document ID.

For example:

```
document #42
```

Step 2: Extract

After uploading the document, Open Clinical History displays information including:

- document ID
- source filename
- character count
- document status
- patient record number
- LLM request count
- LLM token usage

If the required runtime services are ready, click:

Start extraction

Extraction Runs in the Background

The browser does not perform the extraction itself.

`history_import.php` launches:

```
history_job.php
```

using the server's CLI PHP installation.

Conceptually:

```
history_import.php
       |
       | start background process
       v
history_job.php
       |
       v
Clinical processing pipeline
```

The browser then polls a shared job-status file to display progress.

This means:

> It is safe to close the browser page after extraction has started.

The server-side extraction continues independently.

Manual Import and the Ingest Queue

The initial manual extraction does **not** currently use the normal `ingest_queue`.

Instead, `history_import.php` starts a dedicated background process for the document.

This differs from API imports, which normally enter the controlled ingest queue.

Conceptually:

```
Manual import

history_import.php
       |
       v
history_job.php



API import

history_api_import.php
       |
       v
ingest_queue
       |
       v
queue_worker.php
```

Only one extraction worker is allowed to process a particular document at a time.

Background Worker Requirements

Before starting extraction, Open Clinical History checks the CLI environment.

It verifies:

- a usable CLI PHP binary exists
- `pdo_mysql` is available to that PHP installation
- the PHP cURL extension is available
- `proc_open()` is available
- the shared runtime directory is writable

If one of these checks fails, the page displays:

The background worker cannot run

along with diagnostic information.

This is important because the PHP configuration used by the web server can differ from the PHP configuration used by CLI workers.

What Extraction Actually Does

The current patient-history pipeline consists of several separate processing stages.

```
Original text
      |
      v
1. Safety windows
      |
      v
```

```
2. Source segmentation
      |
      v
3. Atomic event extraction
      |
      v
4. Document-wide reconciliation
      |
      v
5. SNOMED resolution
      |
      v
6. Anatomical mapping
      |
      v
7. Clinical audit
      |
      v
Event proposals
```

The separation is deliberate.

The LLM is not simply asked to read the entire document and produce a final patient history in one operation.

---

1. Creating Safety Windows

PHP first divides the document into manageable overlapping safety windows.

The default target size is controlled by:

```
llm_max_chars_per_chunk
```

with a default of:

```
4500 characters
```

The overlap helps avoid losing context where an arbitrary processing boundary falls in the middle of a clinical section.

These windows are transport boundaries, **not clinical event boundaries**.

Maximum Windows Per Document

The number of safety windows is limited by:

```
llm_max_chunks_per_document
```

Default:

```
40
```

This setting is important.

If a document produces more windows than the configured maximum, excess windows are currently truncated.

Therefore, for unusually large patient histories, check the document status message for a truncation warning.

A document should not be considered fully processed if the source was truncated by this limit.

---

2. AI Source Segmentation

The first AI stage analyses each safety window and divides the source into coherent clinical blocks.

Examples include:

```
dated event
narrative event
```

```
procedure
investigation
allergy list
current problem list
past operation summary
copied discharge summary
family history
medication list
negative/rule-out statement
administrative text
```

The purpose is to recognise the structure of the source before trying to extract individual clinical events.

For example:

```
12 March 2018


Presented with sudden onset weakness of the left arm and dysarthria.
CT confirmed acute cerebral infarction.
```

should be understood as a coherent dated clinical section rather than as arbitrary pieces of text.

Every Source Span Is Preserved

The segmentation stage is designed to preserve source coverage.

If the AI fails to classify a piece of text, PHP still persists the uncovered portion as:

```
unclassified
```

rather than silently dropping it.

This gives later audit stages an opportunity to detect information that may not have been extracted correctly.

3. Atomic Clinical Event Extraction

A second AI stage extracts individual clinical events from the persisted source segments.

Events are deliberately atomic.

For example:

```
Left hemiparesis and dysarthria
```

may become:

```
Event 1: Left hemiparesis
Event 2: Dysarthria
```

Similarly:

```
Breast excision and sentinel node biopsy
```

may produce two independent procedure events when clinically appropriate.

The model extracts information including:

- event title
- concise clinical concept
- event type
- date information
- clinical status
- severity
- significance
- persistence
- laterality
- anatomical terms
- source evidence
- terminology search terms

At this stage the model does **not** invent SNOMED identifiers.

Non-Patient Events Are Identified

The extraction process explicitly distinguishes statements that should not become patient clinical events.

Examples include:

- negated conditions
- family history
- copied historical summaries
- administrative information
- medication lists where no independent clinical event is described

These may be retained for source understanding without becoming committed patient events.

---

4. Document-Wide Reconciliation

After individual source segments have been extracted, Open Clinical History considers the document as a whole.

This is important because clinical records frequently contain the same information in different places.

For example:

```
1987
Diagnosed with epilepsy.

...
```

```
Current problems
Epilepsy
```

The second statement should not necessarily create another diagnosis dated today.

The reconciliation stage considers:

- chronology
- repeated diagnoses
- copied past-history lists
- current problem lists
- status changes
- condition progression
- historical summaries

Its purpose is to construct a coherent set of canonical events from the entire source document.

---

5. SNOMED CT Resolution

After reconciliation, Open Clinical History attempts to resolve each clinical event against the local SNOMED database.

It uses the compact tables created by the SNOMED import:

```
snomed_health_history_lookup
snomed_health_history_term_lookup
```

Candidate concepts are generated locally.

Where an LLM is required to resolve ambiguity, it is allowed to choose only from the supplied local candidates.

Conceptually:

```
Clinical event
      |
      v
Local SNOMED search
      |
      v
```

```
Candidate concepts
      |
      v
Optional bounded AI choice
      |
      v
Verified local SNOMED concept
```

The model cannot simply invent a SNOMED concept identifier.

---

6. Anatomical Image Mapping

Once terminology has been resolved, the pipeline attempts to connect the clinical event to the installed anatomical image catalogue.

The principal runtime resources are:

```
body_layer_lookup
body_layer_term
```

The system considers:

- the resolved SNOMED concept
- anatomical site information
- laterality
- installed image candidates
- mapping safety rules

The result may be zero, one or several suitable image layers.

A valid clinical event does not necessarily require an anatomical image.

For example, some systemic or non-local conditions may have no useful body-layer representation.

---

7. Final Clinical Audit

Before the document becomes available for import, Open Clinical History performs several read-only audit stages.

These assess:

Source coverage

Did the extraction capture genuine clinical events present in each source segment?

Event fidelity

Are dates, laterality, status and clinical claims supported by the source?

Duplicate reconciliation

Are events that appear to describe the same clinical episode still duplicated?

SNOMED mapping

Were clinical events successfully resolved to appropriate terminology?

Timeline consistency

Do the events make sense when viewed across the complete chronology?

Body-layer suitability

Are the selected anatomical images appropriate to the clinical event?

Import Quality Score

The final audit produces an overall quality score from:

```
0 to 100
```

and a grade:

```
A
B
C
D
E
```

The score combines:

Component	Weight
Source coverage	30%
Event fidelity	25%
Duplicate reconciliation	15%
SNOMED mapping	15%
Date/status/laterality consistency	10%
Body-layer suitability	5%

The import screen also reports information such as:

- critical blockers
- warnings
- likely duplicates
- uncovered source events
- unresolved SNOMED events

The quality score is a processing-quality indicator.

It should not be interpreted as a measured percentage of clinical accuracy.

Live Extraction Progress

While the background worker is running, the page displays live progress.

Depending on the current phase, you may see activity such as:

```
Creating safety windows
AI source segmentation
AI atomic event extraction
Document-wide reconciliation
SNOMED candidate retrieval
SNOMED concept resolution
Body-layer mapping
Source coverage audit
Global consistency audit
```

The display also reports information such as:

- source windows
- source segments
- extracted events
- canonical events
- requests
- tokens
- individual section status

Extraction Can Take Time

Large clinical histories can require multiple LLM requests and significant local terminology processing.

Processing time depends on:

- document size
- number of source sections
- number of extracted events
- SNOMED ambiguity
- terminology repair requirements
- audit complexity
- LLM response latency

A large longitudinal patient history should not be expected to complete instantly.

Extraction Is Resumable

The pipeline persists its work as it progresses.

If the worker stops unexpectedly, the document may display:

The document says it is processing, but no active worker was found.

The page then provides:

Resume unfinished sections

Completed work is retained.

The resumed worker continues from the first unfinished processing stage rather than deliberately starting the complete AI extraction again.

This can preserve already completed LLM responses and reduce unnecessary model usage.

Step 3: Audit & Import

Once extraction and auditing have completed, the document enters:

```
proposed
```

status.

The screen displays the extracted proposals for review.

Each proposal can include information such as:

- event title
- date
- event type
- laterality
- severity
- significance
- clinical status
- persistence
- source type

- SNOMED concept
- terminology matching method
- anatomical layers
- mapping warnings
- audit status
- audit findings
- source excerpt

Source Evidence

Each extracted event retains a source excerpt.

For example:

```
"CT brain demonstrated an acute left middle cerebral artery infarction."
```

This allows the proposed structured event to be compared directly with the original evidence that produced it.

Source evidence is an important part of the audit trail.

SNOMED Matching Information

Where a SNOMED concept was resolved, the proposal shows:

- selected SNOMED term
- concept identifier
- matching method

Possible matching methods include:

```
exact SNOMED FSN
exact SNOMED synonym
```

```
grounded SNOMED selection
SNOMED prefix suggestion
no SNOMED match
```

The terminology result is generated from the local SNOMED catalogue.

Mapping Warnings

A proposal can also contain anatomical mapping warnings.

Examples may include:

- laterality conflicts
- inappropriate anatomical imagery
- insufficient anatomical evidence
- ambiguous body mapping

The current automatic-import stage takes these warnings seriously.

A proposal with mapping warnings is not automatically committed.

Run Auto-Import

When the proposals are ready, click:

Run Auto-Import

This does **not** blindly commit every extracted proposal.

Each proposed event passes a safety gate.

An event is automatically committed only when:

```
audit status = pass

AND
```

```
a SNOMED concept has been resolved

AND

the statement is not negated

AND

the statement is not family history

AND

the statement is not merely a copied historical summary

AND

there are no anatomical mapping warnings
```

Conceptually:

```
Extracted proposal
       |
       v
Audit passed?
       |
       +--- no ---> Learning queue
       |
      yes
       |
       v
SNOMED resolved?
       |
       +--- no ---> Learning queue
       |
      yes
       |
       v
Mapping safe?
       |
       +--- no ---> Learning queue
```

```
        |
       yes
        |
        v
Commit to patient history
```

Committed Events

Safe events are inserted into:

```
clinical_event
```

The committed record preserves information including:

- patient
- date and date precision
- title
- summary
- clinical status
- event type
- laterality
- severity
- significance
- terminology mapping
- source document provenance
- source evidence
- reviewer/import identity

Associated visual mappings are written to:

```
clinical_event_site
```

where suitable anatomical layers exist.

Events That Cannot Be Automatically Imported

A proposal that does not pass the automatic-import safety gate is marked:

```
rejected
```

from automatic import and routed to:

```
history_unmatched_queue
```

This does **not necessarily mean the clinical statement itself is wrong**.

It means the system did not consider the proposal safe enough for unattended commitment.

The item is categorised according to the problem.

SNOMED Concept Problems

An unresolved terminology item is classified as:

```
snomed_concept
```

For example:

```
Clinical event understood
        |
        v
No sufficiently reliable SNOMED concept
        |
        v
Learning queue
```

Body-Layer Problems

If SNOMED terminology was found but the anatomical mapping raised warnings, the item is classified as:

```
body_layer
```

Specificity Problems

If terminology exists but another ambiguity prevents safe commitment, the item may be classified as:

```
specificity
```

The Learning Queue

The learning queue provides a controlled mechanism for resolving terminology and mapping gaps.

This is deliberately separated from the original extraction.

The system can therefore improve its terminology knowledge without repeatedly asking the LLM to reinterpret the original patient document.

Apply Learned Mappings

If terminology for previously unresolved events is later resolved through the governed learning process, the import page can display:

Apply learned mappings

This operation:

- uses the newly resolved local terminology
- remaps the event against the current anatomy catalogue
- rechecks the original audit gate
- commits or refreshes eligible events
- does not repeat the original source extraction

The application stage itself does **not require another LLM extraction call**.

It is queued through the normal worker pool.

Governed Learning State

Where applicable, the page displays the learning state for the document.

This can include counts for:

```
Learning queue pending
Resolved terminology awaiting application
Queued
Running
Applied
Held for review
Failed
```

This allows unresolved clinical terminology to be managed independently from the original patient import.

Full Re-run Mapping

The screen also provides:

Full re-run mapping

This is an advanced repair operation.

It should **not** be used as the normal way to apply newly learned terminology.

A full mapping re-run:

1. removes proposed and rejected mappings for the document
2. resets mapping status
3. clears the document's unmatched queue entries
4. starts the processing worker again
5. rebuilds terminology and anatomical mapping

The previously persisted AI source extraction is retained, so the intent is to avoid repeating the original text extraction.

Use this when the mapping itself needs to be rebuilt, for example after a significant mapping-system correction.

For ordinary terminology learning, use:

Apply learned mappings

instead.

Step 4: Complete

After automatic import finishes, the document status becomes:

```
committed
```

The screen reports:

Import complete

and provides:

View Patient History

The committed clinical events can now be viewed as part of the patient's longitudinal history.

What `committed` Means

A committed **document** does not mean that every extracted proposal was committed.

For example:

```
24 proposals extracted


20 safe events committed


4 events routed to learning queue
```

The document can still be considered processed and committed.

The unmatched items remain available for governed learning and later application.

Patient History Output

Committed events become part of the patient's longitudinal history.

Conceptually:

```
Raw clinical document
        |
        v
Source segmentation
        |
        v
Atomic clinical events
        |
        v
Chronological reconciliation
        |
        v
SNOMED CT classification
        |
```

```
        v
Anatomical mapping
        |
        v
Clinical audit
        |
        v
Safe committed events
        |
        v
Longitudinal patient history
```

This is the central purpose of the manual history importer.

Recommended Import Workflow

For a new patient:

```
1. Open history_import.php
        |
        v
2. Enter stable record number
        |
        v
3. Enter optional demographics
        |
        v
4. Select plain-text history
        |
        v
5. Upload
        |
        v
6. Verify runtime checks
```

```
        |
        v
7. Start extraction
        |
        v
8. Allow background processing to finish
        |
        v
9. Review audit score and proposals
        |
        v
10. Run Auto-Import
        |
        v
11. Review learning-queue items if present
        |
        v
12. View Patient History
```

Troubleshooting

Start Extraction Is Disabled

Check the messages above the extraction controls.

Common causes include:

- Gemini is not configured
- LLM processing is disabled
- SNOMED import has not completed
- image/SNOMED database has not been built
- lookup tables are empty
- SNOMED and image databases were built from different releases
- the CLI worker environment is unavailable

Runtime Data Is Not Ready

Open Clinical History deliberately prevents extraction when deterministic terminology or image data is missing.

Run:

SNOMED Import

followed by:

Image/SNOMED Summary Build

and reload the patient import.

LLM Is Not Configured

Open:

Admin → Configuration

and verify:

- Enable LLM processing
- Gemini API key
- Gemini model
- request/token limits

A document may remain uploaded until the LLM configuration is corrected.

Background Worker Cannot Run

Check the diagnostics displayed on the page.

The CLI PHP environment requires:

```
PHP
pdo_mysql
curl
proc_open
writable shared run directory
```

The web PHP installation working correctly does not guarantee the CLI PHP installation has the same extensions.

Document Is Stuck on Processing

If the document reports `processing` but no active worker can be located, the page should offer:

Resume unfinished sections

Use this rather than uploading the source again.

Extraction Completed With No Events

The page explicitly warns when extraction finishes without producing proposals.

Do not treat this as a successful empty clinical history without investigating.

Review:

- the source document
- extraction progress
- worker log
- LLM configuration

A Valid Event Was Not Imported

Check the proposal's:

- audit status
- SNOMED match
- mapping warnings
- negation/family-history status
- learning-queue state

A clinically meaningful event may have been deliberately held because Open Clinical History could not establish a sufficiently safe structured representation.

Same File Was Uploaded Again

If the source text is identical and the same patient record number was used, Open Clinical History may return to the existing document because exact document-content deduplication is already implemented.

This is expected behaviour.

Current Limitations

The manual importer currently has several deliberate limitations.

Plain text only

The upload screen currently accepts plain-text clinical histories.

PDF, Word, image and scanned-document ingestion are not handled directly by this screen.

Source material must first be converted to text.

Existing patient demographics are not updated

A matching record number reuses the existing patient.

Name, date of birth and sex supplied during the later import do not currently update that patient.

Cross-document timeline deduplication is not yet complete

The pipeline performs sophisticated reconciliation within the source document being processed.

It does not yet fully reconcile a newly imported document against every clinical event already committed to the patient's existing longitudinal timeline.

This should be considered when importing overlapping source histories.

Related Components

Component	Purpose
`history_import.php`	Manual patient import and review interface
`history_job.php`	Background extraction worker
`lib/history.php`	Core patient-history ingestion and mapping services
`HistorySegmentationService`	Converts safety windows into coherent source segments
`HistoryExtractionService`	Extracts atomic clinical events
`HistoryReconciliationService`	Reconciles the complete source chronology
`ResolveTerminologyAndBodyStep`	Resolves SNOMED terminology and anatomical layers
`HistoryAuditService`	Performs final source and consistency audits
`HistoryLearningApply`	Applies governed terminology learning after extraction

Summary

The manual patient-history importer is designed to turn:

```
Raw unstructured clinical text
```

into:

```
A source-backed,
SNOMED-classified,
anatomically mapped,
audited longitudinal clinical history
```

without requiring the person performing the import to manually structure the source record first.

The process deliberately separates:

```
Source storage
    |
Interpretation
    |
Terminology resolution
    |
Anatomical mapping
    |
Clinical audit
    |
Commitment
```

so that an LLM response is never treated as the final patient record without additional deterministic terminology, mapping and audit controls.

Revision #1
Created 2026-08-25 09:59:02 UTC by Admin
Updated 2026-08-25 09:59:58 UTC by Admin