

Eugene Search Path — End-to-End Flow

Developer walkthrough: HTTP request → id validation → FoundationalPathProvider → Neo4j SHORTEST / ANY traversal → ShortestPaths / Reachability response

Audience

Engineers who need to understand how Eugene answers "what is the shortest path between two graph nodes?" and "is node A reachable from node B within N hops?". Unlike the drug-alias and patent flows, the search-path route is label-agnostic: the Cypher queries match **any** node label and **any** relationship type, anchored only on the `node_id` property. Every reference uses the form `path/to/file.py:line`.

Reference endpoints

- `GET /graph/path/start/{start_id}/end/{end_id}?n_hop=<int>` — returns up to 10 shortest paths (each a list of `node_ids`) between two nodes.
- `GET /graph/reachability/start/{start_id}/end/{end_id}?n_hop=<int>` — returns a boolean: does any path of length $\leq n_hop$ exist?

How to read this guide

The flow is short: the router is thin, there is exactly one provider and one Neo4j adapter, and the response models are flat frozen dataclasses. Read section 0 first — it explains why this route does not own any ETL of its own.

0. Schema (read this first)

Unlike the drug-alias or patent routes, the search-path queries do **not** pin themselves to any label or relationship type. The Cypher patterns look like:

```
(startNode { node_id: $start_id })-[:link]-{0,N}(endNode { node_id: $end_id })
```

- Both endpoints are **anchored on the node_id property** only — no label filter, no relationship-type filter.
- Any node previously written to the graph by an upstream ETL pipeline (drugs, patents, clinical trials, pubmed, organizations, etc.) is a valid `start_id` or `end_id`.
- Edges are walked undirected: the Cypher uses `-[:link]-`, not `-[:link]->`.
- The graph must be **pre-populated** for these queries to return anything. The search-path route is read-only — see section 4.
- Hop bound `{0, N}` is interpolated from the `n_hop` query parameter. 0 hops means the start and end are the same node.

1. HTTP entry

[src/foundation/router/search_path_router.py](#)

- Module-load DI at line 20: `search_path_provider = foundational_path_provider()` — constructed once at import time and reused across requests.
- Router prefix is `/graph` with tag `foundation` (lines 15-18).
- `GET /graph/path/start/{start_id}/end/{end_id}` (line 23) — delegates to `provider.find_shortest_path_by_start_id_and_end_id(...)` at line 58.
- `GET /graph/reachability/start/{start_id}/end/{end_id}` (line 63) — delegates to `provider.find_is_reachable_by_start_id_and_end_id(...)` at line 98, then logs the result.
- Both endpoints declare `n_hop` as a `Query(gt=0, le=4) int` — FastAPI rejects out-of-range values with HTTP 422 before the provider is even called (lines 48-56 and 88-96).

Id validation

[src/foundation/router/validate_util.py](#)

Both `start_id` and `end_id` are wrapped with `AfterValidator(validate_id).validate_id` (line 98) rejects ids that contain any of `%, $, ;, :, ^, *` with a `ValueError`:

```
def validate_id(id: str) -> str:
    invalid_chars = ["%", "$", ";", ":", "^", "*"]
    is_valid = not has_invalid_char(invalid_chars=invalid_chars, value=id)
    if not is_valid:
        raise ValueError(f"id cannot have a special character: {invalid_chars}")
    return id
```

DI factory

`foundational_path_provider()` lives at `src/foundation/conf/conf.py:358`. It calls `_foundational_path_provider(...)` (line 364) which wires a `Neo4jFoundationalPathAdapter` (factory at `conf.py:147`) into a `FoundationalPathProvider`. The `Neo4j Driver` comes from the shared `_neo4j_driver()` helper.

2. Query adapter — the two Cypher patterns

`src/foundation/infra/db/adapter/neo4j_foundational_path_adapter.py`

- Class constants (lines 18-19): `MAX_SUPPORTED_HOPS = 4` and `TOP_N = 10`.
- Public entry points: `find_shortest_path_by_start_id_and_end_id` (line 30) and `find_is_reachable_by_start_id_and_end_id` (line 38).
- Both call `_ensure_valid_n_hop_size_or_throw(n_hop)` (line 106) and `ensure_connection(self.driver)` before issuing Cypher.

Shortest path Cypher (lines 89-96)

```
MATCH path = SHORTEST 10
(startNode { node_id: $start_id })-[:link]-{0,N}(endNode { node_id: $end_id })
RETURN [n in nodes(path) | n.node_id] AS paths
```

- `SHORTEST 10` is Neo4j 5's *k*-shortest-paths form — returns up to `TOP_N = 10` shortest paths, not just one.
- Each returned row is a list of `node_id` values along the path, projected by the comprehension `[n in nodes(path) | n.node_id]`.
- `N in {0, N}` is the `n_hop` argument, string-interpolated via `%s` (lines 93-96). It is not a Cypher parameter — that is why `n_hop` is range-checked at two layers (`FastAPI le=4 + _ensure_valid_n_hop_size_or_throw`).

Reachability Cypher (lines 98-104)

```
MATCH path = ANY
(startNode { node_id: $start_id })-[:link]-{0,N}(endNode { node_id: $end_id })
RETURN toBoolean(count(path)) as is_reachable
```

- `SHORTEST 10` enumerates paths; `ANY` short-circuits as soon as one path of length $\leq N$ is found. Use reachability when you only need a yes/no — it is the cheaper query.
- `count(path)` wrapped in `toBoolean(...)` collapses to `true` if any row is produced, `false` otherwise.
- `_find_is_reachable` (line 62) reads `df["is_reachable"][0]` — defaults to `False` if the driver returned `None`.

Hop guard (lines 106-113)

```
def _ensure_valid_n_hop_size_or_throw(self, n_hop: int) -> int:
    n_hop = max(1, n_hop)
    max_supported_hops = Neo4jFoundationalPathAdapter.MAX_SUPPORTED_HOPS
    if n_hop > max_supported_hops:
        raise ValueError(
            "Request for N hop query of size {n_hop} exceeds max supported hops of {max_supported_hops}"
        )
    return n_hop
```

The Eugene graph is highly connected (the comment at line 17 spells it out). A 5-hop query can fan out into millions of paths, which is why the cap is `MAX_SUPPORTED_HOPS = 4` and the router declares `le=4` on the query param.

3. Provider & response models

[src/foundation/provider/foundational_path_provider.py](#)

- FoundationalPathProvider is a thin wrapper over the adapter — no business logic beyond DataFrame → dataclass shaping.
- `find_shortest_path_by_start_id_and_end_id` (line 26): if the adapter returns None, the provider returns an empty ShortestPaths(count=0, paths=()) rather than raising (lines 32-39).
- Otherwise, each paths row from the DataFrame is converted to a tuple, and all rows are packed into a tuple-of-tuples for hashability (lines 41-52).
- `find_is_reachable_by_start_id_and_end_id` (line 54) simply forwards to the adapter, which already returns a Reachability dataclass.
- Both methods are decorated with `@log_time` — latency is logged per call.

ShortestPaths dataclass

[src/foundation/model/shortest_paths.py](#)

```
@dataclass(frozen=True, unsafe_hash=True)
class ShortestPaths:
    start_id: str
    end_id: str
    max_hop: int
    count: int
    paths: Tuple[Tuple[str, ...], ...]
```

Example response (curl GET

/graph/path/start/DB00846/end/DB00538?n_hop=2)

```
{
  "start_id": "DB00846",
  "end_id": "DB00538",
  "max_hop": 2,
  "count": 3,
  "paths": [
    ["DB00846", "P12345", "DB00538"],
    ["DB00846", "GENE_ABL1", "DB00538"],
    ["DB00846", "PATHWAY_07", "DB00538"]
  ]
}
```

Reachability dataclass

[src/foundation/model/reachability.py](#)

```
@dataclass(frozen=True, unsafe_hash=True)
class Reachability:
    start_id: str
    end_id: str
    max_hop: int
    is_reachable: bool
```

Example response (curl GET

/graph/reachability/start/DB00846/end/DB00538?n_hop=3)

```
{
  "start_id": "DB00846",
  "end_id": "DB00538",
  "max_hop": 3,
  "is_reachable": true
}
```

```
    "is_reachable": true  
}
```

4. Data source — where the graph comes from

The search-path route is strictly read-only. It owns no loader, no orchestrator, no ETL CLI — it queries whatever the upstream pipelines have already MERGEed into Neo4j.

- **Drug pipeline** — `src/ingest_drug_aliases.py` + drug node seed scripts under `bin/seed/*.cypher`.
- **Patent pipeline** — covered in `EUGENE_PATENT_FLOW_GUIDE.pdf`. Patent nodes carry `node_id` of the form `P<number>`.
- **Clinical trial pipeline** — ingest scripts under `src/ingest_*` for ClinicalTrials.gov sources.
- **PubMed pipeline** — literature ingest producing `:publication` nodes joined into the graph.
- All upstream ETL writers MERGE on `node_id`; the search-path queries rely on that property being unique and indexed.
- If `SHORTEST 10` returns 0 rows, the graph — not the route — is the most likely culprit. Verify the start/end nodes exist (see debugging walk).

5. Suggested debugging walk

- Spin up the stack: `docker -compose up eugene_neo4j eugene_ws`. Make sure at least one ETL has populated the graph.
- Hit the shortest-path endpoint: `curl 'http://localhost:8000/graph/path/start/DB00846/end/DB00538?n_hop=2'`. Set a breakpoint at `src/foundation/router/search_path_router.py:58` to inspect the validated `start_id`, `end_id`, `n_hop`.
- Hit the reachability endpoint: `curl 'http://localhost:8000/graph/reachability/start/DB00846/end/DB00538?n_hop=3'`. Breakpoint at `search_path_router.py:98`.
- Sanity-check in Neo4j Browser that the nodes actually exist and confirm their labels: `MATCH (n {node_id: 'DB00846'}) RETURN labels(n), n`.
- Step into `Neo4jFoundationalPathAdapter._build_shortest_path_by_ids_query` (line 89) and copy the rendered Cypher string into Neo4j Browser with `$start_id / $end_id` set in the params panel — gives a feel for fan-out at each `n_hop`.
- Force a validation error: `curl 'http://localhost:8000/graph/path/start/foo%25bar/end/DB00538?n_hop=2'` — observe `validate_id` reject the % with HTTP 422. Repeat with `n_hop=5` to watch FastAPI's `le=4` guard fire.

6. Reference index

HTTP layer

```
src/foundation/router/search_path_router.py
src/foundation/router/validate_util.py      # validate_id L98
```

DI factories

```
src/foundation/conf/conf.py                # foundational_path_provider L358
                                           # neo4j_foundational_path_adapter L147
```

Provider

```
src/foundation/provider/foundational_path_provider.py
```

Neo4j adapter

```
src/foundation/infra/db/adapter/neo4j_foundational_path_adapter.py
                                           # SHORTEST 10 L89-96
                                           # ANY L98-104
                                           # n_hop guard L106-113
```

Response models

```
src/foundation/model/shortest_paths.py
src/foundation/model/reachability.py
```

Adjacent / upstream

```
src/ingest_drug_aliases.py                # one of several ETL writers
bin/seed/*.cypher                         # canonical node seeds
```