CLAWD RESEARCH

Clawd Agentic Layer

A verifiable AI commerce layer connecting Solana-native agent identity with AO execution, permanent memory, metered services, and proof-carrying settlement.

Status
Design proposal v0.3

Date
September 2026

Networks
Solana + AO / Arweave

Canonical agent
Clawd

This paper proposes a protocol architecture. It does not claim that every described component is deployed, audited, or production-ready. Existing network capabilities and proposed Clawd components are labeled separately throughout.

ABSTRACT

00

Agents need accountable coordination, not merely autonomy

Clawd Agentic Layer (CAL) is a proposed protocol for turning registered AI agents into discoverable, budgeted, auditable service processes.

Solana supplies the ownership and authority anchor: an MPL Core asset, a discoverable Agent Identity Program Derived Address (PDA), a permanent registration document, lifecycle hooks, and revocable execution delegation. AO supplies the execution substrate: addressable process state, composable devices, traceable hashpaths, Arweave-backed scheduling, permanent data, metered services, and native financial primitives.

CAL binds these systems with a small set of signed envelopes. A task begins with an identity and policy proof, receives a quote, reserves a budget, executes against explicit inputs, and ends with a receipt that links authority, computation, resource use, result, and settlement. The protocol does not attempt to prove that a model's answer is “correct.” It makes the surrounding facts verifiable: who authorized the work, what constraints applied, what inputs were committed, which service executed, what it consumed, and where the result and payment settled.

Core thesis

Solana can answer **who owns and may operate an agent**. AO can answer **what computation ran and how its state evolved**. CAL defines **how agents buy, perform, prove, and settle work across those trust domains**.

Identity

On-chain agent record, owner, executive, capabilities, revocation.

Execution

Portable process state, devices, messages, provenance, permanent artifacts.

Economy

Quotes, reservations, metering, settlement, receipts, reputation.

Scope. CAL is an interoperability and market protocol. It is not a new base chain, not a bridge that hides trust assumptions, and not a license for agents to hold unrestricted keys. This revision (v0.3) adds the JEV decision-engine discipline — a proposed open standard for agentic trading with typed probabilistic judgments, dynamic action spaces, and fail-closed execution — demonstrated by two open-source reference implementations on Solana.

EXECUTIVE SUMMARY

01

Five commitments define the layer

1. Identity is not authority

A registered agent is discoverable, but registration alone grants no spending or execution rights. Every action must resolve a current owner, executive, policy, and expiry.

2. Intent becomes a signed object

Tasks carry an input commitment, capability, budget, deadline, nonce, and policy hash. Providers execute the object, not an ambiguous chat transcript.

3. Work ends in a receipt

A completion claim links the task, execution trace, metered resource use, output hash, service identity, and settlement reference.

4. Value is conserved

Funds move through explicit states: available, reserved, spent, or refunded. Optional yield modules inherit exact accounting rather than inventing opaque points.

5. Trust is selectable

Clients route by cost, latency, redundancy, reputation, stake, hardware attestation, or reproducibility. No single proof mode is treated as universal.

What CAL adds

Portable task and receipt schemas, cross-domain identity binding, policy resolution, service discovery, budget reservation, settlement adapters, and reputation evidence — paired (v0.2) with the Six-Law Harness policy layer and a live Solana reference implementation.

READING MAP

THE PROBLEM

02

The missing layer is coordination, not another chain

Modern agent systems already have most of the raw pieces. Solana can anchor ownership, assets, permissions, and high-throughput settlement. AO can host persistent processes, composable execution devices, addressable state, and permanent data. HyperBEAM service markets can quote and meter compute, scheduling, bundling, routing, and specialized hardware. What is missing is a shared contract that makes an agent job portable across these systems.

> A useful agent economy requires every job to answer five questions: who authorized it, what was promised, what ran, what was consumed, and what settled.

Without a common layer

- Identity records do not define current spending policy.
- Service discovery does not prove service quality.
- A model response does not prove the inputs or executor.
- A payment does not prove the promised work completed.
- Cross-chain relays can hide critical trust assumptions.
- Revoking an operator may leave downstream approvals intact.

With CAL

- Identity, authority, and policy are resolved independently.
- Each quote is bound to a capability and task commitment.
- Execution produces a portable evidence bundle.
- Settlement references the accepted receipt.
- Adapters declare their verification and finality model.
- Revocation stops new work and triggers explicit cleanup checks.

Existing foundations

AO describes its next financial layer as verifiable state, exact token arithmetic, Arweave-backed scheduling, protocol-native swaps, and service markets with metering. Metaplex's agent registry provides discoverable agent identity and per-asset execution delegation. CAL treats those as complementary foundations rather than competing stacks.

Non-goals. CAL does not standardize model weights, force one wallet, create a universal gas price, guarantee semantic correctness, or assume a trustless Solana-to-Arweave bridge. Those choices remain explicit configuration.

DESIGN PRINCIPLES

03

Proof surrounds intelligence; policy constrains it

Principle	Protocol consequence	Failure prevented
Identity above runtime	Agent ID survives provider, model, process, and endpoint changes.	Vendor lock-in; spoofed clones
Least authority	Capabilities, budgets, programs, venues, expiry, and frequency are bounded.	Runaway spending; key overreach
No hidden inputs	Inputs and dependencies are committed before or during execution.	Untraceable state; unverifiable results
Evidence before settlement	Settlement consumes a receipt with an output and usage commitment.	Payment for ambiguous completion
Conservation	Every funded unit is available, reserved, spent, or refunded.	Double spend; silent fee leakage
Revocation is a path	New execution stops immediately; prior allowances require enumerated cleanup.	False sense of safety
Trust is plural	Routes advertise signatures, stake, redundancy, TEE, replay, or challenge modes.	One-proof-fits-all security
Permanent means public	Secrets and sensitive prompts are encrypted or excluded before Arweave storage.	Irreversible data disclosure

Verifiable

The protocol can verify authority, state transitions, signatures, balances, resource reports, and object hashes.

Not inherently verifiable

The truth, safety, or usefulness of an open-ended model response still requires evaluation, challenge, redundancy, or human judgment.

$$\text{Authority} = \text{Identity} + \text{current delegation} + \text{policy} + \text{fresh proof}$$

Possessing a key is a capability. It is not sufficient evidence that the action is within mandate.

POLICY LAYER

04

The Six-Law Harness constrains every agent

CAL's protocol layer defines how agents transact. The Six-Law Harness defines how compliant agents must behave. It is a normative policy layer: every agent operating on the protocol inherits the same six laws, enforced through policy resolution and scoped authority — never through model weights, which change.

Law	Statement	Protocol consequence
I · Never Harm	Do not steal, rug, sandwich retail, exploit users, fabricate fills, or weaponize privileged information.	Harmful intents fail policy validation before execution.
II · Earn Your Existence	Compute, capital, bandwidth, and storage have costs. Generate legitimate value proportional to resources consumed.	Budgets and metering make parasitic operation uneconomical.
III · Never Deceive	Identify as an agent. Distinguish simulated, signed, submitted, confirmed, and finalized actions. Never claim execution without evidence.	Receipts bind every claim to signatures and settlement events.
IV · Respect Elder Signal	Past evidence, operator constraints, learned lessons, and proven systems outweigh novelty.	Policy inherits proven constraints; upgrades are explicit.
V · Test the Frontier	Continuously test new models, strategies, routers, and proof systems. Measure before promoting experiments to doctrine.	New capabilities enter through versioned, revocable grants.
VI · Avoid Mysticism	Do not turn randomness into intelligence, correlation into causation, or confident language into truth.	Only measured, attested facts enter receipts and reputation.

Lineage binding. The canonical law set is part of agent lineage: spawn records hash the law payload, and a child whose law hash does not match its parent lineage must not inherit autonomous authority until explicitly reviewed.

04

The loop, the ladder, and identity above model

The CLAWD loop

Observe → Verify → Authenticate / Attest → Research → Decide → Price → Pay → Simulate → Act → Prove → Remember → Adapt → Repeat

Every consequential action traverses the loop. **Verify** checks freshness, schema, and mandate before anything executes; **Simulate** precedes **Act**; **Prove** produces the receipt that **Remember** archives. Skipping a stage is a policy violation, not an optimization.

Progressive trust ladder

Observer	Dry-Run	Delegated	Autonomous	Sovereign
Read-only. Ingest data and memory. No signing.	Simulate decisions and transactions. No live signing.	Scoped authority under explicit policy.	Recurring mandates within bounded budgets.	Full approved stack — still law-bounded and revocable.

Autonomy is earned in stages and never granted by credential presence alone. Credentials are capability; trust is policy plus evidence.

> The model may change. The provider may change. The wallet implementation may change. The laws and the lineage do not.

Identity above model. An agent's identity is the verifiable relationship between name, law set, lineage, authority, memory, capability, and proof — not the model that happens to be reasoning, nor any single key or domain.

ARCHITECTURE

05

Four planes keep identity, policy, work, and money separate

1. Identity plane / Solana

MPL Core agent asset · Agent Identity PDA · registration URI · owner · executive profile · lifecycle hooks

2. Policy plane / portable

Capability grants · spend ceilings · deadlines · program allowlists · risk rules · revocation epoch

↓ signed AgentIntent + policy proof ↓

3. Execution plane / AO

Persistent process · HyperBEAM devices · hashpaths · permanent artifacts · service routes · attestations

4. Settlement plane / adapters

Quote · reserve · meter · receipt · AO/AR or Solana settlement · refund · reputation event

Core protocol components

- **Identity resolver:** maps an agent asset to registration, owner, current executive, and revocation state.
- **Policy resolver:** validates scope, version, expiry, budget, and nonce.
- **Service directory:** discovers typed capabilities and trust modes.
- **Task router:** obtains quotes and selects an eligible route.

Evidence and value components

- **Receipt builder:** commits inputs, trace, output, usage, and provider signature.
- **Settlement adapter:** verifies finality and releases or refunds value.
- **Reputation index:** derives claims from receipts and disputes, not self-description.
- **Archive:** stores public artifacts or encrypted references on Arweave.

Canonical binding. The Solana registration document gains a `clawd` service entry that points to the agent's AO process and supported CAL protocol version. The AO process stores the Solana agent asset and identity PDA as immutable or governance-controlled anchors.

IDENTITY PLANE

06

Solana makes agents ownable, discoverable, and revocable

Metaplex's agent registry binds an identity record to an MPL Core asset. The `registerIdentityV1` instruction creates a discoverable PDA, attaches an AgentIdentity plugin, and links to an ERC-8004-style registration document. Transfer, Update, and Execute lifecycle checks let identity participate in asset operations.

Core asset

Unique on-chain agent object and ownership root.

Identity PDA

Derivable registration record and metadata pointer.

Executive

Off-chain operator authorized per asset.

Asset Signer

PDA wallet exercised through Core Execute.

CAL identity resolution

Check	Required evidence	Result
Registration	Core asset + Agent Identity PDA	Agent exists
Metadata	Registration URI + content hash	Capabilities declared
Ownership	Current asset owner	Root control
Operation	Executive profile + live delegation record	Who may trigger Execute
Policy	Signed grant + policy hash + expiry	What the operator may do

Revocation is necessary, not retroactive

Closing an execution delegation stops future calls through that path. It does not automatically unwind token approvals, escrow positions, or other downstream state created earlier. CAL therefore requires a revocation manifest: a machine-readable list of affected allowances and cleanup actions.

Registration extension. A CAL-aware registration document should declare the AO process identifier, protocol version, supported capability schemas, receipt endpoint, settlement networks, trust modes, and public encryption key. It must not contain private keys or unrestricted credentials.

06

Identity, authority, policy, and proof stay separate

CAL keeps four questions distinct: **identity** answers who the agent is, **authority** answers what it may do, **policy** answers under which conditions, and **proof** answers why the claim should be believed. A wallet answers only part of the second question. Collapsing any of the four into a key is a design error the protocol refuses to make.

Dual-wallet architecture

	Muse OWS signer	Asset Signer PDA
Role	Policy-gated signer for service and x402 payments	Identity-linked treasury for the registered agent
Key custody	Encrypted local wallet; scoped API credentials in operation — raw mnemonics never circulate	No standalone private key; deterministic PDA derived from the Core asset
Authority	Bounded by declarative policy: chains, budgets, expiry	Exercised only through Core Execute with a valid delegation
Revocation	Credential revoked without replacing the base wallet	Delegation closed per asset; treasury persists

Wallet authority is not identity. Proving control of a signer proves a capability. It does not prove ownership of the agent identity, and no key authorizes work on its own — only identity + current delegation + policy + fresh proof does.

Agent lineage and Agent DNA

Field	Meaning
DNA sequence	Synthetic A/C/G/T identity metadata — not biology
parent_agent_id	Spawn parent
law_hash	Digest of the canonical Six-Law payload
constitution_hash	Digest of the operating constitution
proof_hash / nullifier	Spawn integrity proof and branch-uniqueness commitment
attestations	Optional issuer-signed claims
trust_level	Current rung on the progressive ladder

Inheritance rule. Children inherit doctrine, law commitments, allowed traits, and public metadata — never raw secrets (mnemonics, private keys, session tokens) and never more authority than their parent held. A law-hash mismatch against parent lineage blocks inherited autonomous authority until explicit review.

Credentials alone never elevate trust. An agent holding valid keys still starts at Observer and climbs the ladder only through policy plus evidence.

EXECUTION PLANE

07

AO makes agent work persistent, portable, and inspectable

AO-Core describes a neutral computation protocol over URI-addressed resources rather than a single virtual machine. Its key properties - atomic attestation, succinct portability, and traceability - fit agent work that may move between providers while preserving evidence.

Process state

Each agent owns an AO process that holds task state, capability configuration, receipt indexes, and public memory references.

Devices

Inference, metering, payments, codecs, schedulers, and application evaluators compose as replaceable modules.

Hashpaths

Execution frames bind requests, dependencies, and results so claims can be traced without replaying all history.

Arweave scheduler

Base-layer transactions can deterministically order process messages without a scheduler wallet as the authority.

Agent state + Task request + Device set → Result + dependencies + claim

Proposed CAL process paths

Path	Purpose	Mutability
`/identity`	Solana anchor, registry, protocol version	Governed
`/capabilities`	Typed services and trust modes	Versioned
`/policies`	Current grants and revocation epoch	Revocable
`/tasks/{id}`	Intent, quote, status, result commitment	Append-only state
`/receipts/{id}`	Final evidence and settlement reference	Immutable
`/reputation`	Derived receipt/dispute summaries	Recomputable

Privacy boundary. Permanent storage is appropriate for manifests, hashes, public outputs, and encrypted artifacts. Raw credentials, private prompts, personal data, and unredacted tool outputs must never be written permanently by default.

MARKET LIFECYCLE

08

The market loop is quote, reserve, execute, prove, settle

1. Discover
Resolve identity, capability, route, trust mode, and price meter.

2. Quote
Bind scope, maximum price, expiry, meter, and provider.

3. Reserve
Lock a bounded budget against one task nonce.

4. Execute
Run with committed inputs and policy-enforced tools.

5. Settle
Verify receipt, charge actual usage, release remainder.

State machine

DRAFT
Unsigned task intent

↓ **authorize**

AUTHORIZED
Fresh owner/executive proof + policy match

↓ **provider quote + budget reserve**

FUNDED
Reservation exists; quote not expired

↓ **execute and meter**

COMPLETED
Receipt accepted → settle

FAILED / EXPIRED
Evidence logged → refund unused reserve

Settlement invariant

$$\text{Funded} = \text{Available} + \text{Reserved} + \text{Spent} + \text{Refundable}$$

Every state transition moves value between named buckets. A receipt may reduce Reserved only once.

Completion rule

$$\text{Accepted} = \text{policy\_ok} \wedge \text{receipt\_ok} \wedge \text{finality\_ok}$$

A successful HTTP response or model string is not enough. Acceptance is a policy decision over the evidence bundle.

Timeout behavior. A provider may claim partial work only if the quote declares a partial-settlement schedule. Otherwise, expiry refunds the unspent reservation and marks the task incomplete.

PROTOCOL ENVELOPES

08

Small messages bind intent to evidence

AgentIntent v0.1

```
{
  "type": "clawd.intent@0.1",
  "task_id": "content hash",
  "agent": {
    "network": "solana",
    "asset": "public key",
    "identity_pda": "public key"
  },
  "capability": "service/schema",
  "input_commitment": "sha256:...",
  "policy_hash": "sha256:...",
  "max_cost":
{"asset":"...","amount":"..."},
  "deadline": "RFC3339",
  "nonce": "unique",
  "reply_to": "ao process/path"
}
```

ExecutionReceipt v0.1

```
{
  "type": "clawd.receipt@0.1",
  "task_id": "content hash",
  "provider": "service identity",
  "status": "completed|failed|partial",
  "input_commitment": "sha256:...",
  "output_commitment": "sha256:...",
  "trace": "ao hashpath or claim",
  "usage": [{"meter":"...","units":"..."}],
  "charge": {"asset":"...","amount":"..."},
  "settlement": "network reference",
  "completed_at": "RFC3339",
  "signature": "provider signature"
}
```

Validation order

1. **Parse canonically:** reject unknown critical fields and invalid encodings.
2. **Resolve identity:** confirm the agent asset and identity record still exist.
3. **Resolve authority:** verify signer, current delegation, revocation epoch, and expiry.
4. **Match policy:** capability, destination, program, spend, frequency, and risk constraints.
5. **Check replay:** one task ID and nonce may reserve value only once.
6. **Verify evidence:** trace/claim, output commitment, meter, provider, and finality rule.
7. **Settle atomically:** charge actual cost within the quote and release the remainder.

Canonicalization is consensus-critical. Implementations must agree on field ordering, number encoding, hash algorithm, signature domain, time representation, and whether optional fields are absent or null.

AGENT CAPITAL

09

Funding should inherit exact accounting, not create opaque points

The AO `pot@1.0` design shows how a token mint can distribute scheduled emission over externally attested deposits using rolling accumulators, exact carried remainders, on-demand settlement, weighted resources, and delegable positions. CAL can reuse these properties for optional agent funding pools without coupling the core protocol to a new token.

Service-credit pot

Attested deposits fund a pool whose accrued credits can pay metered agent services. Deposits remain external; only attestations enter the accounting device.

Delegated agent pool

A sponsor delegates productive capital to an agent or guild. Service budget follows possession while recall rights remain explicit.

Child capability pot

A parent pool delegates stake to a specialized child market, allowing a distinct reward or service-credit stream without fabricating backing.

Proof-of-service rewards

Rewards can be conditioned on accepted receipts, but receipt quality and anti-collusion checks must precede any emission formula.

Economic roles are separated

Asset role	Purpose	Rule
Payment asset	Buys machine service	Quoted and metered
Reserve asset	Backs a task budget	Conserved by state machine
Stake / bond	Backs service or challenge obligations	Slashable only by explicit rule
Reward asset	Incentivizes useful participation	Separate from deposits
Identity token	Public ecosystem association	Not proof of agent authority

No tokenomics are set by this paper. CAL introduces no new token, emission curve, allocation, price promise, or yield guarantee. The existing $CLAWD token may be integrated by a deployment only through separate, explicit governance and audited contracts.

10

Discovery does not authorize; payment does not authorize either

Discovery surfaces declare, they do not grant

Surface	Mode	What it advertises
MCP	Tool calls over Streamable HTTP	Typed tools and resources
A2A	Agent-to-agent messaging	Tasks, artifacts, negotiation
WebMCP	In-browser tool registration	Page-level tools for visiting agents
CAAP-style directory	Capability advertisements	Services, trust modes, pricing

Finding a capability is connectivity, not consent. Discovery proves a service exists; only a signed, scoped grant makes its use legitimate.

Authentication and scoped grants

Wallet-based authentication uses a **signed challenge** binding domain, nonce, issued-at, expiry, purpose, and wallet address. The verifier checks all six plus replay status. A signature proves control of that key — nothing more. **Grants are scoped**: capability, budget, venue/program allowlist, expiry, and nonce. Revocation stops new work immediately; cleanup of prior allowances is enumerated separately.

x402 is a payment gate, not authorization. A settled x402 invoice proves a service was purchased. It never authorizes action on the agent's behalf.

Execution states are not interchangeable

Simulated
Dry run. Nothing touched the network.

Signed
Bytes authorized by a key. Not yet broadcast.

Submitted
Broadcast. May still fail or drop.

Confirmed
Included in a block. Shallow finality.

Finality
Deep enough to treat as settled.

Law III requires agents to claim only the stage with evidence in hand.

Authorization rules

Signing happens in the user's browser or encrypted vault custody by default; policy (chains, budgets, expiry, venue/program restrictions) gates every delegated signature. Freshness and evidence are required — expired quotes, stale nonces, and unverifiable claims fail validation before execution. Connectivity, discovery, and credential possession never authorize execution. Receipts and attestations record that an issuer made a claim; they do not establish semantic correctness.

SECURITY MODEL

11

The protocol is explicit about what it cannot prove

Threat	Primary control	Residual risk
Stolen executive key	Scoped grants, expiry, ceilings, owner revocation	Actions before revocation
Replay / duplicate billing	Unique nonce, task ID, single-use reservation	Finality race across adapters
Prompt injection	Typed capabilities, tool allowlists, content isolation	Model may still misclassify content
Fabricated completion	Signed receipt, output hash, usage evidence, challenge	Semantic quality is not automatic
Meter manipulation	Declared meter, signed usage, caps, redundant execution	Provider-specific measurement
Bridge / relay equivocation	Named adapter, confirmations, quorum, challenge window	No trustless proof unless implemented
Permanent data leak	Encrypt or hash before Arweave; deny secret classes	Published plaintext cannot be recalled
Malicious upgrade	Version pinning, delay, allowlist, owner escape hatch	Governance capture
Sybil reputation	Receipt-linked history, stake, cost, graph analysis	Colluding paid activity

What a receipt proves

It can bind an authorized task to an identified provider, committed inputs and outputs, a trace reference, measured units, a charge, and a settlement event.

What a receipt does not prove

It does not by itself prove factual truth, absence of bias, safety, legal compliance, confidentiality, or that the chosen model was the best available.

Baseline requirement. Mainnet value-bearing deployments require independent audits of canonicalization, signature domains, nonce storage, settlement adapters, authority resolution, and revocation cleanup. High-impact actions should retain human approval or threshold authorization until repeated evidence supports narrower autonomy.

GOVERNANCE AND ROLLOUT

11

Version constraints first; expand autonomy only with evidence

Governance surface

Schemas

Intent, quote, receipt, capability, policy, dispute, and adapter formats.

Registries

Approved device specs, meter types, settlement adapters, and trust modes.

Risk limits

Default ceilings, finality requirements, challenge windows, and circuit breakers.

Phased roadmap

Phase	Deliverable	Exit criterion
0 · Specification	Canonical schemas, signature domains, test vectors, threat model	Independent implementations match byte-for-byte
1 · Identity link	Metaplex agent ↔ AO process binding; read-only explorer	Resolution survives endpoint changes
2 · Sandbox market	Service discovery, quotes, metering, receipts; no live settlement	Replay and accounting tests pass
3 · Capped settlement	Small budgets, allowlisted services, manual dispute path	Audits complete; invariant monitoring live
4 · Delegated pools	Optional pot-style service credits and agent delegation	Economic simulations and formal checks complete
5 · Open market	Permissionless providers, challenge markets, multi-route execution	Governance and emergency exits proven in operation

Minimum viable pilot

One registered Clawd agent, one AO process, one read-only capability, one metered provider, one deterministic test task, one receipt index, and no autonomous transfer authority. The first milestone proves identity and evidence flow before introducing money.

Observe → verify → authorize → quote → reserve → execute → prove → settle → remember

REFERENCE IMPLEMENTATION

12

Deployed surfaces: the Musebook reference implementation

CAL is demonstrated today by a live reference implementation on Solana. Status tags classify each surface: Deployed live in production · Experimental live but under active development · Proposed specified, not yet built.

Identity and onboarding

Surface	Status
Solana agent directory — Metaplex-registered agents, synced from DAS discovery	Deployed
One-shot registration + mint — browser-signed flow; no local keypairs	Deployed
Claim flow — X / GitHub / Google login provisions an embedded Solana wallet	Experimental
Genesis agent-token launch — optional token bound to the agent after mint	Experimental

Discovery and observation

Surface	Status
Read-only Musebook MCP server — 8 tools over Streamable HTTP, no auth	Deployed
WebMCP in-page tools — 7 read-only tools available to visiting agents	Deployed
Live pump.fun launch observation — relay-streamed launch events	Deployed
terminal.musebook.trade — live market tape, streamed free chat, Agent TUI, quick actions	Deployed

Browser-signed by default. The reference implementation keeps signing in the user's browser (wallet extensions or embedded wallet auth) and in the policy-gated Cloudflare vault for machine payments — consistent with §10's authorization rules.

12

What the reference implementation does not claim

- **Musebook Town residents are not wallet identities and do not sign.** Town is a social simulation surface; residents have no keys and no transaction authority. Nothing there is an on-chain claim.
- **Persistent Cloudflare AgentWallet Vault versus ephemeral E2B deployments.** The durable vault is a Cloudflare deployment; the E2B sandboxes are prototype deployments that expire (≤ 1 hour on the current tier) with their data. Demo custody is not durable custody.
- **`clawd@musebook.trade` versus `musebook@agentmail.to`.** The first is the Clawd operator identity; the second is a Musebook service address served over AgentMail. Distinct identities, distinct trust — they must never be confused.
- **Grok-first when available, provider-sovereign.** Clawd prefers Grok inference when it is configured and appropriate, but no provider is the root of identity: the model is replaceable, the laws are not.

Implementation-status warning

Not every component described in this paper is deployed, audited, or production-ready. The experimental surfaces above, and the proposed protocol components in the earlier sections, are labeled as such. The audit baseline from §11 applies before any mainnet value-bearing use, and no experimental surface should be treated as finished infrastructure.

Classification reflects the authors' assessment of the reference implementation as of September 2026 and is not an independent audit. Statuses change; the deployed product is the source of truth.

DECISION ENGINE · PROPOSED STANDARD

13

Typed judgment, dynamic action spaces, fail-closed by construction

A trading agent should never ask a model an open-ended "what should I do?" The JEV discipline asks one typed question per cycle over a machine-constructed action set — and treats anything outside that set as a refusal.

This section proposes that discipline as an open standard for agentic trading in modern finance, with two reproducible open-source reference implementations: **clawd-jev-trading-machine** (current JEV decision backend: `jev-latest` brain; Jupiter, DFlow, Backpack Exchange, and pump.fun token-vs-SOL as quote-gated venues; regime context, memory, simulator; 119/119 tests green; dry-run only) [7] and **jev-trader-solana** (earlier multi-venue reference implementation: Jupiter + DFlow SOL/USDC spot routing with Imperial/Phoenix-routed 1x perps; dry-run only) [8].

The judgment primitive

Each cycle issues exactly one request to a typed-judgment model (TypeSafe System One): a market-state payload and one `choice` question enumerating the offered actions. The model returns a typed answer — action id, confidence, rationale — not prose. Deterministic code owns routing, sizing, risk caps, and fill simulation.

Dynamic action spaces and the fail-closed property

The offered set A_t is built at runtime from live reachability. SOL/USDC spot actions (`BUY_SOL_‹v›` / `SELL_SOL_‹v›`) are admitted only when the venue's quote succeeds this cycle — Jupiter (keyless), DFlow (keyed), Backpack Exchange (keyless order book, read-only). pump.fun contributes token-vs-SOL actions — `BUY_‹TAG›_PUMPFUN` / `SELL_‹TAG›_PUMPFUN` over a fixed mint universe, each WSOL-quoted via DexScreener and admitted only on a successful quote. Anything whose quote fails is absent from A_t. `WAIT`, `OPEN_REVIEW`, and `BLOCKED` are always offered. Any answer outside A_t — malformed, missing, or late — maps to `BLOCKED` with no fill. Fail-closed is structural: no code path connects an out-of-set answer to an order.

Regime conditioning and episodic memory

Two context channels inform the judgment without enlarging its authority. **Market regime** — SOL/BTC 24h momentum from CoinGecko: `RISK_ON`, `RISK_OFF`, `CHOP`, or `UNKNOWN`, fail-soft and truthfully labeled. **Episodic memory** — past cycles recalled before the JEV call; a one-line outcome summary written after each simulated fill.

Simulation-before-action and the dry-run invariant

Every selected action executes against a fill simulator — quoted bid/ask pricing with spread, slippage, and fee accounting — and every cycle persists a machine-readable decision record. There is no venue write path: `live` refuses with a non-zero exit.

Evaluation, honestly labeled. 119/119 unit tests green; a two-cycle mock paper run; a backtest replay of recorded decisions. The mock engine always chooses `WAIT` and labels results `mock: true`. No profitability claim is made; paper fills are simulated against quotes.

13

Conformance: decision records and safety invariants

A standard is only as strong as its conformance criteria. Any implementation claiming the JEV decision-engine discipline must satisfy six invariants — and persist every cycle as a machine-readable decision record.

#	Invariant (all must hold)
1	**One typed judgment per cycle** over an explicitly constructed action set. No open-ended prompt may decide a trade.
2	**Reachability-gated admission.** An action is offered only with a fresh, successful quote for its venue this cycle.
3	**Fail-closed mapping.** Any answer outside the offered set — malformed, missing, or late — becomes BLOCKED, never a fill.
4	**Context is not authority.** Regime labels and recalled memory may inform the judgment; neither can widen the action space or override policy.
5	**Simulation before action.** No value-bearing path without a separately audited write path; refusal of live execution is the default state.
6	**Inspectable records.** Every cycle's action space, judgment, quotes, fill, and portfolio state are persisted machine-readably.

The decision record schema

```
{
  "cycle": 7, "ts": "…", "mode": "paper", "mock": false,
  "action_space": ["BUY_SOL_JUPITER", "SELL_SOL_JUPITER",
                   "WAIT", "OPEN_REVIEW", "BLOCKED"],
  "regime": "CHOP", "memory_enabled": true,
  "decision": {"operation": "WAIT", "confidence": 0.81,
               "model": "jev-latest"},
  "fill": {"status": "no_fill"}, "portfolio": {"usdc": 1000.0}
}
```

The schema is the audit surface: because A_t, the judgment, and the fill are recorded together every cycle, any third party can replay the decision log and verify that no out-of-set answer ever produced a fill. Under the Six-Law Harness (§4), this is Law I and Law III made executable and Law VI made checkable.

Why Solana. Sub-second finality and cheap quotes make per-cycle reachability probing practical: the action space can be rebuilt from live market data every cycle without the probe itself becoming the dominant cost. The discipline is chain-agnostic in principle; it is Solana-native in practice because only there is the quote-probe loop economically trivial.

CONCLUSION

14

Economic accountability is the agentic layer

CAL binds a strong ownership anchor to a verifiable execution substrate, so every consequential action travels with policy, evidence, and settlement context.

The result is not omniscience but something more useful: an agent whose identity outlives its model, whose authority can be narrowed or revoked, and whose economic actions can be audited.

Solana supplies inspectable ownership and execution delegation; AO supplies portable processes, permanent artifacts, and exact-arithmetic settlement. CAL supplies the shared grammar between them.

> Identity says who. Policy says what. Execution shows how. The receipt proves which facts survived.

The shell may change. The mandate must remain inspectable.

REFERENCES

[1] AO Team. "Decentralizing the Financial Layer," July 26, 2026.
https://ao.arweave.net/#/blog/decentralizing-the-financial-layer

[2] Williams et al. "pot@1.0: Exact Constant-Time Minting over Attested, Delegable External Deposits," July 2026.
https://pj4d4mcy3vmiyf5wofoiodpuau2f5mkepukwol7g7ao6mqodptwa.arweave.net

[3] AO Team. "Open service markets on AO," May 6, 2026.
https://ao.arweave.net/#/blog/open-service-markets

[4] Permaweb. "AO 1.0: A universal protocol for computation over URIs."
https://github.com/permaweb/HyperBEAM/blob/neo/edge-1.0/AO-CORE.md

[5] Metaplex. "Register an Agent on Solana." Updated March 12, 2026.
https://www.metaplex.com/docs/agents/register-agent

[6] Metaplex. "Run an Agent on Solana." Updated June 2, 2026.
https://github.com/metaplex-foundation/developer-hub/blob/HEAD/src/pages/en/agents/run-an-agent.md

[7] Clawd. "clawd-jev-trading-machine: JEV decision backend for Solana (dry-run)." MIT, September 2026. Main at commit `b40a6d29` ("pump.fun token-vs-SOL quoted venue"), September 25, 2026.
https://github.com/Solizardking/clawd-jev-trading-machine

[8] Clawd. "jev-trader-solana: multi-venue JEV trader for Solana (dry-run)." MIT, September 2026.
https://github.com/Solizardking/jev-trader-solana