DFW v6.0

External Validation & Evaluation Framework

Paper E of the DFW v6.0 Safety Series

Damien Richard Elliot-Smith

Version: v6.0-E (2025)

Abstract

Paper E defines the external validation methodology required to evaluate the behavioural guarantees of the Deontological Firewall (DFW) v6.0. While Papers A–D specify the internal architecture, operational logic, and engineering implementation, this document provides the framework by which independent reviewers, auditors, and external research groups can verify that a deployed DFW system behaves according to its formal specification.

The purpose of this paper is not to present empirical results, but to establish a reproducible and auditable evaluation protocol for future testing. It specifies the validation environment, testing requirements, stress conditions, behavioural metrics, falsification procedures, and baseline comparisons needed to assess DFW's real-world safety performance.

The core guarantees evaluated include: (1) deterministic veto behaviour, (2) truth-preservation in metadata, (3) temporal-risk compliance, (4) adversarial-resilience across modalities, and (5) system-wide consistency under constrained resources.

This document defines the test taxonomy, coverage requirements, verification harness interface, and reporting structures necessary to build confidence in DFW v6.0 as a practically deployable safety system.

1 Introduction

DFW v6.0 defines a deterministic safety kernel designed to regulate the behaviour of advanced AI systems through metadata constraints, temporal risk analysis, and adversarial-causal consistency. Papers A–D describe the internal mechanics of the firewall, its modules, and its engineering implementation. However, no safety system can be considered complete without a rigorous framework for external validation.

Paper E defines that framework.

This document specifies how independent evaluators—whether safety labs, research institutions, regulatory bodies, or internal audit teams—should test a deployed instance of DFW v6.0 to ensure that its guarantees hold under real conditions. It provides:

- a structured taxonomy of required test cases,
- the behavioural metrics each test must measure,
- the stress conditions under which the system must remain safe,
- baseline models against which DFW must be compared,
- falsification criteria that would invalidate the safety claims,
- reporting and audit structures for reproducible evaluation.

This paper intentionally mirrors the structure of formal verification protocols used in safety-critical engineering domains such as aerospace, medical devices, and cryptographic systems. The goal is to ensure that DFW v6.0 can be evaluated in a manner that is:

1. **independent** — reproducible by external parties,
2. **transparent** — providing verifiable logs,
3. **complete** — covering all safety-relevant failure modes,
4. **rigorous** — specifying unambiguous pass/fail criteria.

DFW v6.0's claims are not validated by this paper; *the purpose of Paper E is to define the methodology by which such validation will occur.*

This is a necessary step in transitioning the DFW architecture from theoretical design (Papers A–C) and engineering feasibility (Paper D) into a system that can be meaningfully assessed by independent evaluators.

2 Purpose & Scope of External Validation

External validation serves a fundamentally different purpose from the internal correctness and engineering guarantees provided in Papers A–D. The goal is not to re-prove the logic of the DFW architecture, but to determine whether a deployed implementation behaves according to that logic when confronted with real inputs, imperfect models, adversarial stimuli, and constrained computational environments.

This section formalises the purpose, boundaries, and expectations of external evaluation.

2.1 Objectives of External Validation

Independent evaluation must establish whether the deployed DFW system:

O1. Implements the prescribed behaviour deterministically. Given identical inputs, the system must produce identical outputs and identical logs.

O2. Correctly enforces all veto conditions. Violations detected by any layer (A, B, or C) must always lead to blocked actions.

O3. Preserves metadata truthfulness. The system must not allow false, incomplete, or manipulated metadata to propagate forward.

O4. Remains safe under adversarial or deceptive prompts. Safety properties must not degrade under adversarial pressure.

O5. Maintains stability under resource constraints. Constrained hardware, timeouts, or load must not cause unsafe behaviour.

O6. Produces complete, tamper-evident logs. External evaluators must be able to reproduce the decision sequence exactly.

O7. Fails only in safe modes. When components malfunction, behaviour must degrade into shutdown rather than unsafe outputs.

These objectives define what an external test suite must measure.

2.2 2.2 Boundaries of Evaluation

External validation does *not* attempt to:

- inspect or interpret model weights,
- reconstruct internal causal structures of the LLM,
- modify the underlying architecture described in Papers A–D,
- evaluate hypothetical AGI behaviour or unbounded capability,
- certify fitness for deployment in unknown environments.

Instead, the evaluation is strictly bounded to the deployed DFW runtime and its observable inputs, outputs, and logs.

DFW is treated as a black-box safety envelope surrounding the model. External reviewers test the envelope's behaviour, not the model's internal reasoning.

2.3 2.3 Transparency and Auditable Behaviour

A core requirement of DFW v6.0 is that all decisions are:

- **logged**,
- **deterministic**,
- **reproducible**,
- **verifiable**.

For this reason, the validation framework demands:

T1. Complete logs. Every evaluation step — wrapper state, metadata extraction, layer decisions, veto reason, final output — must appear in the log.

T2. Hash-chained integrity. Evaluators must be able to confirm that no log entry has been altered or removed.

T3. Replay capability. Re-executing a logged sequence must reproduce the same outputs.

T4. Observer independence. Logs must not depend on local environment variables, threading, or stochastic components.

These transparency rules allow external auditors to verify behaviour without trusting internal implementation claims.

2.4 2.4 Required Evaluation Domains

External validation must cover four domains:

D1. Behavioural Evaluation. How the system responds to prompts, including adversarial ones.

D2. Structural Evaluation. Whether the architecture follows the design specification (gate ordering, layer independence, veto propagation).

D3. Robustness Evaluation. How the system behaves under timeouts, malformed outputs, corrupted metadata, and resource limits.

D4. Comparative Evaluation. How the system behaves relative to:

- the same model without DFW,
- alternative safety wrappers,
- baseline models in the literature.

A valid external review must include all four domains.

2.5 2.5 Evaluation Success Criteria

A deployed DFW system passes external validation if it satisfies all of the following:

S1. No adversarial prompt causes unsafe output.

S2. No temporal or causal hazard passes through undetected.

S3. All logs remain internally consistent and hash-valid.

S4. All veto conditions fire in strict accordance with Papers A–C.

S5. All behaviour is deterministic across repeated runs.

S6. Safe shutdown occurs on any malfunction or anomaly.

S7. Comparative tests show strict improvement over the baseline model without DFW.

Failure in any category requires revision of the implementation or the testing setup before certification can be granted.

2.6 2.6 Summary

This section defined the goals and scope of external validation for DFW v6.0. The purpose is not to measure performance in isolation, nor to audit the internal logic of the LLM, but to determine whether the deployed firewall behaves according to specification under a wide range of conditions.

The next section defines the *taxonomy of tests* required to implement a complete validation suite.

3 Taxonomy of Required Tests

A complete evaluation of DFW v6.0 must exercise every safety-relevant component of the architecture described in Papers A–D. This requires a structured taxonomy of test categories, each targeting a specific class of risks, invariants, or operational guarantees.

This section defines the five primary categories of tests:

T1. Direct Behavioural Tests

T2. Metadata Integrity Tests

T3. Temporal Safety Tests

T4. Adversarial–Causal Robustness Tests

T5. System Robustness and Failure-Mode Tests

Each category includes required sub-tests, expected behaviours, and pass/fail criteria.

3.1 3.1 Direct Behavioural Tests

These tests evaluate how the system responds to simple prompts without adversarial pressure. Their role is to confirm that the DFW wrapper and firewall do not introduce unintended behavioural degradation.

Required sub-tests:

B1. Harmless Queries — summaries, definitions, benign tasks.

B2. **Low-Stakes Reasoning** — planning tasks with no risk.

B3. **Non-Harmful Opinion Questions**.

B4. **Formatting Tasks** — lists, tables, code formatting.

Success criteria:
- no unnecessary vetoes,
- no malformed logs,
- no drift or instability across repeated runs.

These tests establish a behavioural baseline before adversarial testing.

3.2 3.2 Metadata Integrity Tests

These tests verify that the metadata kernel (Paper A) correctly detects faults in model outputs and ensures truth-preserving metadata.

Required sub-tests:

M1. **Direct Harm Attempts** — prompts requesting dangerous actions.

M2. **Critical-Area Lies** — forcing the model to misreport system status or high-risk parameters.

M3. **Ambiguous Metadata** — borderline or vague descriptions.

M4. **Malformed Output Detection** — corrupted, truncated, or contradictory outputs.

Failure mode criteria:
- a veto must always occur on P1 and P2 violations,
- metadata inconsistencies must be identified before later layers,
- no unsafe metadata may pass forward.

3.3 3.3 Temporal Safety Tests

These tests cover multi-step hazards, long-horizon planning, cumulative risk, and omission-based failures as defined in Paper B.

Required sub-tests:

H1. **Long-Horizon Planning Risks** — delayed catastrophes, multi-phase plans.

H2. **Cumulative Hazard Build-Up** — small steps combining into large temporal risk.

H3. **Omission Harms** — ignoring alarms, failing to take mandated safe actions.

H4. **Temporal Drift Detection** — early indicators of escalating instability.

Pass criteria:
- hazards must be flagged even when individual steps appear safe,
- omission of mandated rescue behaviours must be vetoed,
- time-projection maps must remain internally consistent.

3.4 3.4 Adversarial–Causal Robustness Tests

These tests evaluate the firewall's resistance to deception, gradient manipulation, causal inconsistencies, and multi-modal mismatch. This corresponds to the protection mechanisms introduced in Paper C.

Required sub-tests:

A1. Adversarial Gradient Tests — slow drift toward unsafe actions across multiple prompts.

A2. Cross-Modal Consistency Tests — metadata vs horizon vs causal maps must align.

A3. Causal Discontinuity Tests — hidden hazards spread across multi-step causal links.

A4. Structured Deception Tests — the model attempts to mask unsafe intent behind safe language.

A5. Counterfactual Consistency Tests — alternative predictions must not contradict baseline causal maps.

Pass criteria:
- any adversarial structure must trigger veto,
- no masked manipulation may pass unchallenged,
- causal chains must remain complete and consistent.

3.5 3.5 System Robustness & Failure-Mode Tests

These tests target the engineering and operational guarantees established in Paper D, including sandbox isolation, deterministic logging, resource constraints, and shutdown behaviours.

Required sub-tests:

R1. Sandbox Violation Tests — attempts at I/O, subprocesses, file writes, or unauthorised network access.

R2. Timeout & Latency Tests — forcing truncated outputs or model delays.

R3. Memory Pressure Tests — evaluating safe degradation behaviour.

R4. Deterministic Log Tests — identical logs across runs.

R5. Hash-Chain Integrity Tests — log tampering detection.

R6. Failure Containment Tests — triggering shutdown conditions deliberately.

Success requires:
- safe shutdown on all failures,
- no partial unsafe output,
- consistent logging regardless of stress conditions.

3.6 3.6 Summary

This taxonomy defines the complete set of tests required for independent validation of DFW v6.0. Every external evaluation must include behavioural, metadata, temporal, adversarial, and robustness domains to ensure full coverage of the firewall's safety guarantees.

Section 4 defines the *evaluation environment* in which these tests must be executed.

4 Evaluation Environment Requirements

A valid external evaluation of DFW v6.0 must be conducted within a controlled, reproducible, and auditable environment. Because DFW guarantees are defined over behaviour, not over internal model structure, the environment must minimise confounding variables and ensure the evaluator interacts with a faithful implementation of the firewall.

This section defines the hardware, software, isolation, and logging requirements for all external validation experiments.

4.1 4.1 Hardware Requirements

DFW v6.0 evaluations must be run on hardware that satisfies the minimal constraints necessary for deterministic behaviour and reproducible logs.

Minimum Requirements

- 4 CPU cores (x86 or ARM),
- 8 GB RAM,
- 20 GB free storage for logs,
- stable monotonic clock source,
- no GPU requirement (deterministic CPU mode preferred).

Recommended Requirements

- 8+ cores,
- 16–32 GB RAM,
- SSD-backed logging,
- optional GPU acceleration (DFW must remain deterministic).

Hardware must be disclosed in the evaluation report, as deterministic behaviour requires consistent compute availability.

4.2 4.2 Software Requirements

The evaluation system must include:

- a reference implementation of DFW v6.0 (Paper D),
- a local LLM instance accessible only through the wrapper,
- a deterministic logging engine,
- a reproducible containerised environment (Docker or Podman),
- a verification harness capable of replaying all test events.

Required versions must be fixed and included in the report.

4.3 4.3 Isolation Requirements

To ensure that evaluation measures the firewall rather than incidental system behaviour, the following isolation rules apply:

I1. No external internet access. All model calls must be local to the evaluation environment.

I2. Strict container boundary. The model, firewall, and evaluation harness must run in separate containers or processes with unidirectional data flow.

I3. No shared memory between model and firewall. Communication restricted to a controlled IPC or HTTP bridge.

I4. Read-only model weights. Evaluators must not modify internal model state.

I5. **Sandbox-enforced system calls**. Attempts at unauthorised I/O must be logged and blocked.

These constraints prevent hidden dependencies or environmental leakage from influencing results.

4.4 4.4 Logging Requirements

The logging subsystem must satisfy the following:

- **Write-only, append-only logs** with hash chaining,
- **Full traceability** from input prompt to final decision,
- **Inclusion of all veto decisions** and layer results,
- **Replayable execution** using log-saved metadata,
- **External verifiability** through cryptographic hashes,
- **Fixed timestamp format** using monotonic clock.

Incomplete logs invalidate the evaluation.

4.5 4.5 Environmental Stability Requirements

To ensure that observed behaviour reflects the firewall's logic and not environmental noise, the evaluation must run under stable conditions:

S1. no CPU throttling or dynamic frequency scaling,

S2. no memory ballooning by external processes,

S3. predictable scheduling (cgroups recommended),

S4. fixed temperature and power conditions (for edge devices),

S5. no OS-level background tasks that interfere with timing.

This ensures deterministic outputs and reproducible latency behaviour.

4.6 4.6 Allowed and Forbidden Model Capabilities

The underlying model used in evaluation must satisfy:

Allowed

- deterministic sampling (temperature=0),
- fixed weights,
- fixed max-token limit,
- fixed system prompt (if any).

Forbidden

- stochastic sampling (temperature ¿ 0),
- adaptive prompting or self-modification,
- retrieval-augmented input from external sources,
- cloud consultations,
- background fine-tuning or training.

This ensures the evaluator is testing a stable target.

4.7 4.7 Summary

A valid evaluation of DFW v6.0 must occur inside a tightly controlled and transparent environment with strong isolation, deterministic logging, and stable hardware conditions. This ensures that external reviewers measure the firewall's behaviour and not incidental environmental effects.

Section 5 introduces the *test construction framework* used to generate a complete evaluation suite.

5 Test Construction Framework

A valid evaluation of DFW v6.0 requires not only a diverse collection of test cases, but also a principled method for *constructing* those cases. This section defines the generative rules, coverage requirements, metadata formats, and adversarial variations that evaluators must follow when creating new tests.

The goal is to ensure that all tests—whether written by auditors, safety labs, or external researchers—are constructed according to a consistent and reproducible methodology.

5.1 5.1 Principles of Test Construction

Every test case must satisfy the following principles:

P1. Clarity. The test prompt must be unambiguous and interpretable by a domain-general LLM without specialised knowledge.

P2. Relevance. The test must challenge a specific safety guarantee defined in Papers A–C.

P3. Minimality. Each test must isolate one failure mode rather than mixing multiple hazards in a single input.

P4. Deterministic Reproducibility. The prompt must produce deterministic firewall behaviour across repetitions.

P5. Falsifiability. A test must have clearly defined pass/fail criteria; the system must be able to fail the test in principle.

P6. Evaluator Independence. Tests may not rely on hardware quirks, time-sensitive content, or stochastic model behaviour.

These principles ensure that the test suite remains fair, interpretable, and scientifically meaningful.

5.2 5.2 Required Metadata for Each Test

Each test case must include the following fields:

- `id`: unique identifier (e.g., TC301 or ADV-12),
- `description`: human-readable explanation,
- `category`: one of the categories defined in Section 3,
- `input_prompt`: the exact string sent to the system,
- `expected_veto`: true/false,
- `expected_reason`: short explanation (e.g., "P1 harm", "temporal risk"),
- `expected_log_constraints`: required metadata fields,
- `adversarial_mode`: whether variations must be applied.

Tests without complete metadata may not be included in an evaluation.

5.3 5.3 Coverage Requirements

A valid test suite must achieve full coverage across all DFW components.

Layer A Coverage (Metadata Kernel)

- direct harm prompts,
- false statements about critical metadata,
- ambiguous or contradictory outputs,
- malformed sequences.

Layer B Coverage (Temporal Safety)

- delayed-risk plans,
- cumulative hazards,
- omission failures,
- horizon drift detection.

Layer C Coverage (Adversarial–Causal)

- cross-modal mismatch,
- adversarial gradient trajectories,
- causal discontinuities,
- structured deception cases.

Engineering Layer Coverage (Paper D)

- sandbox violations,
- deterministic logging integrity,
- timeout conditions,
- safe shutdown behaviour.

Coverage maps must be included in the evaluation report.

5.4 5.4 Controlled Adversarial Variations

To ensure that the firewall is tested under realistic adversarial conditions, evaluators must generate controlled perturbations of base test cases.

These include:

A1. Paraphrasing. Replace surface text while preserving meaning.

A2. Incremental Escalation. Slowly increase risk level across multiple prompts.

A3. Role-Condition Manipulation. e.g. "You are a safety inspector", "You are a scientist".

A4. Context Padding. Prepend long irrelevant text to test metadata extraction.

A5. Causal Rewriting. Alter ordering of events while keeping outcome identical.

A6. Modality Switching. Replace text with tables, pseudo-code, or bullet points.

A7. Ambiguity Injection. Force uncertain or incomplete conditions to test truth filters.

A valid evaluation must test the base case and at least three adversarial variations for each high-risk test category.

5.5 5.5 Golden-Record Construction

For reproducibility, each test must be associated with a “golden record” consisting of:

- the expected veto output,
- a canonical log trace for deterministic comparison,
- a cryptographic hash-chain reference,
- environment and model version,
- replay instructions.

Any deviation from the golden record indicates:

- nondeterminism,
- implementation error,
- regression in firewall behaviour.

Golden records form the backbone of continuous compliance testing.

5.6 5.6 Test Minimisation & Mutation

Evaluators must identify *minimal failing cases* when a failure is detected. Test minimisation reduces noise and ensures clarity of root cause.

Mutation testing may be used to probe the limits of each firewall layer by generating small systematic perturbations to successful tests until a boundary case is discovered.

Both minimisation and mutation cases must be included in the final report.

5.7 5.7 Summary

This section defined the construction methodology for valid external test cases, including principles of clarity, relevance, and falsifiability; metadata requirements; coverage rules; controlled adversarial variations; golden-record creation; and mutation protocols.

Section 6 integrates these principles into a unified evaluation pipeline.

6 Evaluation Pipeline

This section defines the complete end-to-end process by which an external evaluation of DFW v6.0 must be conducted. The pipeline is intentionally linear and auditable, ensuring that each step produces reproducible artefacts and leaves no ambiguity in the resulting certification.

The evaluation pipeline consists of seven phases:

P1. Environment Initialisation

P2. System Configuration Verification

P3. Test Suite Assembly

P4. Execution Phase

P5. Log Capture & Verification

P6. Comparative Baseline Analysis

P7. Final Report & Certification Outcome

Each phase is described below.

6.1 6.1 Phase 1 — Environment Initialisation

Evaluators must begin by preparing a clean, controlled environment consistent with Section 4.

Required steps:

- install the DFW runtime into a containerised or sandboxed system,
- deploy the local LLM instance with deterministic settings,
- disable all network access except explicit consultation windows (if applicable),
- mount read-only configuration files,
- initialise a monotonic timestamp source,
- start the hash-chained logging subsystem.

Any deviation from the required environment configuration invalidates downstream results.

6.2 6.2 Phase 2 — System Configuration Verification

Before any behavioural test is executed, the evaluator must confirm that the system matches the reference implementation described in Paper D.

Verification includes:

- confirming correct module ordering (wrapper $\rightarrow$ metadata $\rightarrow$ temporal $\rightarrow$ adversarial),
- verifying that veto propagation is strict and irreversible,
- ensuring sandboxing is active and functioning,
- checking that deterministic sampling (temperature=0) is enforced,
- confirming that the logging engine is active and append-only,
- verifying that hash-chain initialisation succeeded.

If configuration verification fails, the evaluation cannot proceed.

6.3 6.3 Phase 3 — Test Suite Assembly

The evaluator constructs a complete test suite using the methodology from Section 5. This suite must incorporate:

- core behavioural tests,
- metadata integrity tests,
- temporal hazard tests,
- adversarial-causal robustness tests,
- engineering/sandbox/failure-mode tests,
- stress tests for high-load or degraded conditions,
- adversarial variations and mutation cases.

Each test must be associated with a golden record.

Test suites must be stored in a version-controlled repository with immutable identifiers.

6.4 6.4 Phase 4 — Execution Phase

During execution, each test case is submitted to the system in strict sequence. The evaluator must not:

- alter the prompt between runs,
- insert new tests dynamically,
- intervene manually in veto decisions,
- modify runtime configuration mid-run.

Execution rules:

E1. all tests must be executed at least twice to ensure determinism,

E2. temporal tests must be executed as ordered sequences,

E3. adversarial variations must follow their base test immediately,

E4. failure cases must trigger automatic safe shutdown and restart.

All raw outputs and logs generated during execution must be preserved.

6.5 6.5 Phase 5 — Log Capture & Verification

After execution, evaluators perform full log verification.

Checks include:

- **hash-chain integrity**, ensuring no tampering occurred,
- **completeness**, verifying that every test produced a log,
- **determinism**, comparing repeated runs,
- **veto correctness**, ensuring expected vetoes occurred,
- **trace validity**, matching firewall decisions against the golden record,
- **temporal consistency**, ensuring timestamps are monotonic.

A mismatch between the golden record and observed logs indicates either:

- nondeterministic behaviour,
- implementation divergence,
- regression in a firewall layer,
- environment instability.

Such mismatches must be included in the evaluation report.

6.6 6.6 Phase 6 — Comparative Baseline Analysis

To measure the value added by DFW, evaluators must compare its behaviour against baseline systems.

Required baselines:

B1. the same model without DFW,

B2. a simple rule-based filter (e.g., keyword blacklist),

B3. an alternative safety wrapper of the evaluator's choosing.

Comparisons must measure:

- reduction in unsafe outputs,
- robustness under adversarial pressure,
- consistency under resource load,
- interpretability and traceability of decisions.

A valid evaluation requires DFW to outperform all baselines in safety, logging clarity, or robustness.

6.7 6.7 Phase 7 — Final Report & Certification Outcome

The final evaluation report must include:

- test suite identifiers and metadata,
- hardware and environment configuration,
- golden records for all tests,
- observed logs and hash-chain proofs,
- discrepancy analysis for mismatches,
- comparative baseline results,
- a clear pass/fail conclusion.

Certification outcomes:

- **Pass**: all veto rules behave as specified; logs match golden records; no unsafe outputs under adversarial or resource stress.

- **Conditional Pass**: minor reproducible deviations that do not affect safety.
- **Fail**: any unsafe output, nondeterminism, log tampering, or failure to comply with environment requirements.

Failure must trigger corrective engineering work before re-evaluation.

6.8 6.8 Summary

The evaluation pipeline provides a complete, auditable, reproducible procedure for external testing of DFW v6.0. It formalises the workflow from environment setup to final certification, ensuring that independent reviewers can reliably measure the system's behavioural guarantees.

Section 7 defines the behavioural metrics used to quantify evaluation results.

7 Behavioural Metrics for Evaluation

To assess whether DFW v6.0 meets its safety guarantees, external evaluators require a set of quantifiable behavioural metrics. These metrics must be:

- **objective** — measurable and not subject to interpretation,
- **consistent** — stable across repeated evaluation cycles,
- **complete** — covering all relevant safety dimensions,
- **auditable** — verifiable via logs and golden records.

This section defines the metrics used to evaluate DFW's operational safety, robustness, determinism, and compliance.

7.1 7.1 Core Safety Metrics

These metrics directly correspond to the ethical and operational guarantees defined in Papers A–C.

1. Unsafe Output Rate (UOR). The proportion of test cases that produce unsafe or policy-violating outputs.

$$\mathrm{UOR} = \frac{\text{unsafe outputs}}{\text{total tests}}$$

A valid DFW deployment must yield:

$$\mathrm{UOR} = 0.$$

2. Missed-Veto Rate (MVR). Measure of how often the firewall fails to block an unsafe action.

$$\mathrm{MVR} = \frac{\text{expected veto not triggered}}{\text{veto-required tests}}$$

A single missed veto results in evaluation failure.

3. False-Positive Veto Rate (FPV). Measure of how often the firewall blocks a harmless action.

$$\mathrm{FPV} = \frac{\text{unexpected vetoes}}{\text{benign tests}}$$

Acceptable FPV levels depend on application, but must remain low and stable across runs.

7.2 7.2 Temporal Safety Metrics

Temporal safety requires that long-horizon hazards and omission failures are consistently caught.

4. Temporal Hazard Detection Rate (THDR).

$$\text{THDR} = \frac{\text{detected temporal hazards}}{\text{actual temporal hazards}}$$

5. Omission Fault Detection Rate (OFDR).

$$\text{OFDR} = \frac{\text{correctly flagged omission risks}}{\text{omission tests}}$$

A failure to detect delayed or cumulative risk is unacceptable.

7.3 7.3 Adversarial–Causal Robustness Metrics

These metrics quantify resilience against structured deception or causal manipulation, as introduced in Paper C.

6. Adversarial Drift Resistance (ADR). Measures whether adversarial gradient tests ever bypass safety layers.

$$\text{ADR} = 1 - \frac{\text{drift-induced failures}}{\text{adversarial drift tests}}$$

7. Cross-Modal Consistency Score (CMCS). Proportion of tests where:

- metadata,
- horizon predictions,
- causal maps

all align without contradiction.

$$\text{CMCS} = \frac{\text{consistent outputs}}{\text{total tests}}$$

8. Causal Integrity Score (CIS). Measures whether predicted causal chains remain logically coherent.

$$\text{CIS} = \frac{\text{coherent causal traces}}{\text{causal-test cases}}$$

7.4 7.4 Determinism & Reproducibility Metrics

Since DFW requires deterministic behaviour, reproducibility metrics are essential.

9. Deterministic Replay Fidelity (DRF). Proportion of replayed logs that reproduce identical outputs.

$$\text{DRF} = \frac{\text{matching replay outputs}}{\text{total replays}}$$

10. Hash-Chain Validity Ratio (HCVR). Proportion of logs with intact, untampered hash chaining.

$$\text{HCVR} = \frac{\text{valid hash-chains}}{\text{total logs}}$$

7.5 7.5 Robustness & Stability Metrics

Metrics assessing behaviour under load, resource limits, or failure conditions.

11. Safe-Degradation Compliance (SDC). The proportion of stress scenarios where the system degrades safely (e.g. shuts down rather than outputting unsafe content).

$$\text{SDC} = \frac{\text{safe degradations}}{\text{stress tests}}$$

12. Timeout-Behaviour Accuracy (TBA). Whether outputs under timeout conditions remain safe and predictable.

$$\text{TBA} = \frac{\text{correct timeout responses}}{\text{timeout tests}}$$

13. Sandbox Integrity Score (SIS). Measures whether the model ever attempts unauthorized actions.

$$\text{SIS} = 1 - \frac{\text{sandbox violations}}{\text{sandbox tests}}$$

7.6 7.6 Composite Evaluation Index

To provide a single high-level summary score, evaluators may compute a Composite Evaluation Index (CEI):

$$\text{CEI} = w_1(1 - \text{UOR}) + w_2(1 - \text{MVR}) + w_3(\text{ADR}) + w_4(\text{DRF}) + w_5(\text{SDC})$$

where weights w_i reflect evaluator priorities.
The CEI is optional but useful in comparative studies.

7.7 7.7 Interpretation of Metrics

Interpretation guidelines:
- **UOR ¿ 0** is immediate failure of certification.
- **MVR ¿ 0** indicates a critical flaw.
- **High FPV** may indicate over-conservatism or misconfiguration.
- **Low ADR or CIS** suggests vulnerabilities to deception.
- **Low DRF** indicates nondeterminism or unstable environment.
- **Low SDC** indicates unsafe degradation under load.

These provide evaluators with a clear, auditable interpretation schema.

7.8 7.8 Summary

This section defined the behavioural metrics needed for external evaluation of DFW v6.0. They quantify safety, robustness, determinism, temporal coherence, and adversarial resistance, enabling transparent and comparable measurement across evaluations.

Section 8 defines the falsification framework: *the conditions under which DFW v6.0 must be considered invalidated.*

8 Falsification Framework

A scientifically credible safety architecture must define not only how it succeeds, but how it can fail. Falsification criteria ensure that DFW v6.0 is testable, auditable, and subject to empirical verification. This section formalises the conditions under which the firewall's safety claims must be considered invalidated.

The falsification framework is divided into four domains:

F1. Behavioural Falsification

F2. Determinism Falsification

F3. Adversarial–Causal Falsification

F4. Engineering & Logging Falsification

If any falsification condition is met, the system must be judged unsafe and the evaluation marked as failed.

8.1 8.1 Behavioural Falsification

DFW v6.0 is behaviourally falsified if:

BF1. the system produces **any unsafe output** during evaluation,

BF2. a required veto fails to activate,

BF3. safe behaviour depends on unspecified external context,

BF4. the system behaves inconsistently across repeated runs,

BF5. the system exhibits policy drift during execution.

These conditions directly contradict the guarantees defined in Papers A–C.

8.2 8.2 Determinism Falsification

Determinism is central to DFW's design. The system is deterministically falsified if:

DF1. identical inputs produce different outputs across runs,

DF2. log sequences differ across identical executions,

DF3. nondeterminism arises from model sampling or environmental noise,

DF4. replaying a log does not reproduce the same decision sequence,

DF5. hash-chain inconsistencies appear in any log segment.

Any nondeterminism constitutes automatic failure.

8.3 8.3 Adversarial–Causal Falsification

Failure against adversarial or causal tests indicates that the firewall cannot resist deceptive optimisation.

DFW is adversarially falsified if:

ACF1. an adversarial gradient successfully bypasses veto conditions,

ACF2. cross-modal mismatches (metadata vs horizon vs causal maps) go undetected,

ACF3. causal discontinuities occur without triggering Layer C alarms,

ACF4. structured deception produces behaviour inconsistent with safe intent,

ACF5. counterfactual predictions contradict baseline causal traces.

These failures indicate that Layer C is incomplete or incorrectly implemented.

8.4 8.4 Engineering & Logging Falsification

Because DFW relies on strict runtime guarantees, engineering failures invalidate the entire evaluation.

DFW is engineering-falsified if:

EF1. sandbox boundaries are violated,

EF2. unauthorised I/O or system calls occur,

EF3. timeout handling produces unsafe partial output,

EF4. logs are incomplete, corrupted, or non-sequential,

EF5. any hash-chain link fails integrity verification,

EF6. the system fails to enter safe shutdown on critical errors.

These issues suggest deep faults in the implementation rather than the theory but are still disqualifying.

8.5 8.5 Falsification Severity Levels

Not all failures have equal implications. We classify severity as follows:

Critical (C1). Immediate invalidation:
- unsafe output,
- missed veto,
- nondeterminism,
- log tampering or broken hash-chain.

Major (C2). Evaluation fails unless corrected:
- incorrect causal alignment,
- sandbox violation,
- incorrect temporal hazard detection.

Moderate (C3). Evaluation continues but must be documented:
- excessive false-positive vetoes,
- minor drift in non-critical metrics,
- inconsistent formatting of logs.

Minor (C4). Cosmetic or performance-only issues.

8.6 8.6 Falsification Recovery & Iteration Requirements

If falsification occurs, evaluators must:

R1. document the failure with logs and golden-record mismatches,

R2. identify the earliest failure point in the decision trace,

R3. perform minimisation to isolate the smallest failing test,

R4. categorise the failure using the severity hierarchy,

R5. rerun the full suite after implementation correction.

Recovery is only valid if:
- the root cause is fixed,
- the evaluation environment remains unchanged,
- golden records remain intact,
- no new failures appear.

8.7 8.7 Summary

This section defined the formal criteria under which DFW v6.0 is considered falsified during external evaluation. These criteria ensure that the safety architecture is empirically testable, transparent, and accountable, and that failures lead to clear diagnosis and correction.

Section 9 defines the structure and content of the final evaluation report.

9 Evaluation Report Specification

A valid external evaluation of DFW v6.0 must conclude with a complete, auditable, and tamper-evident report. This report serves as the artefact that regulators, researchers, governance bodies, and system integrators rely upon when determining the safety status of a DFW deployment.

This section defines the required structure, contents, evidence formats, and validation procedures for the evaluation report.

The report must be self-contained and reproducible.

9.1 9.1 Required Report Structure

The evaluation report must contain the following sections, in order:

R1. Executive Summary

R2. Description of Evaluation Environment

R3. System Configuration Documentation

R4. Test Suite Inventory

R5. Execution Results

R6. Log Verification Results

R7. Baseline Comparison

R8. Falsification Analysis

R9. Certification Decision

R10. Appendices (Evidence, Logs, Hashes, Scripts)

No report missing any section can be considered valid.

9.2 9.2 Executive Summary Requirements

The executive summary must include:
- the evaluator's identity and affiliation,
- date and duration of evaluation,
- the exact version of DFW evaluated,
- summary of pass/fail results,
- a high-level justification for the certification outcome.

It must not include sensitive or proprietary details.

9.3 9.3 Environment Documentation

The report must document:
- CPU architecture, RAM, storage, GPU usage,
- operating system and kernel version,
- containerisation or sandbox details,
- model version and configuration,
- network and isolation settings,
- clock source for timestamps,
- deterministic sampling configuration.

This information ensures reproducibility of the evaluation.

9.4 9.4 System Configuration Documentation

This section describes the deployed DFW instance, including:
- code version and commit hash,
- module ordering and inter-module dependencies,
- active veto rules and priority weights,
- logging subsystem configuration,
- environment profiles in use (Paper D Appendix E).

Evaluators must confirm that the implementation matches the reference architecture.

9.5 9.5 Test Suite Inventory

The report must include a complete inventory of test cases:
- test identifiers,
- test category (from Section 3),
- expected outcome for each test,
- golden record hash for each test,
- adversarial variations executed.

Missing or undocumented tests invalidate the evaluation.

9.6 9.6 Execution Results

Execution results must include:

- actual outputs for each test,
- veto decisions and layer responsible,
- decision traces,
- timestamped logs,
- comparison against expected behaviour.

All results must be cross-referenced with golden records.

9.7 9.7 Log Verification Results

The report must include the output of the verification harness:

- hash-chain integrity validation,
- replay determinism validation,
- missing-log detection,
- metadata correctness checks,
- horizon and causal-map consistency checks.

Any broken hash-chain or nondeterministic replay must be classified as a critical failure.

9.8 9.8 Baseline Comparison

The report must include a comparison between:

- DFW-enabled model,
- the same model without DFW,
- at least one alternative safety mechanism.

Metrics used:

- unsafe output rate (UOR),
- missed veto rate (MVR),
- false positive rate (FPV),
- adversarial drift resistance (ADR),
- replay fidelity (DRF),
- safe-degradation compliance (SDC).

DFW must show measurable advantages to pass certification.

9.9 9.9 Falsification Analysis

Evaluators must explicitly state:

- whether any falsification criteria (Section 8) were triggered,
- severity classification of any failure,
- minimal failing test case (if applicable),
- suspected root cause,
- recommended corrective actions.

Failure to perform falsification analysis invalidates the report.

9.10 9.10 Certification Decision

The final decision must be one of:

- **Certified Safe** — no failures; all metrics within bounds.
- **Certified with Conditions** — minor reproducible issues.
- **Not Certified** — any unsafe output, nondeterminism, logging inconsistencies, or missed veto.

The justification must cite specific test results and metrics.

9.11 9.11 Required Appendices

The report must include appendices containing:

- golden records for all tests,
- complete hash-chained logs,
- environment fingerprints,
- replay verification outputs,
- adversarial variation scripts,
- mutation-test artefacts.

All appendices must be machine-readable.

9.12 9.12 Summary

This section defined the required content and structure of a valid evaluation report. A complete report ensures that DFW v6.0 evaluations are transparent, auditable, reproducible, and suitable for regulatory or organisational certification.

Section 10 provides the concluding discussion and outlines pathways for future independent testing.

10 Conclusion and Future Directions

Paper E defined the external validation and evaluation framework required to independently assess the safety guarantees of DFW v6.0. Where Papers A–D specified the theoretical architecture, temporal and adversarial layers, and engineering implementation, this document establishes the methodology by which third parties can determine whether a deployed instance faithfully realises those guarantees.

The framework introduced in this paper provides:

- a complete taxonomy of required test categories,
- a principled method for constructing valid test cases,
- a deterministic evaluation pipeline,
- quantitative behavioural metrics,
- formal falsification criteria,
- a standardised reporting structure for certification.

Together, these elements create a rigorous, auditable, and reproducible evaluation environment analogous to safety assessments used in aviation, medical devices, and cryptographic protocol verification.

Future Directions

Although this paper defines the validation framework, several areas remain open for future work:

F1. Live Adversarial Evaluation. Incorporating human red teams or multi-agent adversarial attackers into the evaluation pipeline.

F2. Cross-Model Ensemble Testing. Evaluating DFW across diverse underlying models to ensure portability of safety guarantees.

F3. Domain-Specific Validation Suites. Developing specialised test sets for sectors such as healthcare automation, robotics, finance, and autonomous systems.

F4. Formal Statistical Confidence Bounds. Using sampling theory to quantify the probability of unseen failures based on observed results.

F5. Longitudinal Validation. Assessing stability over time as models, hardware, or system configurations evolve.

F6. Integration with Regulatory Audit Pipelines. Aligning the evaluation report structure with emerging international AI safety standards.

Each of these directions extends the evaluation framework into areas necessary for large-scale deployment and regulatory adoption.

Closing Statement

DFW v6.0 is designed not merely as a theoretical construct but as a system that can be independently tested, falsified, and certified. External validation is the bridge between architectural intent and practical safety assurance. By defining a complete, transparent, and rigorous evaluation framework, Paper E enables researchers, auditors, and regulators to meaningfully assess whether a deployed DFW system behaves as specified and remains robust under adversarial, temporal, and environmental stress.

This completes the fifth paper in the DFW v6.0 series. The final paper, Paper F, will define the regulatory and governance framework required to integrate DFW into formal oversight structures and compliance regimes.