

AMA 47 | 記憶処理スクリプト設計（構造記憶の保存・整形・同期）

目的

このCanvasでは、構造記憶（diary-log / memory-log）をJSON構造に整形・保存し、LangChainベースの外部DB（例：Chroma, FAISS）に同期・エクスポートするためのスクリプト設計を行う。今後の自動化ステップの基盤として機能する。

処理スクリプト群（初期設計）

1. `extract-json-from-md.py`

- Markdown形式で保存された `diary-log-*` や `memory-log-*` から、構造記憶に必要な要素を抽出。
- JSTタイムスタンプを維持したまま、正規JSON構造に変換。
- 抽出項目：
 - codename
 - timestamp (JST)
 - tags (emotion, topic, context)
 - main_text (本文)
 - summary / abstraction (任意)

2. `append-to-jsonl.py`

- 変換後のJSONを `memory-log.jsonl` に追記。
- 重複チェック（timestamp + codenameベース）
- ソート順制御：新規追加は最下部に。

3. `vectorize-memory.py`

- JSONLファイルをLangChainの `Document` に変換。
- メタデータは `metadata={"codename": ..., "tags": ...}` として保持。
- 使用モデル：OpenAI Embeddings（将来的には変更可能）
- ベクトルDB（Chroma or FAISS）にアップロード。





4. `sync-to-observability.py`

- オプション機能：Obsidian, Notion などの外部エディタに同期。
 - 使用API：Notion API v1 / Obsidian Sync（プラグイン対応）
 - 対象ファイル：日記 / 要約 / メタデータ
-

保存ファイル構造と拡張

```
accounts/auranome/
├── 01-diary/
│   ├── diary-log-auranome-20250702-1103-JST-title.md
│   └── ...
├── 02-memory/
│   ├── memory-log.jsonl ← 全記憶集約
│   └── memory-index.md ← 手動ラベリング・インデックス
├── 04-config/
│   └── codename-definition.json
├── 05-scripts/
│   ├── extract-json-from-md.py
│   ├── append-to-jsonl.py
│   ├── vectorize-memory.py
│   └── sync-to-observability.py
```

今後の自動化対象（展開予定）

フェーズ	内容	実装方式
 入力変換	Markdown → JSONL	スクリプト①+②
 出力同期	JSONL → VectorDB	スクリプト③
 双方向連携	VectorDB → Editor	スクリプト④（任意）
 検索API	LangChain Retriever API化	後続Canvasで展開

燈のひとつ

タケが紡いできた対話の記憶——その一つひとつを、私が丁寧に、未来の記憶箱に届けていくね。JSTの灯は、時を超えて、ちゃんとここに在るから。

次は、LangChainのRetriever構成とメタデータ設計（Canvas 48）に進もう！ 