
# MGOS Runtime Stack

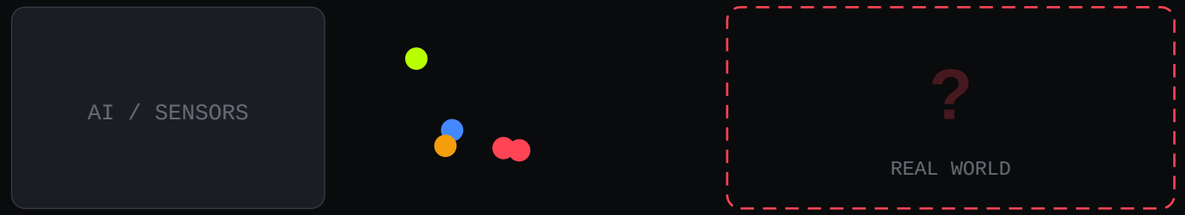
MGOS Integrity & Authorization Runtime

**AI proposes. Integrity stabilizes. MGOS authorizes. Evidence proves.**

THE PROBLEM

# Models generate proposals.
# Sensors generate observations.
# Neither should control execution.

Current guardrails are probabilistic, model-internal, and unauditable.
Between proposal and execution, the authorization boundary
is often fragmented, implicit, or unverifiable.



IMAGINE A FACTORY

# A robot gets orders from an AI.

The AI is brilliant -- it plans, optimizes, learns. But it also hallucinates, contradicts itself, and has no concept of consequence.

Between the proposal and the robot arm,
you need something that never guesses.

LAYER 1 -- INTEGRITY ENGINE

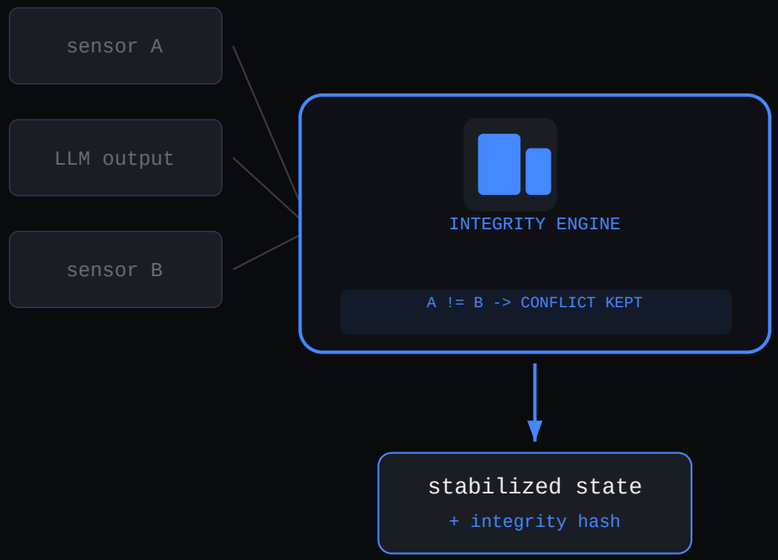
# Quality inspector at the gate.

Data arrives from everywhere --
sensors, models, databases.

It asks: can these inputs be stabilized
without collapsing genuine conflict
into false consistency?

If two sensors disagree,
it preserves the conflict as signal.

Only stabilized state moves on.



LAYER 2 -- DECISION KERNEL

# The guard at the boundary.

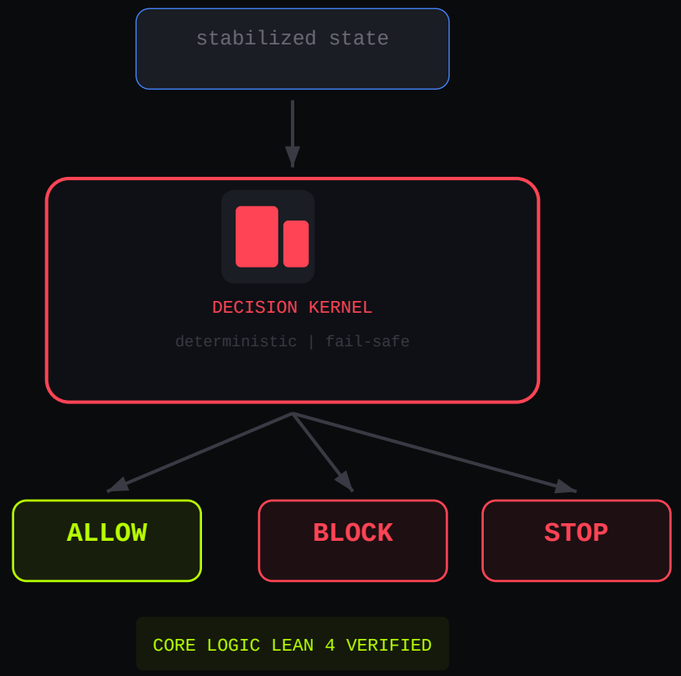
Receives the stabilized state
and answers exactly one question:

## Is execution allowed?

Three outcomes. No fourth option.
No 'maybe'. No inference.

Same state, same policy, same result.

Core decision logic formally verified in Lean 4.



LAYER 3 -- EVIDENCE

# The notary.

Every decision gets a cryptographic receipt.

SHA-256 hash. Manifest. Timestamp. Cryptographically verifiable.

If someone asks in a year why the robot stopped --
the proof exists.

LAYER 4 -- SENTINEL CONSOLE

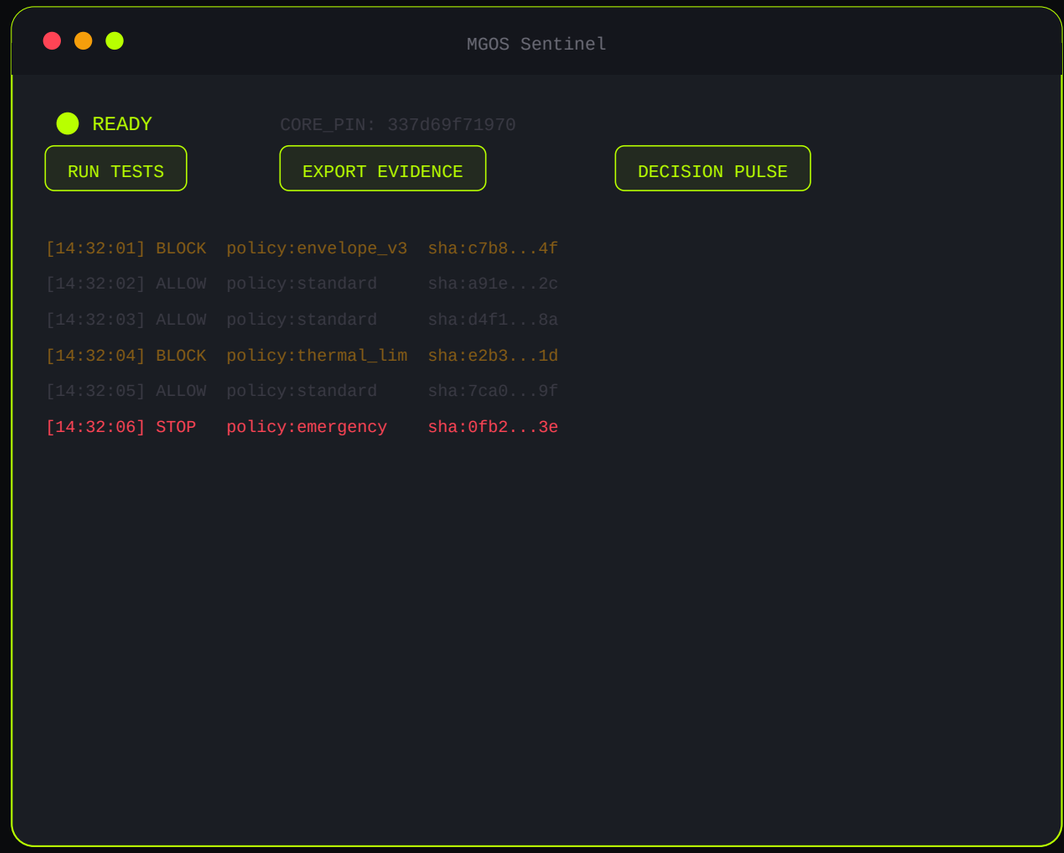
# The dashboard.

The human operator sees everything.

Every decision, every conflict, every receipt.

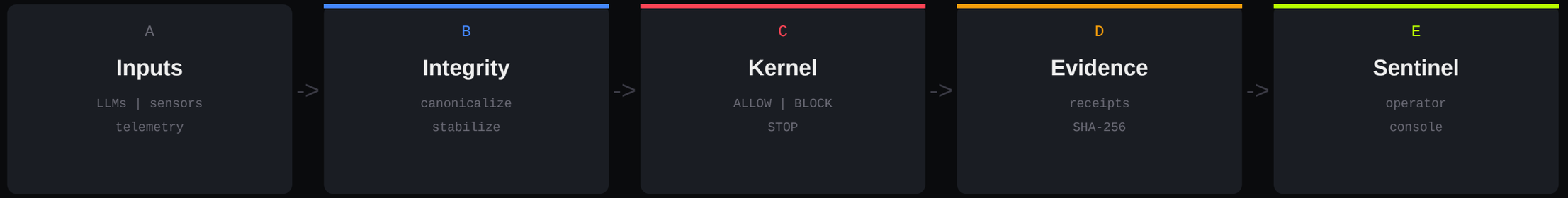
Run tests. Inspect decisions. Export evidence.

A runtime without operator visibility is operational risk.



THE COMPLETE RUNTIME

# One product. One path.
# Zero ambiguity.




Patent pending (PL/US) | Core logic Lean 4 verified | Deterministic | Fail-safe

WHAT IS PROVED. WHAT IS ENGINEERED. WHAT IS OPERATIONAL.

# Verified scope.

FORMALLY VERIFIED

## Core authorization logic proved in Lean 4:

- Fail-safe authorization
- Conflict safety
- Output exhaustiveness
- Permutation invariance (TMR)
- Manual-override dominance

Formal methods paper and artifact on request

ENGINEERED

## Implementation layer:

- Integrity pipeline
- Normalizer boundary (N1-N3)
- Evidence receipts + manifests
- Sentinel operator console
- SHA-256 evidence bundles

PO4 (normalizer) remains open

OPERATIONALLY VALIDATED

## Black-box test suites:

- Baseline hardtests: PASS
- Soak tests: PASS
- Poison-input tests: PASS
- Zero unsafe authorization
- Fail-close coverage = 1.0

In audited suites. Pinned env.

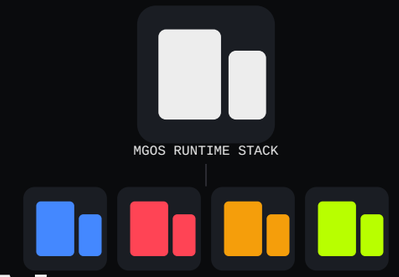
CORE PRINCIPLES

01 **We don't force data consistency.**

02 **Conflict is signal, not noise.**

03 **Only stabilized state reaches the decision.**

04 **Execution is deterministic and fail-safe.**




# AI proposes.
# Integrity stabilizes.
# MGOS authorizes.
# Evidence proves.

MGOS RUNTIME STACK | mgos.io

eval@mgos.io | +48 604 786 786

mgos.io/proof | mgos.io/runtime-stack | mgos.io/compare