ZIRA

Proof of Resonance

A Decentralized Protocol for Real-Time Value Perception in the Autonomous Machine Economy

Document Type: Technical Whitepaper

Version: 1.0

Release Date: May 2026

Network Domain: zira.network

ABSTRACT

Every distributed ledger trying to interface with real-world assets runs into a fundamental wall: a system without a centralized authority cannot natively ascertain the market value of external assets. Traditional networks get around this limitation by implementing external oracle layers. This approach patches the systemic blindness but introduces trusted human intermediaries, running counter to the underlying rationale for building decentralized systems in the first place.

Proof of Resonance (PoR) sidesteps this tradeoff entirely by turning economic perception into an inherent feature of core consensus. The identical set of network validators tasked with confirming ledger state and appending new data tracking logs continuously measure asset pricing profiles. Influence over both state settlement and pricing calculations originates from a single, un-purchasable, earned reputation ledger score: the ZIRA Trust Index (ZTI). Under this layout, no disconnected oracle network exists; the core ledger operates as its own oracle environment.

This technical document outlines the formal algorithmic structure of the PoR system, verifies its resilience against coordinated structural exploits, contrasts it with existing data ingestion methods, and details why this unified architecture is essential for the burgeoning machine economy, where autonomous AI systems trade resource assets at speed without human management.

1. The Oracle Problem

Blockchains are structural vacuums, blind to anything occurring outside their local transaction logs. A standard distributed database can easily track the movement of a digital asset between two addresses, yet it remains blind to the immediate real-world purchasing power or localized spot valuation of that asset. Bridging this disconnect requires networks to bring in external data feeds tracking physical property indexes, market activity, or volatile overhead dynamics such as real-time computational energy demands.

This disconnect represents the classic oracle dilemma. Current systems use a fragile workaround: they insert a separate tier of middlemen to watch external environments and relay data snapshots onto the chain. These systems take various forms, from standalone node infrastructure networks to corporate trading consortia or asset staking pools. Regardless of the exact shape, they all place an administrative selection filter at the center of the protocol, deciding which actors get data reporting rights and why their specific inputs should be accepted.

Relying on administrative selection introduces structural risk. Human coordination delays and governance cycles are workable in traditional finance, but they fall apart entirely under the fast operational requirements of a machine-driven economy.

When independent software entities trade at sub-second speeds, scaling up compute pipelines by the gigaFLOP, re-routing communication paths, and checking energy inputs, they need instant and authentic value parameters. These entities cannot wait for an external oracle system to finish its update cycle. They cannot expose their logic to price skew or front-running bots waiting to exploit stale data feeds. In a machine network, the oracle problem is not an optimization goal; it is a critical threat to system survival.

1.1 Why Existing Approaches Fall Short

The structural problems with standard oracle platforms are deep and cannot be fixed with minor updates:

- **Reactive Latency:** Snapshot oracles update only after hitting specific price thresholds or when explicit logs request a refresh. By the time a value is stored on the ledger, the true market state has moved on. While humans barely notice a multi-second pause, this window is an eternity for autonomous software, giving adversarial arbitrage bots plenty of space to extract value from the protocol.
- **Capital Asymmetry:** Stake-weighted data models try to stop Sybil exploits by forcing participants to lock up upfront capital. But capital can be borrowed or accumulated. An entity with enough funding can buy up the necessary token pool, distort data feeds during their assignment window, capture downstream value, and exit their position. The fixed cost of pulling off this attack remains bounded and predictable, reducing ledger security to a basic financial calculation.
- **Architectural Decoupling:** Modern protocols isolate price tracking from fundamental state consensus. Since block validators have no connection to data observers, a fragmented interface appears. This gap leaves smart contracts open to exploitation vectors that can only be fixed by combining transaction confirmation and asset perception into one mechanism.

Proof of Resonance fixes this structural decoupling by making value perception a native component of network consensus.

2. The Physics of Resonance

In 1665, Christiaan Huygens noted an unusual mechanical event while observing two separate pendulum clocks mounted to a shared wooden beam. Regardless of their starting points or small

internal differences in manufacturing tolerances, the two pendulums slowly altered their rhythm until they locked into a perfectly synchronized swing.

The timepieces lacked connecting gears and sent no explicit signals back and forth. The shared mounting board acted as a physical medium, transferring tiny mechanical pulses between the devices. These minor reciprocal impacts pushed each pendulum away from its isolated movement pattern until they converged on a matching global speed. Today, physical sciences identify this spontaneous ordering behavior as entrainment.

This systemic convergence occurs regularly across nature, from neurons firing together across brain regions to biological systems adjusting to cyclical routines and laser configurations aligning light particles into coherent waves. The primary math describing these coupled systems is found in the Kuramoto model, which tracks the behavior of a population of N connected oscillators:

$$d\theta_i / dt = \omega_i + (K / N) \cdot \Sigma_j \sin(\theta_j - \theta_i)$$

Here, θ_i represents the specific state phase of oscillator i, ω_i is its natural individual speed, and K acts as the system coupling strength factor. When the coupling intensity K crosses a mathematically precise threshold value (K_c), the units stop acting randomly and align to a single, unified frequency.

Proof of Resonance applies this exact mechanical concept to decentralized value tracking. In this model, each validator node acts as an independent oscillator. The individual frequency (ω_i) represents the node's local estimation of an asset's worth, built by analyzing its own data feeds, processing models, and local history.

The baseline peer messaging system acts as the structural mounting beam, while the ZTI metric serves as the explicit coupling strength variable (K). The final Resonant Value, the confirmed price recorded during synchronization, appears naturally whenever the active network operates above its critical threshold.

This structure is a precise mathematical implementation. The convergence paths within Proof of Resonance come directly from the equations of coupled mechanics. The system uncovers accurate market states through the structural interplay of separate nodes, with each entity updating its local calculations while remaining tied to the broader collective consensus.

3. The Proof of Resonance Algorithm

The PoR processing engine runs non-stop across all validator nodes. It does away with old round-by-round ordering, removing leader selection stages, fixed voting schedules, and basic request-response patterns. It maintains an active convergence matrix that records final Resonant Values the moment the network reaches formal mathematical coherence.

3.1 Value Observations

Every active validator checks the real-world value of registered assets through a three-stage routine:

First, the validator grabs raw pricing inputs from its local data pipelines, such as public spot APIs, connected hardware sensors, telemetry data, or external reference channels. Different nodes can use distinct feed combinations; this variety strengthens the network rather than weakening it.

Second, the validator filters these feeds through a local outlier filter. If an individual feed reports a number that deviates by more than 15% from the median of its peer feeds, that input is dropped. This isolates broken or manipulated feeds before they can alter the node's output.

Third, the node packages its data into a Value Observation record and shares it across the peer network:

```
Observation = { asset_id, observer, timestamp,
                zir_value, confidence, dimensions,
                source_hashes }
```

The confidence variable is crucial. It represents the node's self-measured reliability weight for its report, calculated by looking at feed agreement, source diversity, and data age. A node checking three identical real-time feeds reports a high confidence score; a node stuck with one lagging feed reports a low score. This reported confidence level directly influences the node's weight in the global convergence algorithm, giving nodes a strong reason to report accuracy levels honestly.

The dimensions section logs more complex data features, such as available liquidity, volatility, tracking history, scarcity, and network demand metrics. These elements help price advanced ZRC-1 contracts and do not alter the main convergence calculation, but they remain attached for open verification.

Source hashes are SHA3-256 signatures of the raw API text logs. They let external entities verify that an observation came from the stated source without needing to display the full raw data payload.

3.2 The Convergence Step

Once every second, each node runs a local convergence calculation across all observations collected over the past 30 seconds for a given asset. This is the heart of the protocol.

Step one: remove old data. Any observation that has aged beyond 30 seconds is dropped. Nodes that lose connection or experience data delays lose their network influence automatically without needing manual penalties.

Step two: determine the ZTI-weighted median. Every individual observation receives a functional weight determined by multiplying the creator's current ZTI score by their reported confidence value:

$$weight(obs) = ZTI(observer) \times obs.confidence$$

The dataset is organized by reported asset value to locate the weighted median, the point where the cumulative weight balances exactly at half of the total weight pool. This specific number becomes V_{target}, the current field estimate.

Using a weighted median instead of a weighted mean is a deliberate choice designed for security. A mean can be shifted by any entity holding a large weight block, since pushing a number far out moves the average proportionally. A median is resilient: to adjust it, an attacker must control more than half of the active weight pool. There are no partial results. You either sway the midpoint or you do not.

Step three: each node updates its local pricing model toward V_{target}. The rate of adjustment depends inversely on the validator's own ZTI score:

$$step = step\_base \times (1 - zti\_factor \times our\_ZTI)$$
$$our\_model[asset] \mathrel{+}= step \times (V\_target - our\_model[asset])$$

A node with a high ZTI score, meaning a node with a proven history of accuracy, adjusts its position slowly. It acts as a heavy anchor that stabilizes the pricing field. A node with a low ZTI score shifts its model rapidly, aligning with consensus rather than pushing the collective. This structural asymmetry creates built-in network intelligence: veteran nodes anchor the values, while new nodes calibrate themselves against the established baseline.

Step four: track the Coefficient of Variation across the current observation group:

$$CV = \sigma(values) / \mu(values)$$

The Coefficient of Variation gauges the spread of data relative to the average. A CV of 0.02 shows that observations fall within a tight 2% band around the mean. This metric scales perfectly regardless of whether an asset is priced at 0.001 ZIR or 100,000 ZIR.

3.3 Finality and the Lock

When the CV drops below the 2% threshold and at least three unique nodes are contributing data, the pricing field is locked. The network records a verified state event called a Lock:

```
Lock = { asset_id, epoch, resonant_value, CV,
         observation_count, supporting_validators,
         aggregate_signature }
```

The collective signature uses a threshold system requiring verification from validators holding at least 67% of the total ZTI weight in that active set. This delivers hard cryptographic finality, letting any user verify the Lock without relying on individual node integrity.

The Resonant Value fixed inside the Lock becomes the official price for that asset epoch. It is appended directly to the Living Ledger, ZIRA's DAG framework, as an unalterable historical node. Advanced ZRC-1 smart agreements reference these epoch positions for programmatic settlements.

Following a Lock event, the ZTI values of all contributing nodes adjust based on accuracy, measured by how close their initial observation sat relative to the final consensus value. This closes the systemic loop: accurate nodes gain network weight, while inaccurate or outlying nodes lose influence. The system self-corrects over time.

4. ZTI: The Foundation of Trust

Every critical feature of the PoR architecture, including its resilience against manipulation, its alignment speed, and its Sybil resistance, depends entirely on the ZTI tracker. Understanding ZTI reveals exactly why the PoR model functions securely.

ZTI stands for the ZIRA Trust Index. It is a value between 0 and 1 that measures how accurately a node reads external reality. A brand-new node enters the network at 0.000, while the theoretical cap sits at 1.000. This metric cannot be bought, delegated, or inherited. It can only be built through continuous, accurate, and stable contributions to the network's consensus process.

This dynamic marks a complete departure from standard proof-of-stake models. In proof-of-stake, influence scales directly with capital. In PoR, network weight scales with demonstrated precision. Capital can be moved instantly; historical accuracy must be earned over time.

4.1 How ZTI Is Computed

ZTI combines three independent performance tracking components at every epoch boundary:

$$ZTI(v) = 0.55 \times Accuracy(v) + 0.25 \times Consistency(v) + 0.20 \times Uptime(v)$$

Accuracy forms the largest component weight because it directly protects the truth of the pricing field. After a Lock occurs, every node that provided data receives an accuracy calculation:

$$error(v) = |obs.value - Lock.resonant\_value| \,/\, Lock.resonant\_value$$
$$acc\_score(v) = max(0,\ 1 - (2 \times error(v))^2)$$

The non-linear penalty is intentional. A node deviating by 10% receives an accuracy score of 0.96, a minor reduction. A node missing by 30% drops to 0.64, while a 60% error drops the score straight to 0. This setup penalizes systematic attempts to distort data while tolerating small background errors.

These scores merge into an exponential moving average tracking model using a smoothing factor of $\alpha = 0.08$:

$$Accuracy(v) \leftarrow 0.08 \times acc\_score + 0.92 \times Accuracy(v)$$

This slow rate of adjustment, needing roughly 12 full epochs to absorb half of a new trend signal, ensures that a single bad data point will not ruin a trusted node's standing. However, continuous mistakes cause significant damage: a validator that stays inaccurate will see its score decay steadily over days.

Consistency tracks the variance of a node's reporting across short timelines. A node that reports erratic numbers inside a brief epoch window receives a penalty even if its average happens to hit the target. Data noise damages consensus just as much as intentional bias.

Uptime monitors active data submissions across a rolling 30-day window, measuring the percentage of epochs where a node successfully provided an observation. Going offline triggers a

decay function: for every epoch of absence, the node's score is multiplied by 0.9997. Leaving the network offline for a full month degrades the total ZTI score to roughly 44% of its original value.

4.2 The Bootstrap Path

New validators encounter a structural barrier when joining: starting at ZTI = 0 means their data carries zero weight in the consensus calculation. Earning structural trust requires active participation.

To progress, the new node must broadcast accurate observations even though its inputs do not yet alter the consensus path. After a week of reliable data reporting, the node's ZTI score rises toward the 0.05 to 0.10 range, allowing it to join core committees. A month of accurate tracking moves the ZTI toward 0.3 to 0.5. Reaching a score of 0.7 or higher, the baseline required to qualify for Master Node status, requires three months of top-tier precision.

This slow ramp-up acts as a core security feature. Trust requires a significant time investment that cannot be bypassed with capital. This barrier is what secures the pricing field.

This produces a concrete defense: on a network where established nodes hold an average ZTI score of 0.65, an attacker spinning up 1,000 new nodes at ZTI = 0 has absolutely no influence over the Resonant Value. If those same 1,000 nodes spend three months tracking accurately to build their scores to 0.3, they will gain significant network weight, but they can only achieve this by acting as honest participants, converting the attack into supportive operation.

5. Manipulation Resistance

The manipulation defense mechanics of the PoR architecture stem from three combined properties: the weighted median filter, ZTI-gated weight scaling, and the accuracy-based reputation feedback cycle.

5.1 The Weighted Median Threshold

Suppose an attacker manages to control a fraction f of the global ZTI pool and attempts to distort the Resonant Value from the authentic market price P^* to a malicious target P_{attack}. Under a classic weighted mean system, that attacker shifts the consensus point by a direct fraction:

$$\Delta = f \times (P_{attack} - P^*)$$

This means any small slice of capital weight yields a direct shift in consensus output. An actor holding 30% of network stake can move a mean-based price 30% toward their target.

A weighted median changes this logic completely. The median marks the explicit point where cumulative weight hits the 50% boundary. Shifting a median requires an attacker to flood the system with enough data points to relocate that 50th percentile mark. If an attacker controls less than half of the active ZTI weight and honest nodes remain grouped around the true price P^*, the attacker cannot alter the median at all, no matter how extreme their reported values are.

The median function prevents partial exploits. An attacker holding 49.9% of the network's ZTI score cannot move the Resonant Value, because the remaining 50.1% of honest reputation weight anchors the median firmly at the real-world price point.

5.2 The Self-Defeating Attack

Now consider an attacker who manages to cross the 50% threshold by spending months operating honestly to build up their reputation scores. They start sending malicious observations to skew the market price toward their target.

The protocol reacts immediately by degrading their ZTI:

Every single epoch, their accuracy score is measured against the final Lock value. Because their reports deviate heavily from the real-world price, their accuracy component drops. Thanks to the quadratic penalty function and the alpha = 0.08 smoothing factor, just three epochs of a 50% reporting error will drop their Accuracy score from 0.9 to around 0.6. As the attacker's ZTI score decays, their weight in subsequent consensus rounds shrinks, neutralising the exploit.

The attack mechanism destroys the asset needed to execute it. Using ZTI weight to falsify data burns the reputation score itself. This sets PoR apart from stake-based ledgers, where an attacker can execute an exploit, pocket the profit, and retain their locked capital tokens. In PoR, the reputation drains away during the attempt. The attacker pays for the manipulation using the exact resource required to attempt it.

5.3 The Sybil Barrier

Launching thousands of virtual nodes costs very little computer processing power. However, every new node enters the ecosystem with a ZTI score of zero. Building meaningful trust weight across those accounts requires months of consistent performance.

To execute a 50% exploit on an active network with 100 validators averaging a ZTI of 0.65, an attacker would need to match 65 units of total weight. If each malicious node spends three months operating cleanly to hit a ZTI of 0.5, the attacker needs 130 separate nodes. All 130 must provide

accurate data for ninety days, meaning they function as productive components of the network during that entire build-up.

The Sybil attack path dissolves into a standard participation model: the only way to acquire the trust weight needed to manipulate the system is to operate honestly for so long that attacking becomes counterproductive.

6. Comparison to Existing Oracle Designs

The matrix below contrasts the PoR model with three established oracle frameworks. Each of these networks represents significant engineering and has been tested under real market conditions.

Property	Chainlink	Pyth	Band	Proof of Resonance
Architecture	External oracle layer	Publisher network	External oracle layer	Endogenous (validator-native)
Aggregation	Median	Confidence-weighted mean	Median	ZTI-weighted median
Update frequency	On-demand	400ms slots	On-demand	Every 1 second
Sybil resistance	Stake (LINK)	Permissioned	Stake (BAND)	Earned reputation (ZTI)
Oracle/consensus separation	Yes	Yes	Yes	No — unified
Manipulation cost	Buy 50% of stake	Compromise publishers	Buy 50% of stake	Sustained honesty for months

The core difference lies in the foundational layout. Systems like Chainlink, Pyth, and Band view price discovery as an outside service delivered to a ledger. PoR treats value perception as an organic quality of the ledger itself. This is not a superficial tweak; it changes the answer to what a distributed consensus network is built to achieve.

A ledger that relies on external networks for price information is never fully independent. A network whose core nodes handle data perception natively is. This design advantage becomes critical as transaction automation increases.

We recognize the real challenges PoR must address. Long-standing networks possess extensive historical data and deep node infrastructure. Platforms optimized for speed deliver sub-second data updates from financial institutions. In its early phases, a PoR rollout will feature a smaller validator group, resulting in a lighter tracking field that is more sensitive to temporary swings during the initial scaling period. These are real engineering challenges, but they are justified by the architectural advantages for machine-to-machine operations.

7. The Machine Economy Argument

Historically, the oracle problem has been managed around human constraints. Humans draft contracts, request price points, and verify data accuracy. The time elapsed between an oracle refresh and a human transaction is measured in seconds or minutes, making slow on-demand update updates perfectly acceptable.

A machine-to-machine economy operates on a different timeline. An independent AI agent handling asset allocations, pricing cloud hardware streams, or settling carbon offsets acts at digital speeds. It can run thousands of data-dependent actions every second, requiring real-time pricing metrics that can be checked natively on the ledger without outside lookups.

This is the environment where PoR provides significant utility.

7.1 Autonomous Pricing via ZRC-1

The ZIRA ZRC-1 standard defines Resonance Objects, which are automated financial instruments that calculate their value on the fly by checking the active Resonant Value of a linked asset. A ZRC-1 script for data processing power reads as follows:

```
object GpuComputeHour {
  value(): ZIR { return resonance("GPU_H100") }
  distribute() { route(0.90, provider); route(0.10, protocol) }
}
```

When an AI agent deploys this object to clear a hardware balance, the system checks the value function directly against the active epoch Lock for that resource asset. There are no oracle queries, API calls, or third-party pipelines involved. The current price is derived directly from the collective PoR consensus.

For this structure to work securely, data perception must sit inside the core ledger layer. Relying on an external contract introduces an outside point of trust. Accessing the Resonant Value directly from the same ledger tracking the transaction removes any security dependencies outside the core consensus layer.

7.2 AI-to-AI Coordination Without Human Oracles

Consider a group of autonomous programs, such as a coordinator agent that retains five specialist nodes to calculate the value of physical infrastructure assets based on disparate data feeds and models. The coordinator must combine these separate reports into one result. In a traditional environment, this step would require a trusted human administrator or a centralized database service.

Under the ZIRA architecture, the software nodes feed data straight into the PoR engine. Each independent report acts as an active Signal broadcast across the network. The coordinator does not manually aggregate the metrics; the core PoR protocol processes the inputs, weighting each node's submission by its historical ZTI rating. The resulting price value is a product of the consensus field itself.

Settlement occurs automatically following the Lock event. The manager node pays each specialist based on its contribution accuracy. The entire transaction cycle, from project assignment and data processing to final payment, runs autonomously on the ledger without human intervention.

This is exactly what automated machine ecosystems require: systems that can run smoothly without human oversight.

7.3 ZIR as the Numeraire

By tracking the real-time value of all core assets in ZIR, the native token becomes the standard accounting unit for the machine economy. This does not happen due to regulatory mandates, but because it serves as the base unit in which other assets are continuously valued.

Traditional fiat currencies act as registered assets within this framework, with their value relative to ZIR adjusting based on real-world market liquidity. Autonomous programs interfacing with legacy banking networks check the Resonant Value of currencies like USD or EUR directly on the ledger instead of relying on centralized banking APIs.

This is a functional observation about system efficiency rather than a utopian claim about replacing currency. When continuous, decentralized value tracking is available for any asset class, the unit used to measure everything else naturally becomes the foundational reference standard.

8. Open Questions

The PoR model is an evolving architecture. We are actively tracking several core research challenges that we intend to resolve in future versions of the protocol.

8.1 The ZIR Pricing Circularity

The PoR engine calculates asset prices in terms of ZIR, but ZIR's own market value must originate from external trade. On the early testnet, ZIR uses a bootstrapped starting value relative to USD. This temporary baseline will give way to open market pricing as ZIR launches on external exchanges. Managing this transition from a fixed baseline to a dynamic market price requires further protocol design work.

8.2 Thin Field Vulnerability

Networks with few validators face higher systemic risks than mature deployments. With only ten nodes online, a single offline node can push the CV past the target threshold and delay a Lock event. A large network with a thousand nodes is far less sensitive to single points of failure. The security of the pricing field scales with validator growth, making the early rollout phase a sensitive window. We are designing explicit safeguards for this scaling period, including higher minimum node counts and stricter consensus filters for early epochs.

8.3 Feed Quality at the Source

The PoR framework sorts and aggregates observations, but it cannot verify if the underlying data sources are reporting accurately. A coordinated group of data providers reporting identical false numbers could distort the final value even if the network nodes process those inputs honestly. Resolving this challenge requires mandating varied feed sources across validators and integrating zero-knowledge validity proofs for data inputs.

8.4 Optimal Parameter Tuning

The current protocol values, such as the 2% CV cutoff, the alpha = 0.08 smoothing average, the 0.9997 absence decay multiplier, and the 0.55/0.25/0.20 ZTI weights, were selected through simulation modeling rather than live production data. These variables will likely need refinement once the network goes live. The system's governance layer allows for on-chain parameter adjustments, but finding the ideal values for long-term operations requires empirical tracking.

9. Conclusion

The oracle problem is a long-standing challenge in distributed ledger engineering. The traditional solution of using outside networks to relay data works well enough for human trade and systems where minor centralization is acceptable. For an automated machine economy, it is inadequate.

Proof of Resonance is a structural shift in oracle design. It transforms value perception into a core feature of the consensus layer rather than treating it as an outside service. The same nodes that confirm transaction blocks measure asset prices. Their network weight across both functions is tied to a single source: an accuracy record built over months and lost quickly if abused.

The mechanical clock comparison describes the underlying mathematics. Huygens' pendulums synchronized because the physical properties of coupled oscillators made alignment the most stable state. The PoR model achieves pricing consensus for the same reason: when a network of independent nodes updates local models while tied together by a reputation-weighted messaging layer, aligning on the true price becomes the stable equilibrium point. Deception is unstable; honesty is the state the system naturally settles into.

This architecture is built for the emerging digital landscape, where autonomous software entities trade, coordinate, and settle accounts independently. An automated economy requires an underlying ledger that can perceive value natively. The oracle cannot be a centralized human institution; it must be the network itself.

This is the infrastructure delivered by ZIRA's Proof of Resonance.

Appendix A: Mathematical Reference

A.1 Notation

- V — set of active validators
- $N = |V|$ — total validator count
- $O(v, a, e)$ — observation by validator v for asset a at epoch e
- $ZTI(v)$ — trust index of validator v ∈ [0, 1]
- $w(v) = ZTI(v) \times confidence(O)$ — effective weight of observation
- $\mu(S)$ — arithmetic mean of set S
- $\sigma(S)$ — standard deviation of set S
- $M_w(S)$ — weighted median of observations S

A.2 ZTI-Weighted Median

Given a pool of observations $P = \{(v_i, value_i, w_i)\}$, sorted by value:

$$W_{total} = \Sigma_i w_i$$

Find smallest k such that:

$$\Sigma_{i=1}^{k} w_i \geq W_{total} / 2$$

$$M_w(P) = value_k$$

A.3 Coefficient of Variation

$$CV(P) = \sigma(\{value_i : i \in P\}) / \mu(\{value_i : i \in P\})$$

A.4 ZTI Update

$$error(v) = |obs.value - Lock.resonant\_value| / Lock.resonant\_value$$
$$acc\_score(v) = max(0, 1 - (2 \times error(v))^2)$$
$$Accuracy(v) \leftarrow \alpha \times acc\_score(v) + (1-\alpha) \times Accuracy(v) \quad [\alpha = 0.08]$$
$$ZTI(v) \leftarrow 0.55 \times Accuracy(v) + 0.25 \times Consistency(v) + 0.20 \times Uptime(v)$$

A.5 Emission Schedule

$$E(t) = E_0 \times \lambda^t \quad \text{[emission per epoch]}$$
$$E_0 = 1{,}000{,}000{,}000\ \mu ZIR \quad \text{[initial rate]}$$
$$\lambda \approx 0.9999999453 \quad \text{[decay factor]}$$
$$\Sigma_{t=0}^{\infty} E(t) = 6{,}000{,}000{,}000\ ZIR \quad \text{[total emission]}$$

A.6 Convergence Condition

Lock committed when:

- $CV(P) < \tau_{cv}$ [τ_{cv} = 0.02 by default]
- $|P| \geq N\_min$ [N_min = 3 by default]
- $\Sigma_i w_i \geq \theta \times ZTI\_total$ [θ = 0.67]

Appendix B: Protocol Parameters

All parameters below are on-chain governable via ZTI-weighted proposal and 67% approval threshold.

Parameter	Default Value	Description
Epoch duration	5,000 ms	Duration of one consensus round
PoR window	30,000 ms	Sliding window for observation collection
CV threshold (τ_{cv})	0.02	Convergence criterion — Coefficient of Variation
Min observations	3	Minimum validators for Lock commitment
Finality threshold	0.67	ZTI fraction required for threshold signature
ZTI accuracy weight	0.55	Weight of accuracy in ZTI computation
ZTI consistency weight	0.25	Weight of consistency in ZTI computation
ZTI uptime weight	0.20	Weight of uptime in ZTI computation
EMA alpha (α)	0.08	Smoothing factor for ZTI component updates
Absence decay	0.9997	Per-epoch ZTI multiplier when absent
Step size base	0.10	Base model adjustment step size
Step ZTI factor	0.85	ZTI dampening on model adjustment
Master Node threshold	0.70	ZTI required for Light Client service eligibility
Base fee	1,000 μZIR	Minimum transaction fee
Fee burn ratio	0.50	Fraction of fees permanently removed from supply
Signal interval (default)	10 epochs	Default observation broadcast frequency

This document is released for public review. Feedback, corrections, and technical challenges are welcomed at research@zira.network. The protocol described here is implemented in the open-source ZIRA node software and is live on the ZIRA testnet.