

# Input/Output Interface Specification - Universal Predictor System

Systems Architecture

February 19, 2026

## Contents

<b>1 Executive Summary</b>	<b>2</b>
<b>2 Configuration Vector (Hyper-Inputs)</b>	<b>2</b>
<b>3 Input Flow (Data Injection)</b>	<b>2</b>
3.1 1. Calibration Phase (Bootstrapping) . . . . .	2
3.2 2. Operational Phase (Online Stream) . . . . .	2
<b>4 Security Policies in the I/O Layer (Credentials)</b>	<b>3</b>
<b>5 System Outputs</b>	<b>3</b>
5.1 1. Control signal (Prediction signal) . . . . .	3
5.2 2. State telemetry . . . . .	4
<b>6 Abstract I/O Diagram</b>	<b>5</b>
6.1 Internal Process Cycle . . . . .	5
<b>7 Persistence (Snapshotting)</b>	<b>6</b>
7.1 Atomic and Verified Snapshotting Protocol . . . . .	6
<b>8 Telemetry and Deterministic Logging</b>	<b>6</b>
8.1 Non-Blocking Telemetry . . . . .	6
8.2 Parity Audit Hashes . . . . .	6

# 1 Executive Summary

This document defines the abstract input/output (I/O) interface for the Universal Predictor System, independent of the concrete implementation (Python/JAX, C++, Rust, FPGA). It describes the configuration vectors needed to instantiate the system, the runtime data flow, and the structure of the output signals and telemetry.

## 2 Configuration Vector (Hyper-Inputs)

The system is initialized with a configuration vector  $\Lambda$  that defines module topology and sensitivity. These parameters are typically static during an operating session or tuned by an external meta-optimizer.

Parameter	Symbol	Functional Description
Entropic regularization	$\epsilon$	Mass transport smoothing in the JKO orchestrator (Sinkhorn).
Learning rate	$\tau$	Adaptation speed of weights $\rho$ under energy gradients.
Signature depth	$L$	Truncation order of the log-signature (Kernel D - topological).
WTMM memory	$N_{buf}$	Sliding window size for singularity estimation.
Besov cone	$C_{besov}$	Influence radius for wavelet maxima tracking.
Holder threshold	$H_{min}$	Critical regularity that triggers the circuit breaker.
CUSUM threshold	$h$	Accumulated deviation level that triggers weight reset.
CUSUM slack	$k$	Drift tolerance ("white noise") allowed without error accumulation.
Volatility memory	$\alpha$	EMA decay rate to estimate error variance.

Table 1: Hyperparameter vector  $\Lambda$

## 3 Input Flow (Data Injection)

### 3.1 1. Calibration Phase (Bootstrapping)

Initial state required before sequential operation.

**Input:** History  $\mathcal{H} = \{y_{-T}, \dots, y_0\}$

- **Structure:** Time series of vectors  $\mathbf{y} \in \mathbb{R}^d$  or scalars  $y \in \mathbb{R}$ .
- **Purpose:**
  - Initialize history-dependent kernels (e.g., Levy parameters).
  - Stabilize initial orchestrator weights  $\rho_0$ .
  - Fill the singularity buffer for the WTMM module.

### 3.2 2. Operational Phase (Online Stream)

Step-by-step update cycle at time  $t$ .

**Input at step  $t$ :** Tuple  $(y_t, y_{target}, \tau_{epoch})$

- **Timestamp ( $\tau_{epoch}$ ):** Absolute timestamp (Unix nanoseconds). Required for synchronization and latency checks in the staleness policy.
- **Current observation ( $y_t$ ):**
  - New data point available at  $t$ .
  - Used to feed kernels ( $K_A, K_B, K_C, K_D$ ) and generate predictions for  $t + 1$ .
  - **Domain error handling:** If  $|y_t| > 20\sigma$  (relative to historical normalization), the point is classified as a catastrophic outlier. The system must discard the input, keep the inertial state, and emit a critical validation alert to protect kernels from numerical divergence.
  - **Frozen signal detection:** If the stream injects the exact same value for  $N_{freeze} \geq 5$  consecutive steps, the system must:

1. Compute the variance of the last  $N_{freeze}$  values:  $\text{Var}([y_{t-4}, y_{t-3}, y_{t-2}, y_{t-1}, y_t]) = 0$
  2. Identify this as sensor failure or data source corruption
  3. Emit `FrozenSignalAlarmEvent` with the event timestamp
  4. **Mathematical impact:** The Holder exponent in Branch D requires variability:  $H_t = \lim_{s \rightarrow 0} \frac{\log |\gamma(t+s) - \gamma(t)|}{\log s}$ . With a frozen signal, the numerator is zero, causing singularities or indeterminate values. This invalidates the multifractal spectrum. The system must:
    - \* Freeze the topological branch (Kernel D) at the last valid value
    - \* NOT update orchestrator weights (keep inertia)
    - \* Activate degraded inference mode
    - \* Continue predictions using Branches A, B, C only
  5. **Recovery:** Once variance  $> 0.1 \times \text{Var}_{historical}$  is detected for 2 consecutive steps, release the Kernel D lock and resume normal operation.
- **Inference grid (anti-aliasing input):**
    - **Sampling frequency vs scales ( $N_{buf}$ ):** To guarantee WTMM stability (Kernel D singularities), the data injection frequency must maintain sufficient density relative to the finest wavelet scales.
    - *Restriction:* A **minimum injection frequency** (Nyquist soft limit) is enforced based on  $C_{besov}$ . If event density falls below this threshold, the multifractal spectrum collapses and the system must freeze the topological branch update.
  - **Validation target ( $y_{target}$ ):**
    - The "real" value corresponding to the prediction generated at the previous step ( $t - 1$ ).
    - Typically  $y_{target} \equiv y_t$  for causal one-step-ahead prediction.
    - Used to compute error  $e_t = y_{target} - \hat{y}_{t|t-1}$  and the energy gradient  $\nabla E$  for JKO transport.
  - **Staleness policy:**
    - **Time-to-live (TTL):** Parameter  $\Delta_{max}$ .
    - **Violation behavior:** If the delay of  $y_{target}$  exceeds  $\Delta_{max}$ , the JKO update is canceled.
    - **Integrity signal:** The system must emit a persistent *degraded inference* flag ("stale weights"). This alerts the executor that, although the prediction  $\hat{y}$  is still produced, the weights  $\rho$  are stale and risk is no longer optimized geometrically.

## 4 Security Policies in the I/O Layer (Credentials)

The Stochastic Predictor design requires high-frequency ingestion against external market infrastructure (e.g., institutional WebSockets, brokers, or REST APIs). Access to these systems introduces critical vulnerability vectors.

### Secure environment injection

Hardcoding tokens, API keys, database secrets, or connection credentials in any future source module (e.g., `iostreams.py`) is strictly forbidden. Every implementation **must** apply an environment injection pattern, reading credentials at runtime from OS variables or local `.env` files.

### Version control exclusion

The resulting implementation repository must include explicit exclusion rules in version control (e.g., enforce `*.env` in `.gitignore`) to ensure no secret is exposed.

## 5 System Outputs

### 5.1 1. Control signal (Prediction signal)

Primary output for decision-making.

**Output:**  $\hat{y}_{t+1}$

- **Description:** Estimate of the expected process value for the next instant.
- **Inference grid (output quantization):**
  - Output  $\hat{y}_{t+1}$  is delivered in normalized space (Z-score) consistent with input  $y_t$ .
  - The actor/executor applies inverse normalization using rolling-window statistics if an absolute price is required.
- **Composition:** Convex combination of base kernels:  $\hat{y}_{t+1} = \sum_{i \in \{A, B, C, D\}} \rho_i^{(t)} \cdot K_i(y_t)$ .

## 5.2 2. State telemetry

Latent variables that describe system "health" and market regime.

- **Risk state ( $\mathbb{S}_{risk}$ ):**
  - **Local Holder exponent ( $H_t$ ):** Pointwise regularity measure.  $H_t < 0.5$  indicates antipersistence/noise;  $H_t < H_{min}$  indicates imminent crash/shock.
  - **Empirical kurtosis ( $\kappa_t$ ):** Fourth standardized moment of prediction residuals over a rolling window:
$$\kappa_t = \frac{E[(e_t - \mu_e)^4]}{(\sigma_e)^4}$$
where  $e_t = y_{target} - \hat{y}_{t|t-1}$  are prediction residuals.
    - \* **Purpose:** Validate the adaptive CUSUM threshold. Values  $\kappa_t > 3$  indicate leptokurtic distributions (heavy tails), activating the logarithmic adjustment  $h_t = k \cdot \sigma \cdot (1 + \ln(\kappa_t/3))$ .
    - \* **Interpretation:**  $\kappa_t \approx 3$  (Gaussian),  $\kappa_t \in [5, 10]$  (standard financial volatility regime),  $\kappa_t > 15$  (crisis regime with frequent extreme events).
    - \* **Alert:** If  $\kappa_t > 20$  persistently, emit a warning of potential residual model failure or undetected systematic outliers.
  - **DGM predictor entropy ( $H_{DGM}$ ):** Differential entropy of the neural value function  $V_\theta(t, x)$  across the spatial domain:

$$H_{DGM} = - \int p_V(v) \log p_V(v) dv$$

where  $p_V(v)$  is the empirical density of  $V_\theta$  values evaluated on a domain grid.

- \* **Purpose:** Monitor the health of Branch B (HJB solution via Deep Galerkin Method) and detect mode collapse when the network predicts a constant or degenerate solution.
- \* **Collapse threshold:** Compare against the terminal condition entropy:  $H_{DGM} \geq \gamma \cdot H[g]$  with  $\gamma \in [0.5, 1.0]$ . If the inequality is violated persistently (more than 10 consecutive steps), emit ModeDegradationAlert.
- \* **Corrective action:** Reduce the weight of Branch B in the orchestrator ( $\rho_B \rightarrow 0$ ) and prioritize alternative branches until the DGM network is retrained or reinitialized.
- \* **Note:** This indicator is only relevant if Branch B is active ( $\rho_B > 0.05$ ). For systems that do not use DGM, this field may be omitted or reported as NaN.
- **CUSUM statistic ( $G^+$ ):** Accumulated structural mismatch level.
- **Distance to collapse ( $h_t - G^+$ ):** Safety margin before a forced model reset. Note:  $h_t$  is now dynamic and depends on  $\sigma_t$  and  $\kappa_t$ .
- **Residual free energy ( $\mathcal{F}$ ):** Instant value of the JKO functional. It monitors whether the model is "stuck" in a stable local minimum or whether entropic regularization  $\epsilon$  is too high, over-smoothing mass transport and diluting predictive power.
- **Orchestrator state ( $\rho$ ):**
  - **Weight vector:**  $[\rho_A, \rho_B, \rho_C, \rho_D]$  such that  $\sum \rho = 1$ .
  - Indicates which "physics" currently dominates the market (jumps vs diffusion vs memory vs topology).

- **Stochastic health-check:**

- **Sinkhorn convergence (bool):** Indicates whether the mass transport algorithm converged within the maximum iteration count.
- *True*: Exact Wasserstein distance. *False*: Sub-optimal approximation (numerical precision alert).

- **Operational flags (mode and circuit breakers):**

- **Base operation mode:**

- \* *Standard (MSE)*: Normal operation under local Gaussian assumptions.
- \* *Robust (Huber)*: Defensive operation triggered by singularities ( $H_t < H_{min}$ ) or high volatility.

- **Degraded inference mode:** Critical boolean flag for temporal quality monitoring:

- \* **Activation condition:** It activates when the time-to-live (TTL) of  $y_{target}$  exceeds the maximum threshold:

$$\text{TTL}(y_{target}) = t_{current} - t_{signal} > \Delta_{max}$$

- \* **Operational implications:**

1. JKO transport update is suspended immediately
2. Weights  $\rho$  freeze at their last valid value (inertial mode)
3. Predictions  $\hat{y}_{t+1}$  continue, but are sub-optimal because they do not reflect the true market state
4. Risk is no longer optimized geometrically

- \* **Executor signaling:** This flag must explicitly warn that:

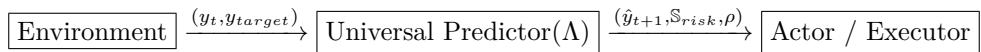
- Current predictions have *degraded confidence*
- Weights are stale
- Exposure should be reduced or a conservative mode should be used
- The system operates in "survival mode" until fresh data flow is restored

- \* **Recovery:** The flag is automatically cleared when a fresh signal arrives with  $\text{TTL}(y_{target}) < 0.8 \cdot \Delta_{max}$  (hysteresis threshold to avoid oscillation). At that moment, JKO transport resumes and `NormalOperationRestoredEvent` is emitted.

- **Emergency mode (singularity fallback):** Flag indicating whether emergency mode was triggered by critical singularity ( $H_t < H_{min}$ ), forcing  $w_D \rightarrow 1.0$  and switching to the Huber metric.

- **Regime change detected:** Flag indicating whether CUSUM detected a regime change at the last step, with entropy reset to a uniform distribution.

## 6 Abstract I/O Diagram



### 6.1 Internal Process Cycle

1. **Ingestion:** Receive  $y_t$ . Update local history.
2. **Singularity analysis:** Compute  $H_t$  using WTMM on the recent window.
3. **Quality control (CUSUM):**
  - Compute error  $e_t$  using  $y_{target}$  and stored prediction  $\hat{y}_{t|t-1}$ .
  - Update drift accumulator  $G^+$ .
  - If  $G^+ > h$  or  $H_t < H_{min} \rightarrow$  emit reset/alert signal.
4. **Transport (JKO):**

- Compute energy gradient  $\nabla E$  based on  $e_t$ .
- Transport probability mass  $\rho_{t-1} \rightarrow \rho_t$  (Sinkhorn).

### 5. Projection:

- Execute kernels  $K_i(y_t)$  to obtain components.
- Aggregate components using new weights  $\rho_t$  to obtain  $\hat{y}_{t+1}$ .

## 7 Persistence (Snapshotting)

To ensure operational continuity, the system must be able to serialize its full internal state  $\Sigma_t$  at any time  $t$ .

$$\Sigma_t = \{\rho_t, G_t^+, \sigma_{ema}^2, \kappa_t, H_{DGM}, \text{Flags}, \text{WTMMBuffer}, \text{KernelsState}\}$$

where:

- $\kappa_t$ : Rolling empirical kurtosis of prediction errors (window size = 252).
- $H_{DGM}$ : Differential entropy of the DGM predictor (for mode collapse detection).
- **Flags**: Boolean flags including `DegradedInferenceMode`, `EmergencyMode`, and `RegimeChangeDetected`.

The `KernelsState` structure must be segmented into independent sub-blocks (K-blocks) to allow modular or partial updates:

$$\text{KernelsState} = \{S_A(\text{Levy}), S_B(\text{PDE}), S_C(\text{Memory}), S_D(\text{Topology})\}$$

The restore operation  $Load(\Sigma_t)$  must allow the flow to resume at  $t + 1$  without recalibration over history  $\mathcal{H}$ . Correct restoration of  $\kappa_t$  and  $H_{DGM}$  is critical to preserve anomaly detection and mode collapse sensitivity after a restart.

### 7.1 Atomic and Verified Snapshotting Protocol

Binary serialization formats (e.g., Protocol Buffers, MessagePack) are required instead of text (JSON/XML) to minimize I/O latency for critical hot-start operations.

- **Mandatory integrity checksum**: Because dense binary formats are used, a single-bit error in kernel matrices or the WTMM buffer could trigger system collapse or undefined behavior. Therefore,  $\Sigma_t$  must include a robust validation hash (e.g., SHA-256 or CRC32c) at the end of the block.
- **Pre-injection validation**: The restore routine  $Load(\Sigma_t)$  must recompute and verify this hash *before* injecting the state into operational memory. If validation fails, the snapshot must be discarded and the system must restart in cold-start mode (history reload).

## 8 Telemetry and Deterministic Logging

### 8.1 Non-Blocking Telemetry

Telemetry emission must be decoupled from JAX/XLA execution. Compute threads must never block on I/O. Telemetry buffers are enqueued in a non-blocking structure and consumed by a separate process or thread.

### 8.2 Parity Audit Hashes

For hardware parity validation, log SHA-256 hashes of the weight vector  $\rho$  and the OT cost at configurable intervals. Hash inputs must use canonical float64 serialization to guarantee deterministic parity across CPU and GPU.

- Hash interval is configuration-driven.
- Logs are append-only and immutable.
- No raw signals or secrets are emitted in telemetry.