

Eugene Organization Alias Search — End-to-End Flow

Developer walkthrough: HTTP boundary → deprecated v1 organization adapter → APOC subgraph traversal → ListResponse

Audience

Engineers spelunking the older organization-alias subgraph (subsidiary, acquisition, merger, demerger, organization_spelling_variation). The lone endpoint GET /list/organization/aliases/{organization_name} is intended to fan out from a canonical organization node along :HAS_ALIAS edges and return the union of alias display names. As of this writing the read path has multiple bugs and the router is **not** registered in src/eugene_ws.py, so the route is effectively dead code in production — this guide documents it for the engineer who must revive or remove it.

Reference endpoint

- GET /list/organization/aliases/{organization_name} — e.g. GET /list/organization/aliases/CSL%20Behring. Returns ListResponse{count, results: [alias_name, ...]}.

Deprecation

Both the router (module-level docstring) and the adapter (Neo4jOrganizationAdapter, class docstring — @deprecated v1 organizations, replaced with v2 eugene load) are explicitly marked deprecated. The v2 model lives at OrganizationNodeEnum.ORGANIZATION_V2 (label = organization) and is read by the modern /organizations/* search router; this alias subgraph uses the v1 labels listed below and predates that migration.

0. Schema (read this first)

The alias subgraph centres on a single anchor `:company` node with one or more typed alias leaves attached via `:HAS_ALIAS` edges:

```
(:company { node_id, node_name, official_name, parent, ... })
(:subsidiary { node_id, node_name })
(:acquisition { node_id, node_name })
(:merger { node_id, node_name })
(:demerger { node_id, node_name })
(:organization_spelling_variation { node_id, node_name })
(:organization { node_id, node_name })

(:company)-[:HAS_ALIAS]-(:subsidiary)
(:company)-[:HAS_ALIAS]-(:acquisition)
(:company)-[:HAS_ALIAS]-(:merger)
(:company)-[:HAS_ALIAS]-(:demerger)
(:company)-[:HAS_ALIAS]-(:organization_spelling_variation)
```

Enum sources

- `OrganizationNodeEnum` — `src/organization/model/organization_node_enum.py`.
Note the **v1/v2/v3 collision**: `ORGANIZATION_V2 = 8, "organization"` and `ORGANIZATION_V3 = 8, "Organization"` share the ordinal 8 and differ only in label case. The v1 write path uses `COMPANY (company)` as the anchor label, while the v1 query path tries to `MATCH (:organization)` — a label mismatch (see section 2).
- `OrganizationRelationshipEnum.HAS_ALIAS` — `src/organization/model/organization_relationship_enum.py` (value `"has_alias"`).
It is the only relationship type in this subgraph; the write path emits an **undirected** `MERGE (organization)-[:HAS_ALIAS]-(alias)` edge.
- `FoundationalFoundationalNodeEnum.ORGANIZATION = 32, "Organization"` is the modern label used by the v2 search router and the patent/clinical-trial joins. It deliberately does **not** overlap with the v1 alias labels above.
- Like the rest of Eugene there are no Neo4j `CREATE CONSTRAINTs`; idempotency comes from `MERGE` on `{ node_name: $name }` (note: *not* `node_id` — two orgs with the same alias display name will collide).

1. HTTP entry — the router

`src/organization/router/organization_alias_search_router.py`

- Router prefix: `/list, tags ["organization"]` (lines 15-18).
- Module-load DI at line 20: `org_adapter = organization_adapter()` — resolved via `src/organization/conf/search_conf.py:23` which returns a `Neo4jOrganizationAdapter` bound to `GraphDbConnectionFactory.remote_neo4j_instance_from_env()`.
- Path parameter `organization_name` is `Annotated[str, Path(...), AfterValidator(validate_value)]` (lines 33-43). `validate_value` (`src/foundation/router/validate_util.py:36`) rejects values containing any of `%, _, $, ;, :, ^, *` — this is the SQL/Cypher-injection guard.
- Path constraints: `min_length=0, max_length=256` (note the `min_length=0` — FastAPI will still 404 on an empty segment, so this is effectively 1-256).
- Response wrapper: `ListResponse{count: int, results: list[str]}` from `src/foundation/router/model/list_response.py`, with `response_model_exclude_none=True`.
- Empty-result behaviour: when the adapter returns `None`, the handler returns `ListResponse(count=0, results=[])` rather than 404 (line 46-47). Otherwise it sorts the aliases in place before wrapping (line 48).

Handler body (lines 33-49)

```
@router.get(
    "/organization/aliases/{organization_name}",
    response_model=ListResponse,
    response_model_exclude_none=True,
)
async def list_organization_aliases(
    organization_name: Annotated[
        str,
        Path(..., max_length=256, min_length=0,
            example="CSL Behring"),
        AfterValidator(validate_value),
    ],
):
    aliases = org_adapter.find_aliases_by_name(
        organization=organization_name)
    if aliases is None:
        return ListResponse(count=0, results=[])
    aliases.sort()
    return ListResponse(count=len(aliases), results=aliases)
```

Is the router wired in `eugene_ws`?

No. `src/eugene_ws.py` imports and includes `drug_alias_search_router` (line 47, line 69) but there is no corresponding import for `organization_alias_search_router` and no `app.include_router(organization_alias_search_router.router)` call. The route therefore does not appear in OpenAPI and `curl /list/organization/aliases/CSL%20Behring` will 404 in any environment running the current `eugene_ws.py`. To revive it: add an `from organization.router import organization_alias_search_router` in `eugene_ws.py:47` block and append it to the router list at line 69.

2. Query adapter — the (broken) read Cypher

`src/organization/infra/db/neo4j_organization_adapter.py`

- Entry point: `find_aliases_by_name(organization)` (line 29). Opens a session, then a transaction, and delegates to `_find_aliases_by_name` (line 41).
- `_find_aliases_by_name` runs the Cypher returned by `_generate_search()` (line 354) with params `{ "organization_name": organization }` and pulls `record["aliases"]` from the single record (line 56).
- Wraps `DriverError` / `Neo4jError` and re-raises — no per-error mapping to HTTP status.

`_generate_search()` — verbatim (lines 354-368)

```
MATCH (o:`organization`)
WHERE o.organization_id = $organization_name
CALL apoc.path.subgraphNodes([n],
{
  relationshipFilter: "has_alias",
  maxLevel: 2
}) YIELD org
RETURN DISTINCT org.node_name as name,
  toStringOrNull(org.node_id) as org_id,
  toStringOrNull(org.organization_name) as org_name
```

Bugs to flag before you depend on this

- **Anchor label mismatch.** The Cypher matches `:`organization`` (the v2 label, via `OrganizationNodeEnum.ORGANIZATION_V2.value[1]`), but the write path in `_upsert_organization_resolution` (line 94-106) creates the anchor as `:company` (via `OrganizationNodeEnum.COMPANY.value[1]`). Anything written by this adapter is unreachable from this read.
- **Wrong anchor property.** The WHERE clause filters on `o.organization_id = $organization_name`, but the write path stores the human-readable name in `organization.node_name / organization.offical_name` (sic, typo preserved). No node has `organization_id` set by the ingest path.
- **Undefined Cypher variable.** `apoc.path.subgraphNodes([n], ...)` references `[n]` but the MATCH binds `o`, not `n` — Neo4j will reject this with `Variable `n` not defined`.
- **YIELD column mismatch.** `apoc.path.subgraphNodes` yields a single column named `node`, not `org`. The `YIELD org` clause will fail with `Unknown procedure output: `org``.
- **Missing RETURN column.** The read path expects `record["aliases"]` (adapter line 56) but the Cypher returns `name, org_id, org_name` — there is no `aliases` column. Even if the Cypher parsed, the dict-key lookup would raise.
- **Sibling adapter has the same bug pattern.** `Neo4jOrganizationQueryAdapter` (`neo4j_organization_query_adapter.py:56`) also references undefined `n` in its WHERE / `org` in its RETURN, and is similarly unused at runtime. Treat both as cut-and-paste drafts.

What the Cypher *should* look like

For reference, the working drug-alias equivalent (see `drug_alias_search_router.py:309-319`) uses the same shape but with consistent bindings:

```
MATCH (n:`company`)
WHERE n.node_name = $organization_name
AND (n.is_hidden IS NULL OR NOT n.is_hidden)
```

```
CALL apoc.path.subgraphNodes([n],
    { relationshipFilter: "has_alias",
      maxLevel: 2 }) YIELD node
RETURN collect(DISTINCT node.node_name) AS aliases
```

3. Mapper / response model

Unlike the drug-alias route there is no dedicated mapper: the adapter is contracted to return `list[str]` of alias display names directly. The router does the only post-processing — sort + wrap.

ListResponse

`src/foundation/router/model/list_response.py`

```
class ListResponse(BaseModel):
    count: int
    results: list[str]
```

Sample JSON (what the route would emit if the Cypher worked)

```
{
  "count": 4,
  "results": [
    "CSL",
    "CSL Behring AG",
    "CSL Behring LLC",
    "ZLB Behring"
  ]
}
```

Note the canonical anchor name itself is **not** filtered out by the read Cypher — the working version above would include the `:company` node alongside its aliases, unless the caller post-filters.

4. ETL — how aliases would be ingested

`src/organization/provider/organization_ingest_orchestrator.py`

- Entry CLI: `OrganizationIngestOrchestrator.ingest(checkpoint_dir)` (line 38). Loads resolutions from a checkpoint dir, runs them through the multipass merge orchestrator, then assigns ids.
- **Neo4j writes are commented out** (lines 51-52): `# todo: ingest into neo4j / # return self.ingest_organizations(...)`. The CLI currently returns only the list of merged `official_name` strings — nothing is persisted by the default ingest entry point.
- The write helper `ingest_organizations` (line 66) does still exist and delegates to `Neo4jOrganizationAdapter.upsert_organization_resolutions` (line 61), so a caller could re-enable writes by uncommenting the orchestrator line or calling the helper directly.
- Per-organization upsert: `_upsert_organization_resolution` (line 85) MERGEs a `:company` anchor on `{ node_name: $official_name }` (typo preserved from the source), then emits one MERGE `(alias:`` {...}) MERGE (organization)-[:HAS_ALIAS]-(alias)` block per alias.
- Aliases come from five buckets on `OrganizationResolution`: `subsidiaries`, `acquisitions`, `spelling_variations`, `mergers`, `demergers`. **Bug:** the merger and demerger generators are both passed `organization_resolution.spelling_variations` (lines 236-243), not the merger/demerger sets — so those bucket-specific aliases are silently dropped and spelling variations are written three times under three different bucket labels.
- All five generators delegate to `_generate_alias_upserts(..., org_node_type=ORGANIZATION)` (lines 260-303) — meaning every alias is labelled `:organization` regardless of which bucket it came from. The richer `:subsidiary / :acquisition / etc.` labels listed in the enum are **not** applied by this code path.
- Batching: aliases are upserted in chunks of `step_size = 200` (line 119), with one chained MERGE script per chunk and a final `RETURN organization.node_id`.

5. Suggested debugging walk

- Confirm the route is missing: `grep -n organization_alias_search_router src/eugene_ws.py` — expect zero hits. Add the `import + include_router` call before doing anything else.
- Stand up the stack: `docker-compose up eugene_neo4j eugene_ws`. Confirm `apoc.path.subgraphNodes` is available: in cypher-shell run `RETURN apoc.version()`.
- Seed at least one anchor: `CREATE (:company { node_name: 'CSL Behring', official_name: 'CSL Behring', node_id: 'organization_test' })` then attach two aliases via `MERGE (:organization { node_name: 'CSL' })` and `MERGE (a)-[:HAS_ALIAS]-(c)`.
- Hit the endpoint: `curl -s "http://localhost:8080/list/organization/aliases/CSL%20Behring" | jq ..` With the current Cypher you will see a 500 with an `apoc.path.subgraphNodes / Variable `n` not defined` error in the `eugene_ws` logs — that confirms you have actually reached the adapter.
- Breakpoint at `neo4j_organization_adapter.py:42` (inside `_find_aliases_by_name`), inspect the generated search string, copy it into Neo4j Browser with the `$organization_name` parameter, and iterate the fixes from section 2 until it returns a row with an `aliases` column.
- To exercise the write path that currently feeds this read, uncomment `organization_ingest_orchestrator.py:52`, point `ingest(checkpoint_dir=...)` at a directory of `OrganizationResolution` JSON, and watch `_generate_alias_upserts` produce one Cypher block per alias. Worth fixing the merger/demerger bucket bug (section 4) at the same time.

6. Reference index

HTTP boundary

```
src/organization/router/organization_alias_search_router.py
src/foundation/router/model/list_response.py
src/foundation/router/validate_util.py          # validate_value (L36)
src/eugene_ws.py                               # router NOT included
```

DI

```
src/organization/conf/search_conf.py           # organization_adapter() (L23)
src/graph/infra/db/graph_db_connection_factory.py # _neo4j_driver()
```

Read path (adapter + buggy Cypher)

```
src/organization/infra/db/neo4j_organization_adapter.py
    find_aliases_by_name          (L29)
    _find_aliases_by_name        (L41)
    _generate_search              (L354) -- bugs documented in section 2
src/organization/infra/db/neo4j_organization_query_adapter.py
    -- sibling adapter, also unused, similar bugs
```

Schema enums

```
src/organization/model/organization_node_enum.py
    COMPANY, ORGANIZATION, ORGANIZATION_V2/V3,
    SUBSIDIARY, ACQUISITION, MERGER, DEMERGER, SPELLING_VARIATION
src/organization/model/organization_relationship_enum.py
    HAS_ALIAS
src/foundation/model/foundational_node_enum.py
    ORGANIZATION = 32, "Organization" # v2 modern label
```

ETL / write path

```
src/organization/provider/organization_ingest_orchestrator.py
    ingest              (L38) -- Neo4j writes commented out (L51-52)
    ingest_organizations (L66)
src/organization/infra/db/neo4j_organization_adapter.py
    upsert_organization_resolutions (L61)
    _upsert_organization_resolution (L85)
    _generate_organization_alias_upserts (L171)
    _generate_alias_upserts            (L305)
src/organization/model/organization_resolution.py
src/organization/load/organization_resolution_loader.py
```