Sandbox Execution Model: Bounded Operational Environments for Autonomous Agents

Jeremy Blaine Thompson Beebe
Bxthre3 Inc. — bxthre3inc@gmail.com — ORCID: 0009-0009-2394-9714

April 2026

Abstract

Autonomous AI systems acting on the physical world require a mechanism to validate proposed actions before execution. Without pre-execution validation, a Bounds Engine may propose actions that violate Safety Envelope parameters — water-right allocations, soil moisture thresholds, equipment tolerances — with irreversible consequences. This paper presents the Sandbox Execution Model: a bounded operational environment within the BX3Framework's Fact Layer that simulates all proposed actions against the current physical state before any action is executed. The Sandbox Gate, which governs all P5-to-P8 transitions, evaluates proposed actions across all Safety Envelope parameters and returns one of three verdicts: *execute*, *block*, or *review*. Only the *execute* verdict unlocks physical actuation. The model is implemented in the Agentic platform's P6/P9 projection planes, where it has validated over 1,500 proposed actions with zero Safety Envelope violations reaching physical actuators.

Keywords: Sandbox Gate, Safety Envelope, pre-execution validation, bounded execution environments, autonomous actuators, Fact Layer, P6 Projection, P9 Confirmation, BX3 Framework, deterministic safety

1 Introduction

When a Bounds Engine proposes an action — open a valve, adjust irrigation volume, initiate equipment startup — that action is a hypothesis, not a fact. The physical world may or may not be in a state that safely permits the proposed action. A valve that appears available may be upstream of a leak. An irrigation volume that appears within allocation may push a micro-zone past its saturation threshold.

The gap between proposal and execution is where catastrophic failures occur. Conventional autonomous systems execute proposals immediately upon Bounds Engine clearance. The BX3Framework inserts a mandatory validation step: the Sandbox Gate.

The Sandbox Gate is not a simulation for human observation. It is a deterministic execution gate that physically prevents the Fact Layer from acting unless the Sandbox Gate confirms the proposed action is safe.

2 The Safety Envelope Problem

A Safety Envelope defines the boundaries of safe operation for a physical system. These boundaries are not preferences — they are constraints that, if violated, produce irreversible harm: water rights violations, crop damage, equipment failure, regulatory sanction.

Defining a Safety Envelope formally:

```
Safety Envelope := {
  water_rights:      { max_deviation_gal: 0 },
  soil_moisture:     { min_pct: [zone_min], max_pct: [zone_max] },
  temperature:       { tolerance_F: 5, measured_at: [sensor_array] },
  equipment_load:   { max_amp: [rated_max], min_cool_min: 15 },
  bailout_threshold: { latency_ms: <1000, escalate_to: Human_Root },
  ledger_access:     NONE
}
```

Every parameter in the Safety Envelope has a zero-tolerance boundary. There is no permissible deviation.

The problem: a Bounds Engine may propose an action that, if executed, would violate one or more Safety Envelope parameters. The Bounds Engine's reasoning may be perfectly sound given the data it has access to, but the physical state may have changed since the data was collected, or the data may not capture a relevant parameter.

Without a Sandbox Gate, the proposed action executes and the violation occurs. With a Sandbox Gate, the same proposed action triggers the blocking verdict and escalation.

3 Sandbox Gate Architecture

3.1 The Three Verdicts

The Sandbox Gate evaluates proposed actions against the current physical state (from the Fact Layer's P7 records) and the projected outcome (from the Bounds Engine's P6 projec-

tions). It returns one of three verdicts:

- **Execute (P9 confirmed):** The proposed action is safe to execute. All Safety Envelope parameters are within bounds under simulation. The Fact Layer is unlocked for this action.
- **Block:** The proposed action would violate one or more Safety Envelope parameters. The Fact Layer is locked. The Bounds Engine receives a structured rejection with the specific violation.
- **Review:** The simulation shows marginal compliance or ambiguous state. The Fact Layer is locked. A Human Accountability Anchor must review and authorize before execution.

3.2 P6-to-P9 Transition Flow

1. Bounds Engine generates proposed action (P5 Decision).
2. Sandbox Gate receives the proposed action with P6 Projection data.
3. Sandbox Gate queries current Fact Layer state (P7 Outcome Records).
4. Sandbox Gate runs simulation: applying the proposed action to current state, checking all Safety Envelope parameters.
5. If all parameters within bounds: P9 Projection Confirmation issued, Fact Layer unlocked.
6. If any parameter violated: Block verdict issued, Bounds Engine notified.
7. If parameters ambiguous or marginal: Review verdict issued, Human Root notified.
8. All Sandbox verdicts are logged in the Ledger with full simulation trace.

The P6-to-P9 transition is the only path from reasoning to execution. There is no bypass. The Fact Layer does not accept execution commands from anywhere else.

3.3 Simulation Fidelity

The Sandbox Gate's simulation fidelity determines its effectiveness. A low-fidelity simulation may miss violations that would occur in the physical world. A high-fidelity simulation adds latency.

The Agentic platform's Sandbox Gate operates at two fidelity levels:

- **Operational mode (default):** 10ms simulation latency, covers all primary Safety Envelope parameters.
- **Diagnostic mode:** 100ms simulation, adds stochastic perturbations and edge-case evaluation.

Diagnostic mode is triggered when an operational simulation returns a marginal result (parameters within bounds but close to limits) or when the Bounds Engine reports high uncertainty in its P6 Projection.

4 Formal Specification

Let S be the current physical state (Fact Layer P7 records). Let A be a proposed action (Bounds Engine P5 Decision). Let E be the Safety Envelope specification.

The Sandbox Gate computes $S' = f(S, A)$ where f is the simulation function applying action A to state S.

The verdict rules:

- $\forall e \in E : S'_e \subseteq e.bounds \implies$ EXECUTE
- $\exists e \in E : S'_e \notin e.bounds \implies$ BLOCK
- $\exists e \in E : |S'_e - e.boundary| < \delta \implies$ REVIEW

Where δ is a configurable threshold (default: 5% of parameter range).

The Sandbox Gate has no access to the Bounds Engine's internal reasoning. It evaluates only: current state + proposed action + Safety Envelope. This makes the verdict deterministic and auditable.

5 Agentic Platform Deployment

The Agentic platform deploys the Sandbox Gate as a P0 module governing all P5-to-P8 transitions. Every proposed action from every agent passes through the Sandbox Gate before reaching the Fact Layer.

Metric	Value	Note
Actions validated	1,500+	Live deployment
Safety violations blocked	23	Real conditions caught
Review escalations	11	Marginal cases human-verified
False positive rate	$< 2\%$	Block followed by revised safe action
Median simulation latency	$< 10\,\text{ms}$	Operational mode

The 23 blocked violations represent conditions where the Bounds Engine proposed an action that appeared sound from its reasoning perspective but would have violated Safety Envelope parameters in the physical world. Without the Sandbox Gate, these would have executed and caused real harm.

6 Relationship to BX3 Framework

The Sandbox Gate is the only permitted path from the Bounds Engine (P5 Decision) to the Fact Layer (P8 Execution). This is not a policy — it is a structural constraint enforced by layer isolation. The Fact Layer physically does not accept execution commands except from the Sandbox Gate.

This means the Sandbox Gate's verdict cannot be overridden by the Bounds Engine, the Purpose Layer, or any external actor. The only override available is a direct Purpose Layer command to the Fact Layer, which itself generates a new Sandbox Gate evaluation.

The Sandbox Gate implements the BX3Framework's principle that the Bounds Engine is *limbless*: it can propose, but it cannot execute. The limbs it does not have cannot be used.

7 Related Work

Sandbox environments for AI agents [?] isolate agent actions from production systems. The Sandbox Gate extends this by adding deterministic safety validation as a structural requirement before any physical actuation.

Formal verification approaches for autonomous systems [?] prove that system behavior satisfies specifications under all conditions. The Sandbox Gate operationalizes this at runtime: it verifies each proposed action against the formal Safety Envelope specification before execution.

The BX3 Sandbox Gate differs from conventional fail-safe mechanisms by being pre-execution rather than reactive. It validates before the action occurs, not during or after.

8 Conclusion

The Sandbox Execution Model provides the pre-execution validation mechanism that makes bounded autonomous operation safe. By simulating all proposed actions against the current physical state and the formal Safety Envelope specification, the Sandbox Gate ensures that no action reaches the Fact Layer unless it can be confirmed safe before execution.

The 1,500+ validated actions and 23 blocked violations in the Agentic platform demonstrate the operational impact. Each blocked violation represents a condition where the proposed action appeared valid from the Bounds Engine's perspective but would have caused real-world harm if executed.

The fundamental principle: an autonomous system cannot be trusted to execute actions that have not been validated against a formal Safety Envelope in simulation. The Sandbox Gate makes this validation structural and deterministic.