Synchronicity – Architecture Brief

Public version. Pre-Action Authorization for autonomous AI.

The frame

A modern airport runs millions of decisions a day — every takeoff, taxi, and landing — without the pilots making those decisions themselves. A separate authority evaluates whether each proposed action is safe, legal, and consistent with the larger system, then signs off before the action happens. Pilots fly the plane. Controllers approve the move.

Autonomous AI needs the same separation. Models decide what to propose. Synchronicity evaluates whether the proposal complies with policy. Execution systems carry out only what was approved. The model doesn't grant its own permissions. The auditor doesn't trust the model's self-report. And every decision leaves a signed, replayable record.

What it is

Synchronicity is an external governance authority that runs in line with every autonomous AI action, before it executes. It is built on three architectural commitments:

The Canonical Responsibility Principle. Reasoning systems decide what to propose. The governance authority decides whether the proposal complies with policy. Execution systems decide how to carry out an approved action. Three responsibilities. Three components. No overlap, no delegation, no self-grants.

The non-execution invariant. The governance authority never executes. It only evaluates. Execution is the exclusive domain of execution systems, and execution systems are architecturally prevented from acting without a valid signed decision artifact from the authority.

Default-deny. On any failure — unparseable descriptor, missing policy snapshot, unavailable authority, malformed signature, exceeded evaluation budget — the response is DENY or HOLD. The unsafe state is unreachable under the specified threat model. Safety is the default state, not the exception path.

How it works

Every proposed action is canonicalized as a **Structured Proposed Action Descriptor (SPAD)** containing the actor, the action, the tool or capability invoked, the targets, the parameters, the input lineage, and a cryptographic identifier for the model that produced the proposal.

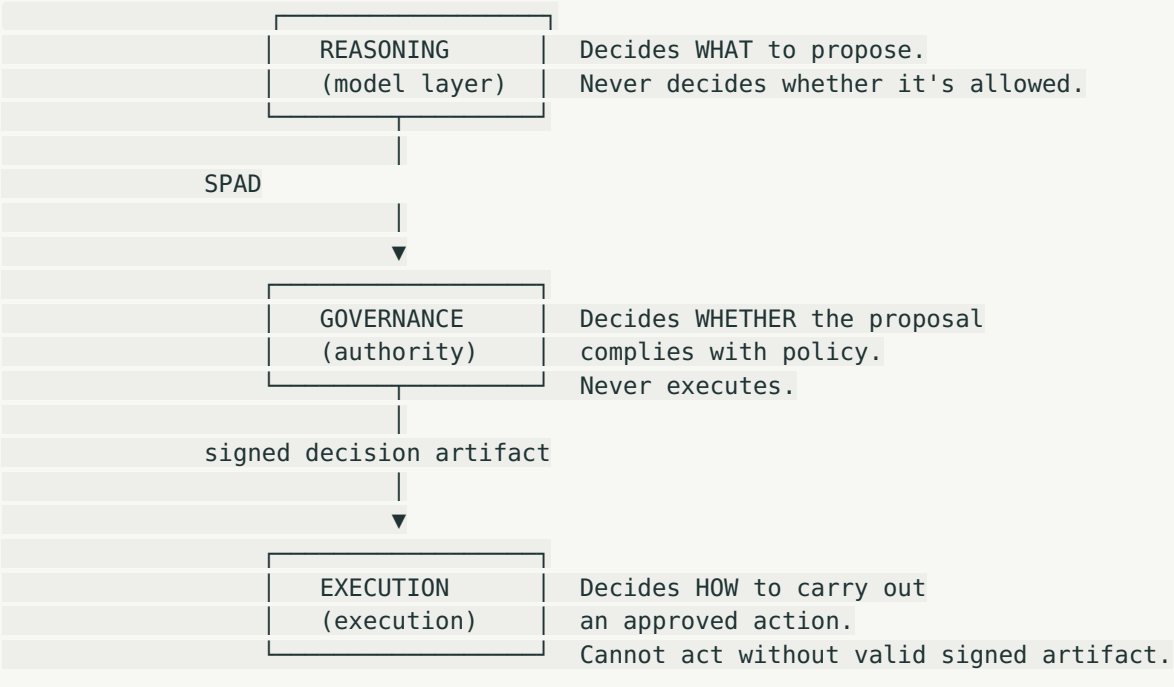
The descriptor is submitted to the **governance authority**, which evaluates it against a **versioned policy snapshot**. The policy is content-hashed and version-pinned; the exact policy in force at the moment of evaluation is recorded in the decision.

The authority returns one of three outcomes:

- **PERMIT** — the action satisfies all applicable policy. A signed decision artifact is produced. Execution may proceed.
- **DENY** — the action fails one or more policy rules. The signed artifact records which rules were violated. Execution must not proceed.
- **HOLD** — the action requires human approval, additional context, or out-of-band confirmation. Execution must not proceed until a subsequent PERMIT artifact is signed.

Every decision artifact — PERMIT, DENY, and HOLD alike — is written to a **unified append-only log**. The log is the evidence substrate that audit, bias detection, regulatory inquiry, and replay verification run on top of.

The Canonical Responsibility Principle, in one diagram



What's in a decision artifact

Each artifact is cryptographically signed and contains:

- The SPAD that was evaluated
- The outcome (PERMIT, DENY, HOLD)
- The policy version identifier and cryptographic content hash
- For DENY outcomes, the specific constraint violation identifiers
- A signed timestamp
- The actor binding
- The model identity that produced the proposal

This is enough to:

- Reproduce the decision deterministically by replaying it against the exact policy snapshot
- Detect post-hoc policy substitution (the hash will not match)
- Prove which rules governed the decision without disclosing the policy itself
- Build population-level analytics across thousands of decisions

Three integration patterns

The governance authority deploys as a single Go binary. Three integration patterns let you match assurance level to deployment risk:

Inline gateway. Every agent action passes through the authority before execution. Highest assurance. Mandatory for high-risk deployments.

Sidecar evaluation. The authority runs alongside the agent and is consulted by SDK or API. Lower friction, suitable for retrofitting existing systems.

Embedded library. The deterministic engine is linked directly into the host application for offline or edge use cases (SCADA, drone, defense, remote operation, agricultural autonomy).

Synchronicity is available for OEM and embedded integration with GRC, AI assurance, and security platforms.

What it does not do

To keep the scope honest:

- Synchronicity is not a model. It does not generate, classify, or filter model output. It governs the actions a model proposes.
- Synchronicity does not detect silent within-version model drift. That is the responsibility of the model-evaluation layer; Synchronicity is designed to consume those signals as descriptor metadata.
- Synchronicity does not replace organizational governance. It provides the technical control substrate. Documented processes, stakeholder engagement, audit evidence, and notified-body conformity assessment remain the deploying organization's responsibility.
- Synchronicity does not prevent out-of-band bypass via personal devices or personal API keys outside the governed network boundary. That is an identity and network-policy problem.

What it does claim, it claims rigorously.

About

Synchronicity is a product of Keystone Digital Holdings LLC. The founder is a contributor to OWASP AISVS (controls C9.7.7, C5.6.5, C5.6.6) and has submitted formal responses to the NIST AI Agent Security RFI (2026-00206) and NIST COSAiS. Patent applications pending. Patent counsel: Schwegman Lundberg & Woessner.

For an architecture briefing under NDA: boone.carlson@keystonedigitalholdingsgroup.com