

Eugene Agent Root Route — End-to-End Flow

Developer walkthrough: HTTP boundary → FastAPI router → static welcome payload (no auth, no I/O)

Audience

Engineers onboarding to the Eugene **agent-ws** sidecar (the agentic chat webservice) who need a single, file-referenced trace of GET / as it lives on that service. This is **not** the same root as the main eugene_ws service — that one is documented in EUGENE_ROOT_FLOW_GUIDE . pdf. The agent-ws variant is thinner still: no JWT dependency, no user claims, just a single static message. The only subtle bit is the FastAPI root_path="/agent/api" prefix on the app, which means the route appears externally at /agent/api/ even though the router itself declares prefix="". Every reference uses the form path/to/file.py:line.

Reference endpoint

GET / on the agent-ws app — externally reachable as GET /agent/api/ when fronted by the standard ingress. No path params, no query params, no body, no auth requirement. Response is a JSON object with a single key, message.

Sample call

```
# direct against the agent-ws container (no ingress):  
curl -s 'http://localhost:18510/' | jq .
```

```
# through the ingress / reverse proxy (root_path applied):  
curl -s 'http://localhost:8080/agent/api/' | jq .
```

Declared in agents/eugene-agent-ws/src/router/root_router.py:14-18. Router prefix "" (empty), tag root.

0. Purpose (read this first)

This route exists for three concrete reasons, all operational rather than functional:

- **Liveness signal for the agent sidecar.** Container probes and curl-from-laptop sanity checks need a cheap endpoint that returns 200 when the agent-ws process is up. Unlike the main service's root, this one is unauthenticated — it is suitable as a Kubernetes / Docker liveness probe target directly.
- **Swagger pointer.** The literal welcome string explicitly directs the caller to /docs for OpenAPI exploration. An operator hitting the bare / with a browser gets actionable guidance instead of a 404.
- **Routing smoke test.** Because the agent-ws app is mounted behind `root_path="/agent/api"`, a successful 200 at /agent/api/ proves end-to-end that (a) the ingress / reverse proxy is forwarding correctly, (b) FastAPI is rewriting incoming paths with the `root_path` stripped, and (c) the router was registered before the app started serving.

What it deliberately is not

- **Not a health check with deep checks.** Real health probes go to /health on the agent-ws health router (`agents/eugene-agent-ws/src/router/health_router.py`).
- **Not authenticated.** Unlike the main service's root (`src/router/root_router.py`), this handler takes no `Depends(get_current_user)` and returns no user claims. Do not treat a 200 here as evidence that JWT auth is wired correctly.
- **Not an OpenAPI landing page.** FastAPI's generated docs live at /docs (i.e. externally at /agent/api/docs); the root route does not redirect there, it only mentions the path in the welcome string.

1. HTTP entry — the FastAPI router

App construction

agents/eugene-agent-ws/src/eugene_chat_ws.py:53-57 constructs the FastAPI app:

```
app = FastAPI(
    root_path="/agent/api",
    title="Eugene Agent WS",
    version="2.0.0",
)
```

The `root_path="/agent/api"` argument is load-bearing for this guide. FastAPI uses it to (a) generate correct URLs in the OpenAPI document so Swagger UI works when fronted by a reverse proxy that strips `/agent/api`, and (b) advertise the effective external mount point. Internally, the route is still declared at `/` on the `APIRouter`; the prefix is applied by the ingress, not by the app itself.

Wiring

agents/eugene-agent-ws/src/eugene_chat_ws.py:32 — `from router import root_router, health_router`. agents/eugene-agent-ws/src/eugene_chat_ws.py:35 packs the routers into a list `[auth_router, root_router, health_router, chat_query_agent_router]`, and line 37 calls `app.include_router(router.router)` for each. `root_router` is registered second — immediately after the auth router — which means the bare slash is bound very early in app startup.

Router file (verbatim)

agents/eugene-agent-ws/src/router/root_router.py

```
import logging

from fastapi import APIRouter

router = APIRouter(
    prefix="",
    tags=["root"],
)

logger = logging.getLogger(__name__)

@router.get("/")
async def app_root():
    return {
        "message": "Welcome to the EUGENE agentic chat webservice. Visit the Swagger docs at /docs"
    }
```

Things to notice

- **Empty prefix.** `prefix=""` means the route binds at the literal `/` of the app. Combined with `root_path="/agent/api"` on the app constructor, this becomes `/agent/api/` externally.
- **No Depends.** The handler signature is `async def app_root()` — no parameters, no dependencies, no auth. Anyone who can reach the port gets a 200.
- **Tag.** `tags=["root"]` groups the operation under a `root` section in the auto-generated Swagger UI at `/docs`.
- **No response_model.** The dict is serialized as-is by FastAPI's default `jsonable_encoder`.

- `async def` **but no I/O**. The handler awaits nothing. It exists as a coroutine purely for stylistic consistency with the rest of the agent-ws routes.

- **Module logger declared but unused**. `logger = logging.getLogger(__name__)` is present at module load but never called inside the handler; successful calls log nothing handler-side.

2. Handler internals

The handler body is a single return of a one-key dict literal. There is no dependency resolution, no database call, no adapter, no mapper, and no logging. By volume this is the simplest handler in the entire Eugene codebase.

Resolution order

- FastAPI receives an HTTP request whose path (after ingress strips `/agent/api`) is `/`.
- It matches against the registered GET `/` operation on `root_router` and invokes `app_root()`.
- No parameters, no dependencies — the coroutine returns immediately with the welcome dict.
- FastAPI serializes the dict via `jsonable_encoder` and writes a 200 response with `content-type: application/json`.

Middleware that still runs

Even though the handler itself does nothing, the request still traverses the middleware stack configured in `agents/eugene-agent-ws/src/eugene_chat_ws.py`:

- **SEC-03 rate limiting** (lines 60-63): `SlowAPIMiddleware` with the limiter from `make_limiter()`. Excessive calls to GET `/` will eventually be throttled.
- **CORS** (lines 66-73): `CORSMiddleware` with origins controlled by the `EUGENE_CORS_ORIGINS` environment variable (default: `http://localhost:18502, http://localhost:3000`).
- **OBS-01 request-id** (line 76): `request_id_middleware` attaches a request id used in logs and surfaced back to the caller via the `X-Request-ID` response header.
- **Rate-limit exception handler** (line 62): `RateLimitExceeded` is translated by `_rate_limit_exceeded_handler` into a 429 response.

Set your first breakpoint

At `agents/eugene-agent-ws/src/router/root_router.py:16` (the return statement). There is essentially nothing to inspect; the value is hard-coded. A breakpoint here is useful only to confirm the handler is being reached at all (useful if you are diagnosing a 404 caused by misconfigured `root_path="/agent/api"` or a stale ingress).

3. Response model

The handler returns a Python dict literal with a single string-valued key. FastAPI serializes it via the default `jsonable_encoder`; the route declares no `response_model`, so the wire format is exactly the dict structure.

Response shape (verbatim from the handler)

```
{
  "message": "Welcome to the EUGENE agentic chat webservice. Visit the Swagger docs at /docs"
}
```

Notes on the message string

- The message references `/docs`. When the service is behind the standard ingress, the externally reachable Swagger UI is at `/agent/api/docs` — the message gives the *internal* path, which is what curl-from-inside-the-container will see, but external callers need to remember the `/agent/api` prefix.
- The string is hard-coded; it does not include the service version. Build / release metadata is not exposed by this endpoint.
- There are no template substitutions; the response is byte-identical from one call to the next.

Status codes

- `200 OK` — normal path. Always returned unless a middleware short-circuits the request.
- `429 Too Many Requests` — from `SlowAPIMiddleware` when the rate limit configured in `make_limiter()` is exceeded.
- `500 Internal Server Error` — only if an unhandled exception escapes the middleware stack. The handler itself cannot raise.

4. Use in production — frontend probe and liveness echo

Because the route does no I/O at all, it is the cheapest endpoint on the agent-ws sidecar and is the natural target for any check that just needs "is the process answering HTTP?"

Recommended uses

- **Frontend probe.** The Next.js UI in `agents/eugene-agent-ui-next/` can hit `GET /agent/api/` on startup to confirm the agent backend is reachable before enabling chat features. A 200 with the expected welcome string is positive confirmation that DNS, CORS, and ingress are all wired correctly.
- **Kubernetes / Docker liveness probe.** Unlike the main service's root, this one is unauthenticated, so a probe definition pointing at `/agent/api/` works without embedding service-principal credentials. (Prefer `/agent/api/health` when available, since it is purpose-built for that role.)
- **Ingress smoke test.** When deploying a new reverse proxy / ingress rule that maps `/agent/api/*` to the agent-ws container, a single `curl /agent/api/` confirms that (a) the rule matches, (b) `root_path="/agent/api"` is being respected, and (c) message comes back with the agent-ws welcome (not the main service's).
- **Operator banner.** An engineer poking at a new environment can hit the bare URL in a browser and instantly learn (i) the service is the agent-ws variant and (ii) `/docs` is the path to Swagger UI.

Anti-patterns

- **Don't use it for auth validation.** The handler has no `Depends(get_current_user)`; a 200 here proves nothing about JWT validity. For that, call the main service's `GET /` instead.
- **Don't parse the welcome string in clients.** The text is descriptive, not contractual; treat message as opaque except for unit tests that explicitly check it.
- **Don't use it as a deep readiness check.** The agent-ws integrates with several upstream services (Neo4j, LLM provider, etc.) — a 200 here says nothing about those. Use the health router for real readiness.

Observability tips

- The OBS-01 request-id middleware tags every call with an `X-Request-ID` response header. Capture it client-side if you need to correlate a failing probe with server logs.
- Access logging is configured via `configure_json_logging("eugene-agent-ws")` at line 49 of `eugene_chat_ws.py`. Filter logs by `path=/` to isolate root-route hits.
- If you need throttle telemetry, watch for `RateLimitExceeded` on this route — an unusual rate here often indicates a misconfigured probe interval.

5. Suggested debugging walk

- **Bring the agent-ws sidecar up.** `docker-compose up eugene_agent_ws` (or run `uvicorn eugene_chat_ws:app` from `agents/eugene-agent-ws/src/`). Confirm the boot sequence in the logs — line 49 prints the JSON logging banner, line 78 calls `add_routers(app)`.
- **Smoke the route directly.** `curl -s -i http://localhost:18510/`. Expect a 200 with body "Welcome to the EUGENE agentic chat webservice. Visit the Swagger docs at /docs". If you get a 404, check that `root_router` is still in the list at `eugene_chat_ws.py:35`.
- **Smoke it through the ingress.** `curl -s http://localhost:8080/agent/api/ | jq ..` If this 404s while the direct call succeeds, the problem is the reverse proxy strip rule, not the app; verify that `root_path="/agent/api"` matches what the ingress is stripping.
- **Set a breakpoint.** Put one at `agents/eugene-agent-ws/src/router/root_router.py:16`. Hit the route; if the breakpoint never fires while the client still gets a non-200, the request is being intercepted in middleware (rate limiter, CORS preflight, request-id) before it reaches the handler.

6. Reference index (file paths)

Router	
agents/eugene-agent-ws/src/router/root_router.py	(entire file, 18 lines)
App construction + wiring	
agents/eugene-agent-ws/src/eugene_chat_ws.py:32	(import root_router)
agents/eugene-agent-ws/src/eugene_chat_ws.py:35-37	(routers list + include loop)
agents/eugene-agent-ws/src/eugene_chat_ws.py:53-57	(FastAPI(root_path="/agent/api", ...))
Middleware stack (request still traverses these)	
agents/eugene-agent-ws/src/eugene_chat_ws.py:60-63	(SEC-03 rate limiter / SlowAPIMiddleware)
agents/eugene-agent-ws/src/eugene_chat_ws.py:66-73	(CORSMiddleware, EUGENE_CORS_ORIGINS)
agents/eugene-agent-ws/src/eugene_chat_ws.py:76	(OBS-01 request_id_middleware)
Related operational routes on the same sidecar	
agents/eugene-agent-ws/src/router/health_router.py	(deep health checks)
Sibling route on the MAIN service (distinct!)	
src/router/root_router.py	(auth-gated, returns user claims)

Key takeaways

- GET / on the agent-ws is a five-line, **unauthenticated** hello-world. No DB, no adapter, no mapper, no Depends, no response model.
- The route appears externally at /agent/api/ because of root_path="/agent/api" on the FastAPI constructor — not because the router declares any prefix.
- It is distinct from the main service's src/router/root_router.py: that one is JWT-gated and returns user claims; this one is anonymous and returns a single message string.
- Its job is operational: confirm the agent-ws process is up and the /agent/api ingress prefix is wired. It is safe to use as a Kubernetes liveness probe target.