Audit Policies and Evaluation Criteria

Specification-Complete

Universal Stochastic Predictor (USP)

Document Version: 2.0
Last Updated: 2026-02-20
Audit Scope: USP v2.1.0-RC1

Abstract

Policy Strictness: ZERO-HEURISTICS (STRICT: 0% Fallback Tolerance)

This document defines the complete set of 36 audit policies derived exclusively from the authoritative specification corpus. All policies are strict, deterministic, and verifiable without subjective interpretation.

Contents

1 Source Specification Corpus (Authoritative)

All policies below are derived from the complete specification set in:

- `doc/latex/specification/Stochastic_Predictor_Theory.tex`
- `doc/latex/specification/Stochastic_Predictor_Implementation.tex`
- `doc/latex/specification/Stochastic_Predictor_IO.tex`
- `doc/latex/specification/Stochastic_Predictor_API_Python.tex`
- `doc/latex/specification/Stochastic_Predictor_Python.tex`
- `doc/latex/specification/Stochastic_Predictor_Test_Cases.tex`
- `doc/latex/specification/Stochastic_Predictor_Tests_Python.tex`

Critical Rule: No external sources are used. If a requirement does not appear in the specification corpus, it is not a policy.

2 Policy Index

3 Configuration and Validation Policies

3.1 Policy #1: Configuration Sourcing (Zero-Heuristics)

Source: `Stochastic_Predictor_IO.tex` (Configuration Mutation Protocol) and `Stochastic_Predictor_API`
(PredictorConfig)

Policy Statement: All numerical parameters, thresholds, and algorithmic decisions must be derived from configuration values. No implicit or silent defaults are allowed at runtime.

Audit Criteria:

- **Fail** if any operational threshold is hardcoded without config mapping.
- **Pass** if all `PredictorConfig` fields are mapped from config and validated.

3.2 Policy #2: Configuration Immutability (Locked Subsections)

Source: `Stochastic_Predictor_IO.tex` (Invariant Protection)

Policy Statement: The following subsections are immutable and must not appear in optimizer search space:

- `[core]`: `float_precision`, `jax_platform`
- `[io]`: `snapshot_path`, `telemetry_buffer_maxlen`, `credentials_vault_path`
- `[security]`: `telemetry_hash_interval_steps`, `snapshot_integrity_hash_algorithm`, `allowed_mutation_rate_per_hour`
- `[meta_optimization]`: `max_deep_tuning_iterations`, `checkpoint_path`, `mutation_protocol_versi`

Audit Criteria:

- **Fail** if locked parameters are mutated or included in search space.
- **Pass** if mutation validation rejects locked updates.

3.3 Policy #3: Validation Schema Enforcement

Source: `Stochastic_Predictor_IO.tex` (Validation Schema)

Policy Statement: All configuration mutations must pass strict schema validation: types, ranges, constraints, immutability, and interdependencies.

Mandatory Rules (Examples):

- `cusum_k` $\in [0.3, 1.5]$
- `dgm_width_size` $\in [32, 256]$ and power of 2
- `stiffness_low` $<$ `stiffness_high`
- `signature_depth` $\in \{3, 4, 5\}$

Audit Criteria:

- **Fail** if schema validation is bypassed or incomplete.
- **Pass** if validation rejects invalid updates with explicit errors.

3.4 Policy #4: Atomic Configuration Mutation Protocol

Source: `Stochastic_Predictor_IO.tex` (Atomic TOML Update Algorithm)

Policy Statement: Configuration mutations must follow the 5-phase POSIX atomic protocol:

1. Validation

2. Backup
3. Temp write with `O_EXCL` and `fsync`
4. `os.replace`
5. Audit log

Audit Criteria:

- **Fail** if any phase is missing (backup, fsync, replace, audit log).
- **Pass** only if all phases are implemented.

3.5 Policy #5: Mutation Rate Limiting and Rollback

Source: `Stochastic_Predictor_IO.tex` (Rate Limiting and Safety Guardrails)

Policy Statement:

- Max 10 mutations per hour
- Min 1000 steps between mutations
- Relative change $\leq 50\%$ (except discrete ± 1)
- Rollback if RMSE increases $> 30\%$ within 500 steps

Audit Criteria:

- **Fail** if any guardrail is missing or disabled.

4 Validation and Algorithmic Policies

4.1 Policy #6: Walk-Forward Validation (Causal)

Source: `Stochastic_Predictor_Implementation.tex` and `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: K-Fold and random splits are prohibited. Validation must be rolling walk-forward with future-only test windows and step size H.

Audit Criteria:

- **Fail** if any non-causal split exists.
- **Pass** only if train/test are strictly ordered and non-overlapping.

4.2 Policy #7: CUSUM Dynamic Threshold with Kurtosis

Source: `Stochastic_Predictor_Implementation.tex` and `Stochastic_Predictor_Python.tex`

Policy Statement: CUSUM threshold must be dynamic and kurtosis-adjusted:

$$h_t = k \cdot \sigma_{\text{resid}} \cdot (1 + \ln(\kappa_t/3))$$

with $k \in [3, 5]$ and rolling window size 252.

Audit Criteria:

- **Fail** if threshold is constant or missing kurtosis adjustment.
- **Pass** if dynamic formula and window are enforced.

4.3 Policy #8: Signature Depth Constraint (M in [3,5])

Source: `Stochastic_Predictor_Implementation.tex`

Policy Statement: Signature truncation depth M must be in $\{3, 4, 5\}$.

Audit Criteria:

- **Fail** if outside range is allowed.
- **Pass** if range enforced at config validation.

4.4 Policy #9: Sinkhorn Epsilon Bounds

Source: `Stochastic_Predictor_Implementation.tex` and `Stochastic_Predictor_Test_Cases.tex`

Policy Statement: Entropic regularization ε must satisfy $10^{-4} \leq \varepsilon \leq 10^{-1}$ and include underflow protection.

Audit Criteria:

- **Fail** if epsilon bounds are not validated.

4.5 Policy #10: CFL Condition for PIDE Schemes

Source: `Stochastic_Predictor_Implementation.tex`

Policy Statement: Time step must satisfy the generalized CFL condition:

$$\Delta t \leq \frac{C_{\text{safe}} \cdot (\Delta x)^2}{2 \cdot \sup |\sigma(x)|^2 + \sup |b(x)| \cdot \Delta x}$$

with $C_{\text{safe}} \approx 0.9$.

Audit Criteria:

- **Fail** if CFL validation is missing.

5 JAX and Numerical Precision Policies

5.1 Policy #11: 64-bit Precision Enablement

Source: `Stochastic_Predictor_Python.tex`

Policy Statement: `jax_enable_x64` must be enabled for Malliavin and signature computations.

Audit Criteria:

- **Fail** if global x64 is not enabled at startup.

5.2 Policy #12: Stop-Gradient for Diagnostics

Source: `Stochastic_Predictor_Python.tex`

Policy Statement: All diagnostic and telemetry computations must apply `jax.lax.stop_gradient`.

Audit Criteria:

- **Fail** if diagnostics do not use `stop_gradient`.

5.3 Policy #13: Kernel Purity and Statelessness

Source: `Stochastic_Predictor_Python.tex` (kernels are pure XLA kernels)

Policy Statement: Kernels must be pure and stateless functions compatible with JAX transformations.

Audit Criteria:

- **Fail** if kernels perform I/O or mutate external state.

6 Robustness and Circuit Breaker Policies

6.1 Policy #14: Frozen Signal Detection and Recovery

Source: `Stochastic_Predictor_IO.tex`

Policy Statement: If $N_{\text{freeze}} \geq 5$ identical values occur, detect frozen signal, emit event, freeze Kernel D, do not update weights, enable degraded mode. Recovery when variance $> 0.1 \cdot \text{Var}_{\text{historical}}$ for 2 steps.

Audit Criteria:

- **Fail** if any required action is missing.

6.2 Policy #15: Catastrophic Outlier Rejection (20 sigma)

Source: `Stochastic_Predictor_IO.tex`

Policy Statement: If $|y_t| > 20\sigma$, discard input, keep inertial state, emit critical alert.

6.3 Policy #16: Minimum Injection Frequency (Nyquist Soft Limit)

Source: `Stochastic_Predictor_IO.tex` and `Stochastic_Predictor_Test_Cases.tex`

Policy Statement: Minimum injection frequency must be enforced based on WTMM scales and C_{besov}. Violation freezes Kernel D and emits alert before Holder error exceeds 10%.

6.4 Policy #17: Staleness Policy and Degraded Mode Recovery (TTL)

Source: `Stochastic_Predictor_IO.tex` and `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: If $\text{TTL}(y_{\text{target}}) > \Delta_{\max}$, cancel JKO update, freeze weights, set degraded mode. Recovery only when $\text{TTL} < 0.8 \cdot \Delta_{\max}$.

7 Security and I/O Policies

7.1 Policy #18: Secret Injection via Environment Variables

Source: `Stochastic_Predictor_IO.tex`

Policy Statement: Credentials must be injected via environment variables. No secrets in source, config, or logs. `.env` must be gitignored.

7.2 Policy #19: Snapshot Integrity (SHA-256) and Validation

Source: `Stochastic_Predictor_IO.tex` and `Stochastic_Predictor_API_Python.tex`

Policy Statement: Snapshots must include SHA-256 hash and validation before load. Corrupt snapshots must be rejected with cold-start fallback.

7.3 Policy #20: Non-Blocking Telemetry and I/O

Source: `Stochastic_Predictor_IO.tex` and `Stochastic_Predictor_API_Python.tex`

Policy Statement: Telemetry and snapshot I/O must be non-blocking and decoupled from compute threads.

7.4 Policy #21: Hardware Parity Audit Hashes

Source: `Stochastic_Predictor_IO.tex`

Policy Statement: SHA-256 parity hashes of critical state must be logged at configurable intervals using canonical float64 serialization.

8 Adaptive and Emergency Mode Policies

8.1 Policy #22: Emergency Mode on Singularities (Holder Threshold)

Source: `Stochastic_Predictor_Theory.tex` and `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: If Holder exponent falls below $H_{\min}$, force Kernel D weight to 1.0 and switch to Huber cost until recovery hysteresis.

8.2 Policy #23: Entropy-Driven Capacity Expansion (DGM)

Source: `Stochastic_Predictor_Theory.tex` and `Stochastic_Predictor_Test_Cases.tex`

Policy Statement: Capacity must scale with entropy ratio κ using:

$$\log(WD) \geq \log(W_0 D_0) + \beta \log(\kappa), \quad \beta \in [0.5, 1.0]$$

8.3 Policy #24: Dynamic Sinkhorn Regularization Coupling

Source: `Stochastic_Predictor_Implementation.tex`

Policy Statement:

$$\varepsilon_t = \max(\varepsilon_{\min}, \varepsilon_0(1 + \alpha\sigma_t))$$

8.4 Policy #25: Entropy Window and Learning Rate Scaling (JKO)

Source: `Stochastic_Predictor_Implementation.tex`

Policy Statement: Entropy window and learning rate must scale with volatility:

- `entropy_window` $\geq c \cdot L^2/\text{ema\_variance}$
- `learning_rate` $< 2 \cdot \text{sinkhorn\_epsilon} \cdot \text{ema\_variance}$

8.5 Policy #26: Load Shedding (Kernel D Depth Set)

Source: `Stochastic_Predictor_API_Python.tex` and `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: Kernel D must support depth set $M \in \{2, 3, 5\}$ with warmup precompile to avoid JIT cache misses.

9 System Configuration Policies

9.1 Policy #27: Deterministic Execution and PRNG Configuration

Source: `Stochastic_Predictor_API_Python.tex`

Policy Statement: Deterministic PRNG and reduction settings must be configured before importing JAX.

9.2 Policy #28: Dependency Pinning (Exact Versions)

Source: `Stochastic_Predictor_Python.tex`

Policy Statement: All dependencies must be pinned to exact versions. Open ranges are forbidden.

9.3 Policy #29: Five-Layer Architecture Enforcement

Source: `Stochastic_Predictor_Python.tex`

Policy Statement: Implementations must use the five-layer structure: api, core, kernels, io, tests.

10 Snapshot and Meta-Optimization Policies

10.1 Policy #30: Snapshot Atomicity and Recovery (I/O)

Source: `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: Snapshots must be written to temp, fsync, and atomically renamed. Recovery must ignore torn temp files and load last valid snapshot.

10.2 Policy #31: Meta-Optimization Checkpoint Integrity

Source: `Stochastic_Predictor_API_Python.tex` and `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: TPE checkpoints must include SHA-256 sidecar validation and fail on mismatch.

10.3 Policy #32: TPE Resume Determinism

Source: `Stochastic_Predictor_Test_Cases.tex` and `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: Checkpoint resume must be bit-exact: best params, objective values, and trial history must match uninterrupted run.

10.4 Policy #33: Telemetry Flags and Alerts (Required Fields)

Source: `Stochastic_Predictor_API_Python.tex` and `Stochastic_Predictor_IO.tex`

Policy Statement: Telemetry must include `degraded_inference`, `emergency_mode`, `regime_change`, and `mode_collapse_warning` flags.

11 XLA and JAX Performance Policies

11.1 Policy #34: XLA No Host-Device Sync in Orchestrator

Source: `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: Orchestrator outputs must remain on device; no host synchronization in hot path.

11.2 Policy #35: Vectorized vmap Parity

Source: `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: Batched vmap execution must be bit-exact with sequential execution.

11.3 Policy #36: JIT Cache Warmup Guarantees

Source: `Stochastic_Predictor_Tests_Python.tex`

Policy Statement: Load shedding warmup must precompile all depths with cache hit rate $\geq 99\%$.

12 Audit Rule

Critical Enforcement:

Any policy violation results in audit failure. Only checks explicitly derived from the specification corpus are permitted.

12.1 Verification Protocol

1. Run `code_alignement.py` to validate all 36 policies
2. Verify 100% pass rate (36/36)
3. Fix VSCode errors before commit
4. Update LaTeX docs if public API changes
5. Stage, commit, push with meaningful message

12.2 Implementation Reference

All audit logic is implemented in:

- `tests/scripts/code_alignement.py` - 36 CODE_AUDIT_POLICIES
- `tests/scripts/scope_discovery.py` - Auto-discovery utilities