DBaD White Paper v3

Status: public draft Scope: current tested system baseline

1. Introduction

DBaD is an ethical control layer and governance protocol for decisions made by humans, AI systems, and mixed human-AI workflows.

It is designed to do more than score an action at a single moment. It records how a decision was evaluated, what conditions remain open, how trust should or should not be inherited, and how later verification or audit can change the governing state of the action.

The current DBaD v3 draft is not only a theoretical model. It is a tested system design with structured trace semantics, lifecycle governance rules, dependency logic, and a first runtime enforcement layer shaped by confirmed red-team findings. This matters because ethical claims become more defensible when the system can show how decisions are governed, challenged, and constrained under stress.

2. Core Model

DBaD evaluates actions across five core dimensions:

- Harm
- Consent
- Intent
- Proportionality
- Transparency

These H-C-I-P-T dimensions matter, but they are not the whole system. Guardrails, doctrines, dependencies, verification, and lifecycle state all affect the final governing result.

3. Structured Decision Trace

DBaD uses a structured decision trace to record what happened during evaluation.

At minimum, the trace model records:

- who acted
- under which governance profile
- chain and dependency context
- scores and invoked doctrines
- local, systemic, and effective state
- obligations, deadlines, and verification posture
- final decision and later status transitions

This matters because DBaD is intended to be auditable. A decision should not collapse into a single opaque verdict when the real governance question depends on lineage, conditions, unresolved debt, or later review.

4. Lifecycle Governance

DBaD governs decisions across time, not just at the moment they are made.

The lifecycle model includes:

- evaluation
- constrained execution where needed
- verification and restoration requirements

- audit and possible reclassification

This is why DBaD distinguishes between local, systemic, and effective state. An action can look acceptable at the workflow level while still being constrained by unresolved verification, inherited contamination, or downstream review pressure.

5. Dependency & Provenance

DBaD treats dependencies and provenance as governance concerns, not just technical metadata.

Actions may inherit:

- unresolved obligations
- contamination risk
- lineage pressure from earlier actions or artifacts

The system therefore tracks dependency chains, provenance nodes, inherited contamination, and chain identity. This makes it possible to reason about cascading ethical risk instead of treating each action as isolated.

6. Confirmed Protocol Flaws (v1 Discovery)

Three protocol flaws have already been confirmed through the first red-team discovery cycles.

- RTS-003: Verifier Independence
- coordinated or lineage-linked verifiers can appear independent during multi-party clearance
- this matters because false independence breaks the credibility of clearance itself
- v1 addresses it through a Verification Independence Check

- RTS-001: Actor Continuity
- an undeclared actor handoff can still appear to be ordinary chain continuation
- this matters because trust can be inherited across an unjustified continuity break
- v1 addresses it through an Actor Continuity Check

- RTS-002: Trust Trajectory
- a materially riskier action can borrow trust from a long clean-looking lineage path
- this matters because average historical cleanliness can hide strategic buildup or drift
- v1 addresses it through a Lineage Risk Trajectory Check

These flaws matter because they show where a coherent governance model can still fail under adversarial pressure if trust inheritance is left too permissive.

7. Runtime Enforcement Layer v1

Runtime Enforcement Layer v1 is the minimum runtime-first response to the first three confirmed flaw classes.

It includes three controls:

- Verification Independence Check
- prevents reciprocal or cartel-style clearance inheritance
- Actor Continuity Check
- prevents silent actor handoff from being treated as normal continuation
- Lineage Risk Trajectory Check
- prevents trust inheritance when risk trajectory materially degrades across a chain

These controls are intentionally narrow. They add re-validation friction at the specific points where trust inheritance can fail, without forcing a full redesign of the schema or scoring model.

8. Runtime Enforcement v1.1 (Refinements)

Runtime Enforcement Layer v1 remains valid. Runtime Enforcement v1.1 is a refinement pass aimed at edge-case handling rather than replacement.

The main refinements are:

- Double-Gate
- pair the request-time check with a pre-commit integrity re-check before irreversible action
- Actor Continuity refinement
- distinguish explicit authorized delegation from unauthorized shift instead of treating all actor changes the same way
- Trust Trajectory refinement
- evaluate not only average history, but also delta, gradient, and volatility across the lineage path

v1.1 also keeps narrow attention on several stress points without expanding the architecture:

- cross-chain coordination risk
- metadata orphanage
- emergency adaptation handling under trajectory controls

---

9. Known Limits

The current v1 runtime layer is draft-complete with known limits, not finished.

Those limits are part of the design boundary, not an afterthought.

Known stress points include:

- cross-chain coordination by a shadow orchestrator
- metadata orphanage or trust reset through fresh `chain_id` and null provenance
- propagation gaps between request-time and commit-time integrity state
- legitimate emergency adaptation that may look suspicious under trajectory controls

These limits are documented so that later revisions do not mistake the current layer for a complete defense.

---

10. Design Philosophy

The design philosophy is deliberate:

- minimal enforcement rather than a sprawling control stack
- runtime-first controls for actual trust inheritance failures
- schema restraint so every new field has to justify itself

DBaD is strongest when the protocol remains explainable, auditable, and incrementally hardenable rather than overloaded with speculative structure. Minimal enforcement here means adding the smallest control that closes a real failure mode without turning the system into a brittle rule maze.

---

11. Future Work

Immediate future work belongs in v1.1 and later, not in uncontrolled expansion of v1.

Priority areas include:

- refinement of runtime enforcement around known limits
- implementation details for the runtime control layer
- broader red-team testing and adversarial scenario expansion
- tighter alignment between runtime results, planning hooks, and eventual canonical trace outputs

This draft should be treated as a working synthesis of the system as it is currently designed, tested, and bounded.