Configuration

`admin_config.php` provides the central runtime configuration interface for Open Clinical History.

Application settings are stored in the `app_config` database table and accessed through `lib/app_config.php`. These settings are separate from deployment-level configuration such as database connection details.

The page also manages API bearer tokens through the `api_token` table.

Configuration architecture

Runtime configuration follows this general flow:

```
Admin → Configuration
        |
        v
admin_config.php
        |
        v
AppConfig::set(...)
        |
        v
app_config database table
        |
        +------------------------+
        |                        |
        v                        v
AppConfig::get()          AppConfig::llmLimits()
        |                        |
        v                        v
Application code          Clinical processing pipeline
```

The `app_config` table is the source of truth for application runtime settings.

Values are stored with:

- configuration key

- configuration value
- value type
- user that last changed the value
- update timestamp

Changes made through the Configuration page take effect without modifying PHP configuration files.

Dates

Date format

Configuration key

```
date_format
```

Default

```
dd/mm/yyyy
```

Allowed values

Value	Example
`dd/mm/yyyy`	`25/08/2026`
`mm/dd/yyyy`	`08/25/2026`

Controls how dates are displayed throughout the user interface.

It also determines how non-ISO dates submitted through the API are interpreted.

ISO dates using:

```
YYYY-MM-DD
```

are always accepted regardless of this setting.

Machine integrations should therefore use ISO dates wherever possible.

Used by

- UI date rendering
- API date parsing
- API date responses
- token expiry display
- clinical history date presentation

SNOMED CT RF2

RF2 release path

Configuration key

```
snomed_release_path
```

Default

Blank.

Example

```
/var/www/private/snomed/SnomedCT_Release_AU1000036_20260731
```

Specifies the server location of the SNOMED CT RF2 distribution used by the SNOMED import process.

The value may point to:

- the root of an RF2 release; or
- the release's `Snapshot` directory.

The path must be an **absolute server path** beginning with `/`.

Trailing `/` characters are removed when the setting is saved.

If the setting is blank, the SNOMED subsystem falls back to its built-in private-folder discovery mechanism.

Temporary override

`snomed_import.php` can temporarily override the configured directory using:

```
?dir=/full/path/to/release
```

The configured value remains unchanged.

Used by

- `snomed_import.php`
- `snomed_job.php`

Gemini / LLM

Enable LLM processing

Configuration key

```
llm_enabled
```

Default

Enabled.

Controls whether Open Clinical History is allowed to perform LLM processing.

When disabled, `LlmFactory` refuses to create an LLM provider and model-dependent extraction fails with a configuration error.

Disabling this setting therefore disables functionality including model-assisted clinical extraction, terminology processing and other AI-dependent pipeline stages.

LLM provider

Configuration key

```
llm_provider
```

Current value

```
gemini
```

The current administration page does not provide a provider selector.

Whenever configuration is saved, `admin_config.php` explicitly sets:

```
llm_provider = gemini
```

The underlying LLM factory is designed to support provider selection, but the current implementation only recognises Gemini.

Gemini API key

Configuration key

```
llm_gemini_api_key
```

Default

Blank.

Stores the Gemini API credential used by the LLM provider.

For security, the existing key is never rendered back into the configuration page.

If the API key field is left blank when saving the page, the existing key is retained.

The **Clear the stored Gemini API key** option explicitly replaces the stored value with an empty string.

Security

Unlike Open Clinical History API bearer tokens, the Gemini API key must remain retrievable by the application and is therefore stored as a configuration value in `app_config`.

Database access to `app_config` should consequently be treated as secret-bearing access.

Gemini model

Configuration key

```
llm_gemini_model
```

Default

```
gemini-3.5-flash-lite
```

Determines which Gemini model is used for LLM operations.

The configured value may contain:

```
A-Z
a-z
0-9
.
_
-
```

The configuration page validates the syntax of the model name but does not verify with Gemini that the model actually exists.

Provider limits

Gemini timeout

Configuration key

```
llm_gemini_timeout_seconds
```

Default

```
120 seconds
```

Allowed range

```
10 to 600 seconds
```

Sets the maximum HTTP request time allowed for a Gemini request.

A request exceeding this period is treated as a model/API failure.

Daily request cap

Configuration key

```
llm_daily_request_cap
```

Default

```
4000
```

Controls the maximum number of Gemini requests permitted per day.

A value of:

```
0
```

means unlimited.

Before making a model request, the LLM budget manager checks current daily usage. Once the configured request count has been reached, further calls are rejected before Gemini is contacted.

This provides a hard cost and usage protection mechanism.

Daily token cap

Configuration key

```
llm_daily_token_cap
```

Default

```
4,000,000
```

Controls the total number of recorded prompt and output tokens permitted per day.

A value of:

```
0
```

means unlimited.

The budget check uses:

```
prompt tokens + output tokens
```

from recorded daily usage.

Once the cap is reached, additional LLM requests are blocked.

Maximum output tokens per request

Configuration key

```
llm_max_output_tokens
```

Default

```
4096
```

Configuration range

```
256 to 131072
```

Sets the requested maximum response size sent to the Gemini provider.

Larger values allow the model to produce larger structured responses but can increase:

- latency
- token consumption
- cost
- JSON processing requirements

The actual maximum may also be constrained by the capabilities of the selected Gemini model.

Maximum prompt characters per request

Configuration key

```
llm_max_chars_per_request
```

Default

```
24000
```

Allowed range

```
8000 to 1,000,000 characters
```

Provides a hard application-level limit on the total size of the system and user prompt sent in a single LLM request.

The budget manager calculates:

```
strlen(system prompt) + strlen(user prompt)
```

before contacting Gemini.

Requests exceeding the configured limit are rejected locally.

This setting also influences the safe batch size used by SNOMED grounding and repair.

Document extraction

Maximum characters per legacy chunk

Configuration key

```
llm_max_chars_per_chunk
```

Default

```
4500
```

Allowed range

```
500 to 100,000
```

Controls the target maximum size of chunks created by the legacy document chunking pipeline.

Large source documents are divided into sections before further processing.

Increasing this value creates fewer, larger chunks.

Decreasing it creates more, smaller chunks.

Maximum chunks per document

Configuration key

```
llm_max_chunks_per_document
```

Default

```
40
```

Allowed range

```
1 to 500
```

Limits the number of legacy chunks generated for a document.

If a document would generate more chunks than this limit, the excess chunks are truncated and are not processed.

This is therefore a **processing completeness limit**, not merely a performance setting.

> A very low value can result in part of a long clinical record not being processed.

Maximum characters per source segment

Configuration key

```
llm_max_chars_per_source_segment
```

Default

```
2800
```

Admin UI range

```
500 to 100,000
```

Effective runtime range

```
1200 to 5000
```

Controls the size of source segments produced by the current segmentation pipeline.

Oversized source sections are divided into smaller pieces before extraction.

The segmentation service currently clamps the value internally:

```
minimum 1200
maximum 5000
```

Consequently, setting a value such as `10000` in the administration page does **not** produce 10,000-character source segments. The runtime uses 5,000.

Maximum characters per extraction batch

Configuration key

```
llm_max_chars_per_extraction_batch
```

Default

```
6500
```

Admin UI range

```
1000 to 1,000,000
```

Effective minimum

```
1500
```

Controls how much source text can be grouped into one clinical extraction request.

Extraction batches are built until either:

- the character limit is reached; or
- the segment-count limit is reached.

Whichever limit is reached first closes the batch.

Maximum segments per extraction batch

Configuration key

```
llm_max_segments_per_extraction_batch
```

Default

```
6
```

Admin UI range

```
1 to 100
```

Effective runtime range

```
1 to 10
```

Controls how many source segments may be sent to the extraction model in one request.

The extraction service currently imposes a hard maximum of 10 regardless of the higher value allowed by the administration page.

Retry behaviour

Transient retries

Configuration key

```
llm_transient_retries
```

Default

```
1
```

Admin UI range

```
0 to 10
```

Effective runtime range

```
0 to 3
```

Determines how many times a transient Gemini/API error can be retried.

Only errors classified as transient are retried.

A value of `0` disables application-level retries.

The current pipeline caps retries at three even if a higher value is configured.

Retry delay

Configuration key

```
llm_transient_retry_delay_seconds
```

Default

```
4 seconds
```

Admin UI range

```
0 to 300 seconds
```

Effective runtime range

```
1 to 30 seconds
```

Sets the base delay between transient retries.

The retry code applies an increasing delay based on the retry attempt:

```
delay × attempt number
```

With the default value of four seconds:

```
Retry 1: 4 seconds
Retry 2: 8 seconds
Retry 3: 12 seconds
```

Although the UI accepts zero, the runtime enforces a minimum of one second.

Advanced SNOMED grounding and repair

These settings control the terminology-resolution pipeline after clinical events have been extracted.

The normal flow is conceptually:

```
Extract clinical event
        |
        v
Generate local SNOMED candidates
        |
        v
Grounded LLM resolution
        |
        +---- confident result ----> accept
        |
        +---- unresolved/suspicious
                    |
                    v
                critic
```

```
                |
                v
        optional repair
                |
                v
        re-run grounding
```

Enable AI-assisted SNOMED repair

Configuration key

```
llm_enable_snomed_ai_repair
```

Default

Disabled.

Allows the pipeline to use model-assisted repair when normal terminology grounding cannot confidently resolve an event.

When disabled, unresolved mappings do not enter the AI repair pathway.

This is separate from normal grounded terminology resolution.

Grounded batch characters

Configuration key

```
llm_grounded_batch_chars
```

Default

```
18000
```

Admin UI range

```
1000 to 1,000,000
```

Effective minimum

```
6000
```

Controls the target maximum size of a grounded SNOMED resolution batch.

The actual batch budget is also constrained by `llm_max_chars_per_request`.

The pipeline reserves approximately 4,500 characters for prompt overhead:

```
effective budget =
min(
    grounded_batch_chars,
    max(6000, max_chars_per_request - 4500)
)
```

With the defaults:

```
max_chars_per_request = 24000
grounded_batch_chars  = 18000

effective budget = 18000
```

Candidate limit

Configuration key

```
llm_grounded_candidate_limit
```

Default

```
12
```

Admin UI range

```
1 to 100
```

Effective runtime range

```
4 to 20
```

Controls the maximum number of locally generated SNOMED candidates supplied for each event during grounded terminology resolution.

More candidates can improve recall but also:

- enlarge prompts
- increase ambiguity
- consume more tokens
- make model selection more difficult

The runtime currently limits the value to 20.

Search terms per event

Configuration key

```
llm_grounded_search_terms_per_event
```

Default

```
5
```

Admin UI range

```
1 to 20
```

Effective runtime range

```
1 to 8
```

Controls how many clinical search expressions may be used to locate local SNOMED candidates for an event.

The runtime currently caps this value at eight.

Resolution minimum confidence

Configuration key

```
llm_grounded_resolution_min_confidence
```

Default

```
0.72
```

Allowed range

```
0.00 to 1.00
```

Specifies the minimum confidence required before a grounded model-selected SNOMED candidate is accepted.

Higher values make terminology resolution more conservative.

Lower values permit weaker matches to be accepted.

For clinical terminology, this setting should be changed cautiously because it directly influences whether a proposed terminology match is accepted.

SNOMED repair

Repair rounds

Configuration key

```
llm_grounded_repair_rounds
```

Default

```
2
```

Admin UI range

```
0 to 10
```

Effective runtime range

```
0 to 3
```

Sets the number of terminology repair cycles permitted for unresolved or suspicious matches.

A value of zero disables repair rounds even if the AI repair pathway is otherwise enabled.

The runtime currently permits a maximum of three rounds.

Repair candidate limit

Configuration key

```
llm_grounded_repair_candidate_limit
```

Default

```
15
```

Admin UI range

```
1 to 100
```

Effective runtime range

```
4 to 20
```

Controls how many SNOMED candidates may be considered when an event is being re-evaluated during repair.

Repair deliberately uses a separate limit from first-pass terminology grounding.

Repair search terms per event

Configuration key

```
llm_grounded_repair_search_terms_per_event
```

Default

```
6
```

Admin UI range

```
1 to 20
```

Effective runtime range

```
1 to 8
```

Controls how many revised search expressions may be used when attempting to resolve an event during a repair round.

Repair maximum terms

Configuration key

```
llm_grounded_repair_max_terms
```

Default

```
6
```

Admin UI range

```
1 to 20
```

Effective runtime range

```
2 to 8
```

Limits the final collection of SNOMED search terms assembled for a repaired event.

The terms may originate from:

- repair-plan search terms
- existing SNOMED search terms
- condition text
- event title

Duplicates are normalised before the limit is applied.

Repair batch characters

Configuration key

```
llm_grounded_repair_batch_chars
```

Default

```
15000
```

Admin UI range

```
1000 to 1,000,000
```

Effective minimum

```
6000
```

Controls the target size of LLM batches used to plan terminology repair.

The actual limit is constrained by the global prompt limit:

```
effective budget =
min(
    grounded_repair_batch_chars,
    max(6000, max_chars_per_request - 5000)
)
```

The reserved space accommodates system instructions and additional repair context.

Grounded critic

Enable grounded critic pass

Configuration key

```
llm_grounded_critic_enabled
```

Default

Enabled.

Enables an additional quality-control pass for grounded SNOMED resolutions.

The critic can flag a mapping for repair when:

- the selected match is below the configured confidence threshold
- the resolver's explanation contains uncertainty language
- terminology has very weak lexical similarity to the source event

This provides an additional safeguard against plausible but poorly grounded terminology selections.

Critic minimum confidence

Configuration key

```
llm_grounded_critic_min_confidence
```

Default

```
0.85
```

Allowed range

```
0.00 to 1.00
```

Defines the confidence level below which a grounded AI terminology match is considered worthy of critic review.

This differs from the **resolution minimum confidence**.

Conceptually:

- `llm_grounded_resolution_min_confidence` determines whether a proposed match can be accepted.
- `llm_grounded_critic_min_confidence` determines whether an accepted grounded match should receive additional scrutiny.

Anatomy and body-layer selection

Body selection batch size

Configuration key

```
llm_body_selection_batch_size
```

Default

```
8
```

Admin UI range

```
1 to 100
```

Effective runtime range

```
1 to 16
```

Controls how many clinical events requiring AI-assisted anatomical layer selection are processed in a single model batch.

The body-layer model receives:

- the clinical event
- extracted anatomical information
- the selected SNOMED concept
- bounded installed image-layer candidates

The runtime currently caps the batch size at 16 events.

Body candidate limit

Configuration key

```
llm_body_candidate_limit
```

Default

```
8
```

Admin UI range

```
1 to 100
```

Effective runtime range

```
3 to 15
```

Controls how many candidate anatomical image layers may be presented to the body-layer selection model.

Candidate layers are ranked before this limit is applied.

The model may select zero to four layers from those supplied candidates.

The runtime currently guarantees at least three candidates where available and caps the configured limit at 15.

Import API

Enable the import API

Configuration key

```
api_enabled
```

Default

Enabled.

Controls access to:

```
history_api_import.php
```

When disabled, the API returns:

```
HTTP 503
```

to callers before processing an import.

This can be used as an application-level emergency stop for external ingestion.

Require HTTPS

Configuration key

```
api_require_https
```

Default

Enabled.

Rejects API requests that do not arrive over HTTPS.

The API recognises TLS using:

- the PHP `HTTPS` server variable
- `X-Forwarded-Proto: https`
- server port `443`

Requests received without HTTPS while this setting is enabled return an error.

This setting should remain enabled for any environment carrying real clinical records.

Auto-commit by default

Configuration key

```
api_default_auto_commit
```

Default

Enabled.

Controls what happens when an API caller does not explicitly supply an `auto_commit` value.

When enabled, an imported document automatically proceeds to the audit and auto-commit stage after extraction.

A caller may explicitly override the default.

If auto-commit is requested, the caller's API token must have the:

```
commit
```

scope.

Maximum document size

Configuration key

```
api_max_upload_mb
```

Default

```
2 MB
```

Allowed range

```
1 to 64 MB
```

Sets the maximum size of a document accepted by the import API.

Uploads larger than the configured value are rejected with:

```
HTTP 413
```

This value is also exposed to the LLM limits structure as the application upload limit.

Ingest queue

Queue settings are shown only when the `ingest_queue` table exists.

When the queue is enabled, API imports are inserted into the queue instead of immediately spawning an independent worker for every request.

The queue provides:

- bounded worker processing
- retries
- job claiming
- failure tracking
- leases
- recovery of abandoned jobs

`queue_worker.php` must be running or scheduled for queued documents to be processed.

Route submissions through the queue

Configuration key

```
queue_enabled
```

Default

Enabled.

When enabled, new API imports are submitted to `ingest_queue`.

When disabled, API submission falls back to spawning an individual worker for each request.

Queue workers also check this setting before claiming work. If disabled, a daemon worker remains alive but does not claim queued documents.

Attempts before parking as failed

Configuration key

```
queue_max_attempts
```

Default

```
3
```

Allowed range

```
1 to 10
```

Controls how many processing attempts a newly queued item receives before being permanently marked:

```
failed
```

Each claim increments the item's attempt counter.

Failures that still have attempts remaining are automatically returned to the queue.

Retry delay uses quadratic backoff:

```
attempt 1:  60 seconds
attempt 2: 240 seconds
attempt 3: 540 seconds
...
```

with a maximum delay of one hour.

The configured value is copied into each queue item's `max_attempts` field when it is created.

Changing the configuration therefore primarily affects subsequently queued work.

Extraction timeout

Configuration key

```
queue_job_timeout
```

Default

```
2700 seconds
```

or:

```
45 minutes
```

Allowed range

```
60 to 21600 seconds
```

Defines the maximum processing time allowed for an extraction job managed by the queue worker.

A job running beyond this period is considered timed out and enters the queue failure/retry process.

Claim lease

Configuration key

```
queue_lease_seconds
```

Default

```
3600 seconds
```

or:

```
60 minutes
```

Allowed range

```
60 to 21600 seconds
```

Controls how long a queue item remains claimed by a worker.

When a worker claims an item, a lease expiry timestamp is stored.

If a worker crashes or disappears without completing the item, an expired lease allows another worker to return the item to the queue.

The lease should normally be **comfortably longer than the extraction timeout**.

With the defaults:

```
Job timeout: 2700 seconds / 45 minutes
Lease:       3600 seconds / 60 minutes
```

This gives a 15-minute safety margin.

API token management

`admin_config.php` also provides API bearer-token management.

These values are stored in `api_token`, not `app_config`.

Token format

New tokens have the form:

```
och_<48 hexadecimal characters>
```

For example:

```
och_0123456789abcdef...
```

The complete plaintext token is displayed only once immediately after creation.

Only its SHA-256 hash is stored in the database.

The first 12 characters are stored separately as a token prefix so administrators can identify tokens without exposing the credential.

Token label

Maximum length

```
120 characters
```

A human-readable name identifying the system using the token.

Recommended naming convention:

```
PMS nightly feed
Hospital integration
Test import client
```

Tokens should be named for the calling system rather than an individual user.

Token expiry

The administration screen accepts an expiry period in days.

```
0 = does not expire
```

The UI permits values between:

```
0 and 3650 days
```

An expired token can no longer authenticate API requests.

Token scopes

Three scopes are currently supported.

`import`

Allows the caller to submit documents for clinical extraction.

`status`

Allows the caller to retrieve job, document and processing status.

`commit`

Allows the caller to initiate the audit/auto-commit process.

A token should ideally receive only the scopes required by its integration.

Revoking a token

Revoking a token:

- immediately prevents further authentication
- retains the database record
- retains the audit history

- records who revoked it and when

Revocation should normally be preferred when retiring a credential.

Deleting a token

Deleting permanently removes the token row from the database.

This also removes its direct token audit record.

Use deletion primarily for incorrectly created or unnecessary credentials. Use revocation when the history of the credential should be retained.

Security considerations

Gemini credential

The Gemini API key is stored in `app_config` because the application must recover the plaintext credential to call Gemini.

Access to the database and configuration administration interface should therefore be appropriately restricted.

Open Clinical History API tokens

OCH API tokens use a different security model.

The plaintext token is:

1. generated using cryptographically secure random bytes
2. displayed once
3. hashed using SHA-256
4. stored only as a hash

The original token cannot subsequently be recovered from the application database.

HTTPS

`api_require_https` should remain enabled outside explicitly isolated development environments.

Clinical documents, authentication credentials and patient metadata must not be transmitted over an unencrypted connection.

Effective runtime limits

Several advanced settings currently have different administration-page ranges and runtime ranges.

Setting	Admin range	Effective runtime
Source segment characters	500–100,000	**1,200–5,000**
Extraction segments per batch	1–100	**1–10**
Transient retries	0–10	**0–3**
Retry delay	0–300 sec	**1–30 sec**
Grounded candidate limit	1–100	**4–20**
Grounded search terms/event	1–20	**1–8**
Repair rounds	0–10	**0–3**
Repair candidate limit	1–100	**4–20**
Repair search terms/event	1–20	**1–8**
Repair maximum terms	1–20	**2–8**
Body selection batch size	1–100	**1–16**
Body candidate limit	1–100	**3–15**

`Grounded batch characters` and `Repair batch characters` are additionally constrained dynamically by `llm_max_chars_per_request`.

Current implementation notes

Provider is currently fixed to Gemini

Although `llm_provider` exists as a configuration setting, `admin_config.php` currently saves:

```
gemini
```

every time settings are saved.

The UI does not currently support selecting another provider.

Empty token scope selection

There is a current implementation inconsistency in token creation.

The token store correctly rejects a token with no scopes, but `admin_config.php` currently converts an empty scope selection into:

```
ApiTokenStore::ALL_SCOPES
```

before calling the token store.

As a result, manually unticking **all three scopes currently creates a token with all scopes**, rather than rejecting the request.

This should be corrected because it violates least-privilege expectations.

Some advanced UI ranges do not match runtime behaviour

The administration form permits values that are later clamped by the processing pipeline.

The UI should eventually be changed to reflect the actual effective limits documented above.

Legacy configuration message

The legacy document chunker still contains an error message instructing the administrator to:

```
Raise the limit in config.php
```

The setting has moved to `app_config` and should instead direct the administrator to:

```
Admin → Configuration
```

Related components

Component	Responsibility
`admin_config.php`	Configuration administration UI
`lib/app_config.php`	Configuration persistence, defaults and access
`lib/llm.php`	Gemini provider and LLM budget enforcement
`lib/history.php`	SNOMED grounding, repair and body-layer processing
`lib/history/ingest/HistorySegmentationService.php`	Source segmentation
`lib/history/extract/HistoryExtractionService.php`	Extraction batching
`lib/history/support/HistoryPipelineAiService.php`	LLM retry handling
`lib/api_token.php`	API token creation, validation and revocation
`lib/ingest_queue.php`	Ingest work queue
`queue_worker.php`	Queue execution
`history_api_import.php`	External import API

`snomed_import.php`	SNOMED import interface
`snomed_job.php`	SNOMED import worker

Configuration defaults summary

Configuration key	Default
`date_format`	`dd/mm/yyyy`
`snomed_release_path`	blank / automatic
`llm_enabled`	enabled
`llm_provider`	`gemini`
`llm_gemini_api_key`	blank
`llm_gemini_model`	`gemini-3.5-flash-lite`
`llm_gemini_timeout_seconds`	`120`
`llm_daily_request_cap`	`4000`
`llm_daily_token_cap`	`4000000`
`llm_max_output_tokens`	`4096`
`llm_max_chars_per_request`	`24000`
`llm_max_chars_per_chunk`	`4500`
`llm_max_chunks_per_document`	`40`
`llm_max_chars_per_source_segment`	`2800`
`llm_max_chars_per_extraction_batch`	`6500`
`llm_max_segments_per_extraction_batch`	`6`
`llm_transient_retries`	`1`
`llm_transient_retry_delay_seconds`	`4`
`llm_enable_snomed_ai_repair`	disabled
`llm_grounded_batch_chars`	`18000`
`llm_grounded_candidate_limit`	`12`
`llm_grounded_search_terms_per_event`	`5`
`llm_grounded_repair_rounds`	`2`
`llm_grounded_repair_candidate_limit`	`15`

`llm_grounded_repair_search_terms_per_event`	`6`
`llm_grounded_critic_enabled`	enabled
`llm_grounded_critic_min_confidence`	`0.85`
`llm_grounded_repair_max_terms`	`6`
`llm_grounded_repair_batch_chars`	`15000`
`llm_grounded_resolution_min_confidence`	`0.72`
`llm_body_selection_batch_size`	`8`
`llm_body_candidate_limit`	`8`
`api_enabled`	enabled
`api_require_https`	enabled
`api_default_auto_commit`	enabled
`api_max_upload_mb`	`2`
`queue_enabled`	enabled
`queue_max_attempts`	`3`
`queue_job_timeout`	`2700` seconds
`queue_lease_seconds`	`3600` seconds

Revision #1
Created 2026-08-25 09:25:16 UTC by Admin
Updated 2026-08-25 09:25:35 UTC by Admin