

Agent platform: runtime and security model

Internal platform on which LLM agents execute business scenarios under an executable security policy.

Period: 07/2026 – present

Role: developer of the runtime and the security module in the platform team

Source code not public

BACKGROUND

Agent scenarios run for hours or days and pause while they wait for human approval. A worker restart must not cost any progress. At the same time, every call has to be traceable: who triggered it, which zone it runs in and which rule allowed it. Until then, the access model existed only as a document.

SOLUTION

- A scenario is described as a snapshot and compiled into a LangGraph graph that Temporal executes. Each node runs as a separate activity, so crashes and waits for approval do not abort the run.
- Compilation is a pure function: the same snapshot always yields the same graph. Otherwise Temporal fails with a non-determinism error when it replays the history.
- Graph state and history hold only references to artifacts, never the payload data.
- Errors fall into three classes. Transient errors are retried, fatal errors and policy denials are not, and business outcomes such as a data mismatch are an ordinary transition in the graph.
- The security module expresses zones, roles, data classes, autonomy levels and absolute prohibitions as Python code. `decide()` checks every call in seven steps in a fixed order and returns the rule it relied on.
- Java services query the module over HTTP with mTLS. No decision is issued without an entry in the append-only journal.

RESULTS

Status 09/2026, backed by code and tests

- Deterministic compilation is tested across process boundaries: the same snapshot gives the same hash in separate processes.
- Runtime: 176 tests in the standalone version, 58 tests in the platform repository.
- Security model in code: 8 zones with 10 permitted transitions, a 12×9 role matrix, 22 action types, 10 absolute prohibitions, autonomy levels L0–L3 with a kill switch.
- Security module: 145 tests, type checking with mypy in strict mode.
- Next step: connecting the security module to the core service of the platform.

TECHNOLOGIES

Python · Temporal · LangGraph · gRPC · HTTP with mTLS · Policy as code · mypy strict · pytest