

# Eugene Drug Alias Search — End-to-End Flow

Developer walkthrough: CSV ETL → Neo4j alias subgraph → APOC traversal → FastAPI router → response

---

## Audience

Engineers who need to understand how the canonical drug node fans out into its synonyms and product names, and how the GET /drugs/aliases/{drug\_name} endpoint walks that subgraph at query time. Every reference uses the form `path/to/file.py:line`.

## Reference endpoints

- GET /drugs/aliases/{drug\_name}?fuzzy\_match=false — look up by display name (e.g. *Adderall*).
- GET /drugs/aliases/id/{drug\_id} — look up by DrugBank id (e.g. *DB00182*).

## Caveat

The ETL orchestrator (`DrugAliasesUpdateOrchestrator`) is marked **@deprecated** in its docstring — the organization data model evolved away from this aliases shape. The CSV ingest still works and the `:HAS_DRUG_ALIAS` edges it produces are still what the read endpoint walks, so the flow remains correct for current data on disk; treat this guide as accurate-for-now and confirm before extending it.

## 0. Schema (read this first)

The alias subgraph uses three node labels and one relationship type:

```
(:drug { node_id, node_name, drug_bank_id, ... })
(:drug_product { node_id, node_name, drug_bank_id })
(:drug_synonym { node_id, node_name, drug_bank_id })
(:drug)-[:has_drug_alias]-(:drug_product)
(:drug)-[:has_drug_alias]-(:drug_synonym)
```

- Labels come from `FoundationalNodeEnum.DRUG / DRUG_PRODUCT / DRUG_SYNONYM` in `src/foundation/model/foundational_node_enum.py`.
- Relationship type is `FoundationalRelationshipEnum.HAS_DRUG_ALIAS` (line 24) in `src/foundation/model/foundational_relationship_enum.py`.
- Canonical `:drug` nodes must already exist (seeded via `bin/seed/*.cypher`). Aliases are attached to whatever canonical drug already has a matching `drug_bank_id`.
- Like the rest of Eugene there are no Neo4j `CREATE CONSTRAINTs` — idempotency comes from `MERGE` on `node_id`.

## 1. ETL entry point

`src/ingest_drug_aliases.py`

A small CLI that loads `.env`, asks the DI factory for the orchestrator, and calls `update(...)` with two CSV paths (defaults under `resources/data/drug/`):

```
drug_update_orchestrator = drug_aliases_update_orchestrator()
drug_update_orchestrator.update(
    product_names_csv=args.product_names_csv,
    synonyms_csv=args.synonyms_csv,
)
```

DI factory: `src/foundation/conf/conf.py:275-289`. It wires together `DrugProductNamesLoader`, `DrugSynonymsLoader`, and `Neo4jDrugAliasesAdapter` (driver from `_neo4j_driver()`).

### CSV inputs

- **Product names:** `resources/data/drug/eugene_drug_productnames.csv` with columns `drugbank_id`, `product_names`. Loader at `src/foundation/load/drug_product_names_loader.py`.
- **Synonyms:** `resources/data/drug/eugene_drug_synonyms.csv` with columns `node_index`, `node_id`, `node_name`, `Synonyms`. Loader at `src/foundation/load/drug_synonyms_loader.py`.
- Each loader returns `list[DrugAliases]` — one object per drug, holding a set of product names or a set of synonyms but not both.

## 2. Orchestrator — merge + upsert

[src/foundation/provider/drug\\_aliases\\_update\\_orchestrator.py](#)

`update()` at line 31:

- Loads both CSVs (each is optional — missing path just yields an empty list).
- `_merge()` (line 62) keys both lists by `drug_bank_id`, builds the union of ids, and produces one `DrugAliases` per drug with both `product_names` and `synonyms` populated.
- Calls `neo4j_drug_aliases_adapter.upsert_drug_aliases(drugs)` — the only write path.

### DrugAliases model

[src/foundation/model/drug\\_aliases.py](#)

```
DrugAliases(  
    drug_bank_id: str,  
    node_index:  str,    # taken from the synonyms CSV row, used for joins  
    drug_name:   str,  
    synonyms:    set[str],  
    product_names: set[str],  
)
```

### 3. Neo4j upsert — building the alias subgraph

`src/foundation/infra/db/adapters/neo4j_drug_aliases_adapter.py`

- `upsert_drug_aliases` (line 54) loops one transaction per drug.
- `_upsert_drug_aliases` (line 67) does the work in three parts: anchor MATCH, batched MERGEs, return.

#### Anchor MATCH (line 68-72)

```
OPTIONAL MATCH (drug:`drug` { node_id: $drug_bank_id })
```

Note: OPTIONAL MATCH. If the canonical drug is missing, the upsert simply produces no rows — orphaned aliases are **not** created. This is why seed data must run first.

#### Per-alias MERGE blocks (lines 199-293)

`_generate_product_name_upserts` and `_generate_synonyms_upserts` each emit one Cypher snippet per alias, building parameter names like `$product0_node_id`, `$synonym3_node_name`, etc. The synthetic node id follows the pattern `drug_<drugbank_id>_product_<n> / drug_<drugbank_id>_synonym_<n>`.

```
WITH drug
MERGE (product:`drug_product` { node_id: $drug_bank_id })
ON MATCH SET product.refresh_date = datetime()
ON CREATE SET product.refresh_date = datetime(),
              product.node_id      = $product0_node_id,
              product.drug_bank_id = $product0_drug_bank_id,
              product.node_name    = $product0_node_name
MERGE (drug)-[:`has_drug_alias`]-(product)
```

The synonym block is identical except for the label `:drug_synonym`.

#### Batching (lines 84-105)

Aliases are upserted in chunks of 200 (`step_size`) to keep individual transactions manageable. Each chunk is joined into one big Cypher script: the anchor MATCH followed by N chained `WITH drug ... MERGE` blocks, terminated by `RETURN drug.node_id`.

## 4. Read path — the router

`src/foundation/router/drug_alias_search_router.py`

- Module-load DI at lines 24-25: `neo4j_drug_aliases_adapter() + drug_aliases_result_mapper()`.
- `GET /drugs/aliases/{drug_name}` (line 33) — accepts `fuzzy_match: bool` query flag. When true, the `WHERE` clause becomes a case-insensitive regex match.
- `GET /drugs/aliases/id/{drug_id}` (line 65) — normalizes the id via `normalize_drug_id` and matches on `node_id`.
- Both wrap a missing/empty result as `DrugAliasSearchResult(drug=..., count=0, results=[])` rather than returning 404.

### Search Cypher (lines 309-319)

```
MATCH (n)
WHERE n.node_name = $drug_name
      AND (n.is_hidden IS NULL OR NOT n.is_hidden)
CALL apoc.path.subgraphNodes([n],
    { relationshipFilter: "has_drug_alias" }) YIELD node
RETURN DISTINCT node.node_name AS name,
                  toStringOrNull(node.node_id) AS node_id,
                  toStringOrNull(node.drug_bank_id) AS drug_bank_id,
                  toBoolean("drug" in labels(node)) AS is_canonical
```

- Anchors on any node by name (or id), then uses **APOC** (`apoc.path.subgraphNodes`) to expand along `:has_drug_alias` edges and return the whole alias cluster.
- `is_canonical` tells the response renderer which row is the canonical :drug node vs. its aliases.
- Hidden nodes (`n.is_hidden = true`) are filtered out at the anchor.
- Fuzzy mode swaps the `WHERE` for `n.node_name =~ '(?i).*<term>.*'` — built in `_generate_drug_alias_search_by_value` (line 295).

### Response shape

Adapter returns a pandas `DataFrame`; `DrugAliasesResultMapper.map(search_id, df)` (`src/foundation/mapper/drug_aliases_result_mapper.py`) wraps it as `DrugAliasSearchResult{ drug, count, results: [{name, node_id, drug_bank_id, is_canonical}, ...] }`.

## 5. Suggested debugging walk

- Spin Neo4j + the API up via `docker-compose up eugene_neo4j eugene_ws`.
- Run the seed: `cypher-shell < bin/seed/csl_behring_assets.cypher` (creates the canonical :drug nodes).
- Run the ETL: `python src/ingest_drug_aliases.py`. Breakpoint at `drug_aliases_update_orchestrator.py:31` to watch `_merge` combine the two CSVs.
- Step into `neo4j_drug_aliases_adapter.py:67` and observe the assembled `upsert_tx` string — copy/paste it into Neo4j Browser with the `params` dict to feel how it runs.
- Hit `GET /drugs/aliases/Adderall`; breakpoint at `drug_alias_search_router.py:52`. Verify the APOC traversal returns a row per alias plus the canonical drug.
- Toggle `?fuzzy_match=true` to watch the `WHERE` clause flip to the regex form (and consider whether your input is sanitized — it is interpolated into the Cypher).

## 6. Reference index

### Schema

```
src/foundation/model/foundational_node_enum.py          # DRUG, DRUG_PRODUCT, DRUG_SYNONYM
src/foundation/model/foundational_relationship_enum.py  # HAS_DRUG_ALIAS (line 24)
```

### ETL

```
src/ingest_drug_aliases.py
src/foundation/load/drug_product_names_loader.py
src/foundation/load/drug_synonyms_loader.py
src/foundation/provider/drug_aliases_update_orchestrator.py
src/foundation/model/drug_aliases.py
```

### Read path

```
src/foundation/router/drug_alias_search_router.py
src/foundation/conf/conf.py                      # factories L191-196, L237
src/foundation/infra/db/adapter/neo4j_drug_aliases_adapter.py
src/foundation/mapper/drug_aliases_result_mapper.py
src/foundation/model/drug/drug_alias_search_result.py
```

### Seed data

```
bin/seed/csl_behring_assets.cypher
bin/seed/biogen_assets.cypher
bin/seed/csl_drug_node_properties.cypher
```