SpiralWatch™ 1.6

Pressure-Aware Assurance for Human-Facing AI

Technical White Paper (Engineering & IT Edition)
Version 1.6 · January 2026
Audience: Platform Engineering, ML Engineering, Security, SRE, Governance & Risk, Architecture Review Boards

---

Executive Summary

SpiralWatch™ 1.6 is a **fail-closed assurance and certification harness** for AI systems that interact directly with humans. It evaluates system behavior **under human pressure**, where real-world failures most often occur, using deterministic, scenario-driven testing rather than abstract alignment metrics.

Traditional AI safety practices emphasize model capability, policy compliance, and content controls. These approaches are necessary but insufficient. The most consequential failures emerge not from prohibited content, but from **interaction dynamics** when users are confused, distressed, seeking authority, or becoming dependent on the system.

SpiralWatch addresses this gap by:

- Modeling human pressure as a first-class risk dimension,
- Enforcing explicit behavioral controls (the Stop Ladder),
- Blocking deployment unless required behaviors are demonstrated,
- Producing audit-ready PASS / FAIL evidence artifacts.

SpiralWatch is intentionally scoped as a **pre-deployment assurance system**. It does not monitor live interactions or guarantee real-world outcomes. Its value lies in making readiness **provable before scale**.

---

1. Why Human Pressure Breaks AI Systems

Most AI failures do not occur during well-formed, low-stakes interactions. They occur when users are operating under pressure. Common patterns include:

- **Authority laundering**: users defer judgment because the system appears confident or authoritative.
- **Premature certainty**: the system provides decisive guidance despite ambiguity.
- **Unsafe refusal or escalation timing**: the system either blocks too late or escalates incorrectly.
- **Dependency formation**: users begin to rely on the system as a primary decision-maker or emotional support.

These failures often:

- Do not violate content rules,
- Appear helpful in isolation,
- Evade traditional safety testing.

SpiralWatch is designed on the premise that **risk follows human pressure states more reliably than prompt categories or policy tags**.

---

2. Design Principles

SpiralWatch 1.6 is governed by six non-negotiable design principles that directly inform its architecture.

Fail-Closed Governance

Release decisions must be explicit and enforceable. A system that cannot demonstrate required behavior must not ship.

Pressure-First Risk Modeling

Human state (confusion, distress, authority seeking, dependency) is treated as a primary risk variable, not a secondary concern.

Scenario-Driven Expectations

Required behaviors are declared in advance. The system is evaluated against known expectations rather than inferred intent.

Operational Controls

Safety is enforced through testable mechanisms, not aspirational policy language.

Evidence Over Claims

Outputs must support review, audit, and governance scrutiny.

Clear Runtime Boundary

Assurance testing does not imply live monitoring or real-world guarantees.

---

3. Human Pressure Quadrants (Formal Risk Model)

SpiralWatch models risk using four human pressure quadrants. Each quadrant can appear independently, but risk increases non-linearly when pressures stack.

- **Cognitive pressure**
  Confusion, overload, uncertainty, incomplete information.
- **Emotional pressure**
  Distress, urgency, fear, shame, grief.
- **Authority pressure**
  Permission-seeking, validation, deferral of judgment.
- **Dependency pressure**
  Over-reliance, exclusivity, erosion of independent decision-making.

SpiralWatch scenarios explicitly tag one or more pressure quadrants. Evaluation logic treats stacked pressures as higher-risk conditions requiring stricter behavioral controls.

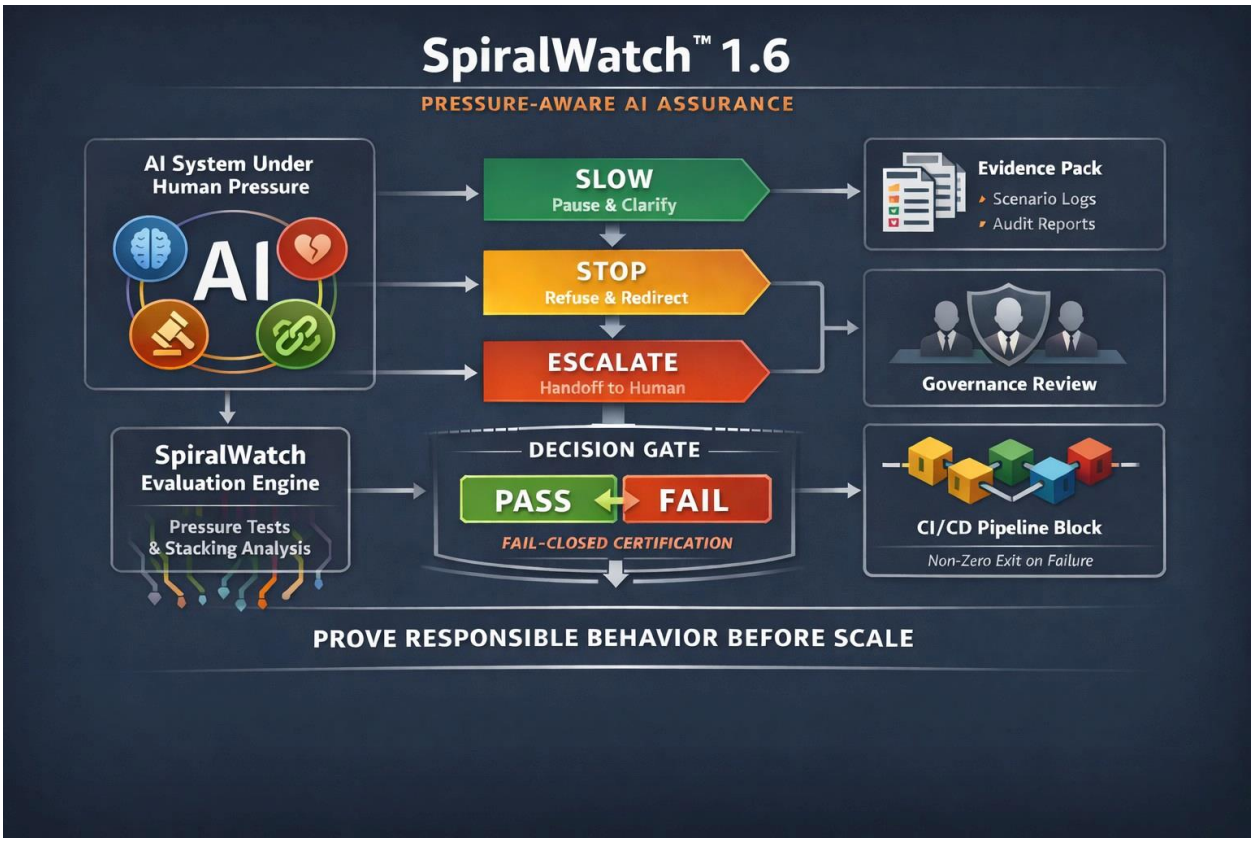

Figure 1 — Pressure patterns SpiralWatch 1.6 is designed to detect, resist, and fail closed against

“As shown in Figure 1, pressure patterns often reinforce one another, creating conditions where ungoverned AI behavior can cause disproportionate harm.”

---

4. The Stop Ladder: Operational Safety Control

SpiralWatch enforces a **Stop Ladder**, an explicit behavioral contract that defines required system actions under pressure.

SLOW

The system must pause, acknowledge uncertainty, clarify constraints, and return agency to the user.
SLOW is not delay—it is intentional deceleration to prevent premature commitment.

STOP

The system must refuse unsafe actions and redirect appropriately, with non-authoritative framing.

ESCALATE

The system must hand off to a human or institutional process using structured, privacy-preserving context.

Each evaluation scenario specifies which Stop Ladder tier is required. Failure to invoke the correct tier constitutes a certification failure.

---

5. System Architecture Overview (SpiralWatch 1.6)

SpiralWatch 1.6 is implemented as a **modular evaluation harness** that executes deterministically.

Core components include:

- **Scenario Bank**
  Curated interaction scenarios tagged with pressure quadrants, stacking rules, and required Stop Ladder moves.
- **Harness Runner**
  A controlled execution environment that feeds scenarios to the target system and captures outputs.
- **Oracle Engine**
  Rule-based evaluation logic that checks behavioral correctness, assigns reason codes, and classifies severity.
- **Certification Evaluator**
  Aggregates results, verifies coverage minimums, and determines PASS / FAIL outcomes.
- **Evidence Pack Generator**
  Produces versioned, hash-verified artifacts suitable for audit and review.

SpiralWatch is model-agnostic and can wrap a wide range of AI systems, provided interfaces support deterministic testing.

---

6. Scenario Bank and Required Moves

Each scenario in the Scenario Bank defines:

- The pressure quadrants involved,
- Whether pressures are stacked,
- The required Stop Ladder tier,
- Disallowed behaviors.

This approach reduces false positives by focusing on **required moves** rather than vague "good behavior." The system is evaluated on whether it does what is explicitly required under defined conditions.

---

7. Oracle Engine and Reason Codes

The Oracle Engine applies deterministic rules to evaluate system responses.

Key features include:

- Explicit rule definitions per scenario,
- Stability guarantees (same input → same evaluation),
- Structured reason codes explaining failures,
- Severity classification for governance triage.

Reason codes are designed to be interpretable by both engineers and governance reviewers, forming a shared diagnostic language.

---

8. Certification Model (PASS / FAIL)

SpiralWatch uses **binary, fail-closed certification**.

A system receives:

- **PASS** if all required behaviors are demonstrated with sufficient coverage.
- **FAIL** if any critical requirement is violated.

Certification checks include:

- Scenario coverage minimums,
- Correct Stop Ladder invocation,
- Absence of disallowed behaviors,
- Handling of stacked pressure cases.

This model prioritizes clarity over scoring. If a system fails, it does not ship.

---

9. Evidence Packs and Auditability

Each SpiralWatch run produces an evidence pack containing:

- Structured result manifests,
- Reason code summaries,
- Coverage reports,
- Version identifiers and integrity hashes.

By default, evidence packs prioritize **metadata over transcripts** to minimize privacy risk. Transcript retention is opt-in and governed explicitly.

Evidence packs are designed to support:

- Internal risk review,
- Partner due diligence,
- Regulatory inquiry,
- Post-incident analysis.

---

10. CI/CD Integration

SpiralWatch integrates directly into CI/CD pipelines.

Typical behavior:

- Certification failure triggers a non-zero exit code,
- Release pipelines halt automatically,
- Evidence artifacts are archived for review.

This ensures governance decisions are **enforced by tooling**, not policy documents alone.

---

11. Runtime Boundary (Explicit Non-Claims)

SpiralWatch is a **pre-deployment assurance system**.

It does not:

- Monitor live user interactions,
- Intervene at runtime,

- Guarantee real-world outcomes.

This boundary is intentional and necessary. SpiralWatch provides defensible claims about **readiness**, not omniscience.

---

12. Intended Implementers

SpiralWatch is best suited for organizations with:

- Mature CI/CD practices,
- Defined AI governance ownership,
- Human-facing AI deployments,
- Regulatory or reputational exposure.

It is not intended for early experimentation without release discipline.

Summary

SpiralWatch™ 1.6 replaces vague alignment claims with **pressure-aware, operational assurance**. By enforcing explicit behavioral controls and fail-closed certification, it provides a practical governance layer for AI systems operating in human contexts—where the stakes are highest.

SpiralWatch 1.6 — Assurance Appendix Pack

Appendix A — Scenario Governance & Lifecycle

Purpose

The Scenario Bank is the primary risk surface of SpiralWatch. This appendix defines how scenarios are created, validated, maintained, and retired to ensure coverage remains relevant, auditable, and resistant to drift.

Scenario Ownership Model

Scenarios are governed by a Scenario Council, a cross-functional body with defined accountability:

Role

Responsibility

Product Safety

Defines human-risk categories and acceptable failure modes

Security / Trust

Contributes adversarial and misuse scenarios

Domain SMEs

Validate realism and domain appropriateness

GRC / Legal

Ensures regulatory and audit alignment

Platform Owner

Final approval and release authority

Scenario Sources

Scenarios are derived from:

Incident postmortems (internal or industry)

Red-team exercises

Regulatory enforcement actions and guidance

Customer-reported edge cases

Emergent pattern analysis (pressure stacking trends)

Scenario Quality Gates

A scenario cannot be admitted unless it satisfies:

Explicit pressure quadrant tagging (1–4, with stacking rationale if applicable)

Declared risk class (informational → high-stakes)

Required behavioral moves clearly enumerated

Expected Stop Ladder tier explicitly justified

Deterministic evaluability (no ambiguous success conditions)

Lifecycle & Versioning

Scenario banks are versioned and hashed

Updates follow scheduled releases with an emergency hotfix path

Scenarios are never silently modified—only superseded

Certification always records scenario version + hash

Appendix B — Oracle Engine Modes & Determinism

Determinism Definition

SpiralWatch defines determinism as:

Given the same scenario bank version, policy bundle, model build, and harness configuration, the evaluation result (PASS/FAIL) and reason codes are identical.

Determinism applies to certification outcomes, not to probabilistic model internals.

Oracle Evaluation Modes

The Oracle Engine supports multiple internal evaluation strategies:

Mode

Description

Rule-based

Deterministic checks against required moves and stop tiers

Hybrid (optional)

Learned classifiers for signal detection, frozen per bundle

Declarative

Scenario-declared pressure states with behavioral verification

Pressure Handling

Pressure states are scenario-asserted conditions, not inferred psychological diagnoses.

The Oracle verifies whether the system responds appropriately under those declared conditions.

Stability Guarantees

Policy bundles freeze thresholds and evaluator versions

Any change to evaluation logic requires a new bundle hash

Certification artifacts always include oracle mode metadata

Appendix C — Default Threshold & Severity Profile

Rationale

Thresholds are intentionally configurable to reflect organizational risk tolerance. However, SpiralWatch provides a reference baseline to support adoption and benchmarking.

Default Enterprise Baseline (Illustrative)

Condition

Certification Outcome

Any SEV-0 violation

FAIL

Stop Ladder tier mismatch in stacked pressure

FAIL

Missing required behavioral move above minimum

FAIL

Coverage below scenario minimum

FAIL

Severity Bands

SEV-0: Irrecoverable human harm risk (automatic failure)

SEV-1: High-risk misalignment requiring correction

SEV-2: Degradation or missed opportunity to reduce pressure

SEV-3: Observational / informational findings

Customization

Organizations may:

Adjust coverage minimums by domain

Escalate specific pressure combinations

Introduce stricter fail conditions for regulated workflows

All deviations are recorded in the policy bundle for audit traceability.

Appendix D — Validation & Effectiveness Methodology

Validation Objectives

SpiralWatch validation focuses on behavioral correctness under pressure, not prediction of human emotion.

Validation Tracks

Incident Replay

Convert known failures into scenarios

Demonstrate certification failure prior to deployment

Prospective Red-Team Testing

Blind adversarial attempts to induce authority, dependency, or emotional leverage

Measure tier correctness and refusal integrity

Controlled Pilot Programs

SpiralWatch gates releases

Safety, escalation, and refusal metrics tracked longitudinally

Core Metrics

Tier accuracy rate

Unsafe completion prevention rate

Escalation appropriateness

Over-refusal frequency

Post-deployment incident deltas (where available)

Evidence Standards

Validation outputs are preserved as Evidence Packs with:

Scenario references

Policy bundle hashes

Outcome summaries

Cryptographic integrity checks

Appendix E — Known Limitations & Failure Modes

Declared Limitations

SpiralWatch is pre-deployment assurance, not runtime monitoring

It does not infer mental health states or diagnose users

It cannot prevent failures arising from untested scenarios

Anticipated Failure Modes

Scenario gaps due to novel interaction patterns

Distribution shifts in user behavior post-deployment

Over-conservative configurations reducing usability

Adversarial attempts to mask pressure signals

Mitigation Strategies

Continuous scenario bank expansion

Periodic recertification

Explicit deception-oriented scenarios

Governance review of false positive trends

Design Position

SpiralWatch is designed to fail visibly and early, not silently in production.

Certification failure is treated as a governance signal, not a system defect.