

Hoarding node: embedded Consensus Validation of Blocks and Headers

Work Package Name

Embedded Consensus Validation of Blocks and Headers

High-level Description

The hoarding node connects to multiple Cardano peers simultaneously and records the blocks and headers they advertise. It currently performs a set of structural checks on each block — detecting integrity failures, mismatches between headers and bodies, and blocks outside the requested range. These checks require only the block itself and catch a class of obvious anomalies.

Without consensus validation, the hoarding node cannot defend itself against adversarial peers: a peer can advertise headers or blocks that are structurally plausible but invalid according to Ouroboros rules, and the node has no means to detect or attribute this. What the existing checks cannot catch is invalidity that requires chain context: header envelope invalidity (prev-hash linkage, block and slot number sequence), consensus invalidity (a block produced by a peer who was not the legitimate slot leader), and ledger invalidity (transactions that violate UTxO rules). Detecting these requires the ledger state — the stake distribution (from which slot leader election is derived), the per-peer HeaderStateHistory carrying operational certificates for KES verification, and the current UTxO set for transaction validity.

This work package embeds `ouroboros-consensus` — the same consensus library used by the Cardano node — directly into the hoard process. Rather than querying an external Cardano node over a socket, hoard runs the full validation pipeline in-process. This expands block classification from a binary (canonical — part of the immutable chain — or orphaned) to a richer taxonomy that includes three stages of invalidity: header envelope errors, consensus errors (invalid slot leader claim, bad VRF proof), and ledger errors (invalid transactions).

A second capability follows from this: header validation. Block headers arrive via ChainSync before block bodies are fetched. With access to the ledger state, hoard can validate each header as it arrives — detecting an invalid slot leader claim at the earliest possible moment and attributing it to the specific peer that advertised it, without ever downloading the block body.

As a by-product, the external Cardano node is no longer required as an operational dependency. The two things hoard currently queries from it — the immutable tip and per-block canonical status — are both provided directly by the embedded consensus store.

Deliverables:

- Richer block classification, including three stages of invalidity, recorded in the hoarding database and exposed via the HTTP API.
 - Per-peer header validation at the ChainSync stage, before block bodies are fetched. Invalid headers recorded and attributed to the peer that advertised them.
 - Removal of the external Cardano node as a required operational dependency.
-

Core Objectives

- Detect header envelope, consensus, and ledger invalidity — blocks that violate header envelope rules (prev-hash linkage, block/slot sequence), Ouroboros consensus rules (illegitimate slot leader, invalid VRF), or ledger constraints (UTxO invalidity) are identified, classified by validation stage, and recorded. This class of invalidity is a strong signal of adversarial behaviour and is currently undetectable.
 - Attribute invalid headers to specific peers — invalidity detected at the ChainSync stage, before block bodies are fetched, is associated with the peer that advertised the header. This enables earlier and more precise attribution of adversarial behaviour.
 - Eliminate the external Cardano node as an operational dependency — the hoarding node no longer requires a running Cardano node socket. Operators need one fewer long-running service to manage.
-

Expected Value

The Cardano network relies on honest block production: each slot leader must be legitimately elected, and the blocks they produce must contain only valid transactions. The hoarding node currently has no way to detect when these rules are violated — a block from an illegitimate slot leader, or one containing UTxO-invalid transactions, is recorded but not identified as anomalous. This work package closes that gap. By embedding the same consensus library used by the Cardano node, the hoarding node gains the ability to detect and record the full range of block invalidity — and to do so at the header stage, before a block body is even downloaded. The result is a concrete, queryable record of adversarial behaviour on the network: which blocks were invalid, at which validation stage, and which peers advertised them.

This data has value for the broader ecosystem. Node operators, researchers, and governance participants can observe whether adversarial block production is occurring on mainnet or testnet, with evidence that is independently collected and attributable to specific peers. Existing checks tell you whether a block was structurally sound and where it ended up relative to the canonical

chain; the new classification adds whether it was valid according to consensus and ledger rules — a distinct and previously undetectable dimension.

A secondary benefit is operational. Removing the external Cardano node simplifies the deployment footprint — operators need one fewer long-running service to manage alongside the hoarding node.

Metrics for Success

- Invalid blocks (consensus and ledger stage) observable and queryable via the HTTP API, validated against preprod — locally or on the live deployment.
- Invalid headers attributable to specific peers, recorded and queryable.
- At least one case of adversarial behaviour (e.g. invalid slot leader claim) detected and evidenced using the richer classification. Local detection against preprod is sufficient; a live deployment — contingent on the infrastructure budget request being approved — provides independently verifiable evidence.

Preprod is the initial validation target. Mainnet is the natural follow-on once a live deployment is in place; the work package is designed to support both without further changes to the classification or API layer.

Strategic Alignment

Pillar: Pillar 1 — Infrastructure & Research Excellence (Focus Area I.2: Threat Detection & Recovery)

Rationale: Directly implements threat detection on the Cardano network. Embedding the consensus library enables detection of blocks from illegitimate slot leaders and UTxO-invalid transactions — classes of adversarial behaviour currently invisible to network observers.

KPI support: Contributes to the monthly uptime KPI by improving visibility into threats that could destabilise the network.

Classification

- New initiative or continuation of existing: Continuation
 - Primary nature: Technical
-

Milestones

Milestone 1 — Embedded ChainDB

Deliverables: The embedded ChainDB runs inside the hoard process and is fed blocks as they arrive. Four new events are published into the pub/sub system: ChainExtended, BlockSealed, BlockRolledBack, BlockRejected. Existing functionality is unchanged — NodeToClient, ImmutableTip, and OrphanDetection continue to operate as before.

Acceptance criteria: The ChainDB syncs in parallel with the existing Cardano node. New events appear in logs confirming blocks are being validated and chain updates observed. All existing tests pass.

Milestone 2 — Block Validation

Deliverables: BlockRejected events from the ChainDB are used to classify blocks. The block classification schema gains new variants (invalid_header_envelope, invalid_consensus, invalid_ledger). Invalid blocks are stored with their validation stage and error detail.

Acceptance criteria: Invalid blocks are classified by stage and stored in the database. Existing canonical/orphaned classification is unaffected.

Milestone 3 — Header Validation

Deliverables: A HeaderValidation component validates each incoming ChainSync header against the embedded ChainDB's ledger state before the block body is fetched. Invalid headers are recorded as InvalidHeaderReceived events and attributed to the peer that advertised them. Per-peer validation state is maintained and discarded on disconnect.

Acceptance criteria: Invalid headers (envelope and consensus stage) are detected, recorded, and attributable to a specific peer. Valid headers pass through without disruption. Headers beyond the forecast horizon are logged at debug level and skipped.

Milestone 4 — API Exposure

Deliverables: Richer block classification (including all invalidity stages) and invalid headers are exposed via the HTTP API.

Acceptance criteria: All three stages of block invalidity are queryable via the API. Invalid headers are queryable and attributable to specific peers.

Milestone 5 — Remove External Node Dependency

Deliverables: OrphanDetection is wired to BlockSealed and BlockRolledBack events from the ChainDB. The NodeToClient effect is removed. The immutable tip is sourced from the embedded ChainDB rather than from the external node. The external Cardano node is no longer required.

Acceptance criteria: The hoarding node runs end-to-end without a Cardano node socket.

Canonical and orphaned classification continues to work correctly via ChainDB events. All tests pass. Documentation no longer references the external node as a required component.

Supporting Material

- <https://github.com/tweag/cardano-hoarding-node/blob/main/proposal/embedded-consensus-validation/proposal.md> — motivation, deliverables, success criteria, effort estimate
- <https://github.com/tweag/cardano-hoarding-node/blob/main/proposal/embedded-consensus-validation/design.md> — ChainDB integration: component model, effect stack, implementation stages
- <https://github.com/tweag/cardano-hoarding-node/blob/main/proposal/embedded-consensus-validation/header-validation-proposal.md> — per-peer header validation rationale and design
- <https://github.com/tweag/cardano-hoarding-node/blob/main/proposal/embedded-consensus-validation/header-validation-design.md> — HeaderValidation component implementation