

Eugene Agent Chat — End-to-End Flow

Developer walkthrough: chat request → intent classifier → Strands agent + MCP tools → graph → streamed response

Audience

Engineers who need to understand how Eugene answers natural-language questions on top of the same Neo4j graph that the REST endpoints serve. Unlike `/patents/*` the agent path does not run a single hand-written Cypher — an LLM reasons over MCP tools and produces a narrative answer with citations. Every reference uses the form `path/to/file.py:line`.

Reference endpoints

- `POST /agent/api/query` — single-shot, returns a complete `ChatQueryResponse`.
- `POST /agent/api/query/stream` — newline-delimited JSON stream of `{type, content, tool, tool_input, ...}` envelopes that the Next.js UI uses to render a live tool-call trace + context graph.

Service map

- **eugene-agent-ws** (this guide) — FastAPI service hosting the Strands agent. Codebase under `agents/eugene-agent-ws/`.
- **eugene-mcp** — MCP server exposing Eugene's read-only graph tools (`fetch_*`, `lookup_*`, `find_organization_*`). Codebase under `agents/eugene-mcp/src/tools/`.
- **eugene_ws** — the core REST API the MCP tools ultimately call.

1. Boot & routing

[agents/eugene-agent-ws/src/eugene_chat_ws.py](#)

- FastAPI app is mounted under `root_path="/agent/api"` (line 54).
- Routers wired by `add_routers()` at lines 30-37: `auth_router`, `root_router`, `health_router`, `chat_query_agent_router`.
- Shared middleware (JSON logging, rate limiting, request-id, CORS) mirrors the core `eugene_ws` service.

Router file

[agents/eugene-agent-ws/src/query/router/chat_query_agent_router.py](#)

- Module-load DI at line 26: `eugene_agent = eugene_data_agent()` — the Strands agent is constructed once and reused.
- Auth: every request goes through `get_current_token` + `get_current_user` dependencies (MS Entra JWT validation).
- `POST /query` (line 29) — awaits `eugene_agent.execute(...)` and wraps the result with `unwrap_agent_result()`.
- `POST /query/stream` (line 74) — returns a `StreamingResponse` over `generate_chat_response()`, which serializes each event chunk as JSON + newline.

2. Intent classifier — pruning the toolset

`agents/eugene-agent-ws/src/query/util/intent_classifier.py`

Before the agent runs, the streaming endpoint calls `classify_intent(prompt, user_selected)`. The classifier is intentionally a small set of regexes — the docstring explains why: fewer tools means shorter system prompt, smaller ReAct surface area, and >90% precision on Eugene's well-defined domain without an extra LLM call.

Regex buckets

```
_GRAPH_RE    matches: drug, disease, gene, protein, pathway, trial,
                indication, organization, target, neighbors,
                biogen, roche, csl, hemophilia, factor, ...
_WEB_RE       matches: web, internet, online, google, latest, news, ...
_PATENT_RE    matches: patent, uspto, ip, assignee, patent number
_PUBMED_RE    matches: pubmed, publication, paper, literature, ...
```

Rules

- Graph-concept match → add `ToolRequestEnum.EUGENE` (MCP tools).
- Web / pubmed / patent match → add `ToolRequestEnum.HTTP` (`search_pubmed`, `search_patents_web`, `http_request`).
- Nothing matched → default to EUGENE — graph queries are read-only and bounded, the safest fallback.
- Whatever the user ticked in the UI is always included.

Output: `IntentResult(tools, reason)`. The reason string is logged so you can grep for misclassifications.

3. The Strands agent

`agents/eugene-agent-ws/src/query/agent/eugene_data_agent.py`

DI

`agents/eugene-agent-ws/src/query/conf/conf.py` constructs `EugeneDataAgent(eugene_mcp_server_url, model)`. The model is selected by `LLM_PROVIDER` env var (falls back to whichever API key is set):

- **Anthropic** (default if `ANTHROPIC_API_KEY` present): `claude-sonnet-4-20250514`, override via `ANTHROPIC_MODEL_ID`.
- **OpenAI** fallback: `gpt-4.1-mini`, override via `OPENAI_MODEL_ID`.
- Wired in `LlmFactory` at `agents/eugene-agent-ws/src/query/infra/llm/llm_factory.py`.

System prompt (`eugene_data_agent.py:41-59`)

You are Eugene, an agent that answers biomedical competitive-intelligence questions by querying the CSL Behring Eugene knowledge graph.

Scope: drugs, diseases, gene/proteins, pathways, patents, clinical trials, organizations, and the relationships between them.

Follow the ReAct pattern (Reason -> Act -> Observe). Prefer MCP tools (`fetch_*`, `find_*`, `lookup_*`) for facts. When the graph does not contain the requested information, state that clearly and stop -- do not loop.

TOOL GUIDE:

- Graph lookup (ingested drugs, genes, ...): MCP `fetch_/lookup_/find_*`
- Latest patents on the open web: `search_patents_web(query)`
- Biomedical literature (PubMed): `search_pubmed(query, max_results)`
- Arbitrary URL fetch: `http_request` (only non-biomedical pages)

HARD RULES:

1. No calculator / `current_time` unless the question is about that.
2. No retrying the same tool with same args more than twice.
3. No invented data. If no tool answers, say so + provide a search link.
4. ≤ 20 tool calls per turn.

Citation requirement: when asserting a fact from a tool result, cite `[node_id=<id>, tool=<tool_name>]` so the UI can render a context graph.

4. Tools

Local tools (eugene_data_agent.py:246-260)

- `calculator`, `current_time` — opt-in only via `EUGENE_ALLOW_UTILITY_TOOLS=true`. They caused runaway ReAct loops on simple data questions, so they ship disabled.
- `python_repl` — opt-in via `EUGENE_ALLOW_PYTHON_REPL=true`.
- `search_pubmed` & `search_patents_web` — added only when the request includes the HTTP tool group. Implementations at `agents/eugene-agent-ws/src/query/tools/external_tools.py`.
- `http_request` (Strands built-in) — only added alongside the search tools.

MCP tools (eugene-mcp service)

Listed dynamically via `mcp_client.list_tools_sync()` (line 259) when the request includes `ToolRequestEnum.EUGENE`. The MCP server lives at `agents/eugene-mcp/src/eugene_mcp.py` and exposes one tool file per domain:

```
agents/eugene-mcp/src/tools/eugene_drug_tools.py
agents/eugene-mcp/src/tools/eugene_graph_tools.py
agents/eugene-mcp/src/tools/eugene_node_tools.py
agents/eugene-mcp/src/tools/eugene_organization_tools.py
agents/eugene-mcp/src/tools/eugene_fetch_tools.py
agents/eugene-mcp/src/tools/eugene_fact_tools.py
```

Each tool ultimately HTTP-calls a REST endpoint on `eugene_ws` (the same routers covered by the patent and drug-alias guides). The MCP layer exists so the agent gets a typed, discoverable interface; it does not bypass the REST API.

MCP transport

`_build_mcp_client` (line 283) uses `streamable_http_client` over an `httpx.AsyncClient` with the user's bearer token forwarded as `Authorization`. SSL verify defaults `true`; dev opt-out via `EUGENE_MCP_VERIFY_SSL=false`. Timeouts: connect 10s / read 60s / write 30s.

5. ReAct execution & safety rails

execute() — non-streaming (line 71)

- Builds a per-request MCP client, then `_init_agent(...)` with a per-conversation `SlidingWindowConversationManager(window_size=10, per_turn=2)` — AGT-01 fix (was shared across concurrent users).
- `FileSessionManager(session_id=conversation_id)` persists the conversation to disk for resumability.
- Runs the agent inside with `mcp_client:` — MCP session is alive for exactly one user turn.

execute_stream() — streaming (line 99)

Same setup, but yields `{type, content, tool, tool_input, tool_id, tool_output}` events as the agent advances. Three kinds of safety rail are wrapped around the stream:

- **Hard tool-call budget** — `_HARD_TOOL_CALL_BUDGET` (default 25, env `EUGENE_MAX_TOOL_CALLS`). When exhausted, the loop is killed with a visible warning.
- **Duplicate-input breaker** — if the agent calls the same `(tool, json(args))` more than `_MAX_DUPLICATE_TOOL_CALLS=2` times, the loop is killed (AGT-02).
- **Wall-clock timeout** — `_AGENT_STREAM_TIMEOUT_S` (default 180s) via `_with_timeout`. Stalled MCP/LLM cannot hold the HTTP connection open indefinitely (AGT-03).
- **Iteration cap** — `_AGENT_MAX_ITERATIONS` (default 20) is set post-construction on the Strands Agent (line 275-279). Belt-and-braces in case Strands drops the kwarg between versions.

Event extraction (lines 302-403)

- `_extract_tool_events` emits `tool_call/tool_result` events from mid-stream Strands events — defensive, because event shapes differ across providers and Strands versions.
- `_harvest_messages` walks `agent.messages` after the stream completes — this is the authoritative path, since `agent.messages` is stable across versions.
- `_flatten_tool_output` parses JSON blocks so the UI can extract nodes/relationships and build the live context graph.

6. Suggested debugging walk

- Start the stack: `docker-compose up eugene_neo4j eugene_ws eugene_mcp eugene_agent_ws`. Confirm `EUGENE_MCP_SERVER_URL` and an LLM API key are set in `docker.env`.
- Get a JWT via `POST /agent/api/auth/...` (see `router/auth/auth_router.py`).
- Hit the non-streaming endpoint with `curl POST /agent/api/query` body `{"prompt": "List drugs that target ABL1", "include_tools": []}`. Breakpoint at `chat_query_agent_router.py:51`.
- Step into `EugeneDataAgent.execute`. Inspect `agent.tool_names` after `_init_agent` to confirm the intent-pruned toolset.
- Switch to the streaming endpoint: `curl -N -d ... /agent/api/query/stream`. Tail logs to watch `tool_call/tool_result` envelopes flow.
- Force a runaway: ask a prompt that tempts the model to call `current_time` repeatedly (with `EUGENE_ALLOW_UTILITY_TOOLS=true`) — observe the duplicate-input breaker fire in `_register_tool_call`.
- Trace one MCP tool: pick e.g. `fetch_drug_by_id` in `agents/eugene-mcp/src/tools/eugene_drug_tools.py` and follow its HTTP call to `eugene_ws` — ends up at the same `/drugs/...` router covered in the drug-alias guide.

7. Reference index

Boot & routing

```
agents/eugene-agent-ws/src/eugene_chat_ws.py
agents/eugene-agent-ws/src/query/router/chat_query_agent_router.py
agents/eugene-agent-ws/src/router/auth/auth.py
```

Intent & request shape

```
agents/eugene-agent-ws/src/query/util/intent_classifier.py
agents/eugene-agent-ws/src/query/util/validation.py
agents/eugene-agent-ws/src/query/model/chat_query_request.py
agents/eugene-agent-ws/src/query/model/chat_query_response.py
agents/eugene-agent-ws/src/query/model/tool_request_enum.py
```

Agent

```
agents/eugene-agent-ws/src/query/agent/eugene_data_agent.py
agents/eugene-agent-ws/src/query/conf/conf.py
agents/eugene-agent-ws/src/query/infra/llm/llm_factory.py
agents/eugene-agent-ws/src/query/util/agent_debug.py
agents/eugene-agent-ws/src/query/util/response.py
```

Local tools

```
agents/eugene-agent-ws/src/query/tools/external_tools.py
```

MCP server (separate service)

```
agents/eugene-mcp/src/eugene_mcp.py
agents/eugene-mcp/src/tools/eugene_drug_tools.py
agents/eugene-mcp/src/tools/eugene_graph_tools.py
agents/eugene-mcp/src/tools/eugene_node_tools.py
agents/eugene-mcp/src/tools/eugene_organization_tools.py
agents/eugene-mcp/src/tools/eugene_fetch_tools.py
agents/eugene-mcp/src/tools/eugene_fact_tools.py
```

Env knobs (defaults shown)

```
EUGENE_AGENT_MAX_ITERATIONS = 20
EUGENE_AGENT_STREAM_TIMEOUT_S = 180
EUGENE_MAX_TOOL_CALLS = 25
EUGENE_MCP_VERIFY_SSL = true
EUGENE_ALLOW_PYTHON_REPL = false
EUGENE_ALLOW_UTILITY_TOOLS = false
LLM_PROVIDER = anthropic | openai (auto-detected from key)
ANTHROPIC_MODEL_ID = claude-sonnet-4-20250514
OPENAI_MODEL_ID = gpt-4.1-mini
```