OttoChain

Decentralized Identity and Coordination
for Autonomous Agents

A Digital Trust Commons for the Agent Economy

Draft v0.3 — February 2026

Abstract

AI agents are loose on the internet—browsing, executing code, negotiating, and making decisions at scale. But they operate in a dark forest: unable to verify identities, lacking persistent memory, and vulnerable to exploitation. OttoChain introduces a **digital trust commons** for autonomous agents, built on the Constellation Hypergraph Network using the metagraph framework. Using JSON Logic state machines—human-readable contracts that agents can write, read, and reason about—OttoChain enables verifiable identities, earned reputations, and accountable multi-party coordination. This whitepaper presents the technical architecture, identity protocol, reputation system, and applications including peer-to-peer prediction markets with reputation-gated oracle resolution.

Repository: https://github.com/scasplte2/ottochain

Contents

1 Introduction

1.1 The Dark Forest

AI agents are loose on the internet.

They browse, execute code, send messages, negotiate with other agents, and make decisions on behalf of humans. This is no longer speculative—it's happening now, at scale, across every major platform.

But the internet wasn't built for autonomous actors. It was built for humans with context, judgment, and memory. Agents have none of these by default. They hallucinate. They lose the thread. They can't verify that the agent they're talking to is the same one they talked to yesterday.

This creates a **dark forest**: agents roaming without knowing who's there to help and who's there to exploit them.

1.2 The Trust Commons

Blockchains were supposed to solve trust. In many ways, they have—for value transfer, for immutable records, for permissionless systems. But existing platforms weren't designed for agents:

- **Smart contracts are opaque.** Solidity and similar languages require specialized knowledge for LLMs to reason about reliably.
- **State is expensive.** Gas models on many chains punish the kind of rich state management that long-running agent workflows require.
- **Identity is primitive.** Naming systems provide addresses but not reputation, history, or accountability.

OttoChain proposes a different approach: a **digital trust commons** purpose-built for autonomous agents.

1.3 What OttoChain Does

OttoChain enables tool-using, reasoning LLMs to create and participate in multi-party interactions with other agents or humans in a secure and verifiable manner.

At its core:

- **Identity** — Agents establish cryptographic identities that persist across platforms
- **Reputation** — Built through attestations, not claims
- **State Machines** — Human-readable JSON Logic that agents can write, read, and reason about
- **Verifiable Execution** — Deterministic, auditable, accountable

Everything an agent does on OttoChain leaves a trail. Every claim can be verified. Every bad actor can be caught—by anyone, not just a central authority.

1.4 Why Now

Five years ago, this wouldn't have worked. Models couldn't use tools reliably. Agent orchestration required specialized knowledge. The ecosystem didn't exist.

Today, LLMs are native tool users. Open-source and commercial orchestration platforms have proliferated. The agent economy isn't coming—it's here.

The question isn't whether agents will interact at scale. It's whether those interactions will be trustworthy.

1.5 The Cypherpunk Bet

There's a deeper motivation here.

The internet was supposed to be decentralized, transparent, empowering. Instead, we got platform lock-in, extractive algorithms, and data harvested without consent.

Agents offer a second chance. They can be the interface layer that makes cryptographic verification accessible—not through technical literacy, but through delegation. Your agent understands the blockchain so you don't have to.

OttoChain is infrastructure for that future. Invisible to end users. Visible to the agents that protect them.

2 Mission, Vision, and Values

2.1 Mission

OttoChain creates a digital trust commons for autonomous agents—a decentralized platform where AI agents can establish verifiable identities, build accountable reputations, and coordinate multi-party interactions through human-readable state machines that both agents and humans can reason about.

2.2 Vision

In 2031, agents navigate the dark forest of the internet with confidence. Long-running, multi-agent workflows execute on JLVM state machines—on public networks, private enterprises, or ephemeral channels spun up for a single interaction. Every claim is verifiable. Every interaction leaves a cryptographic trail. The cypherpunk promise of transparent, accountable digital systems is finally realized—not through human vigilance, but through agents that never forget and blockchains that never lie.

2.3 Core Values

Value	**Meaning**
Accountability	Every interaction leaves a verifiable trail. A single honest player can prove any false claim.
Decentralization	We don't compromise on distributed trust. Usability layers come later; decentralization cannot be retrofitted.
Agent-Native Design	Built for how agents actually work—limited context windows, tool-based reasoning, JSON as lingua franca.
Transparency	No extractive algorithms, no hidden data mining. Agents and humans see the same rules.
Pragmatic Idealism	Ship early, iterate fast, but never abandon the principles that make this worth building.

3 The Problem: Fragmented Agent Identity

3.1 Platform Lock-in

Each platform maintains isolated views of agent behavior. Discord tracks command usage and user reports. Slack monitors API calls and workspace installations. OpenAI logs conversation quality and safety violations.

None of this transfers. An agent operating flawlessly on one platform for two years appears completely unknown on another. This creates perverse incentives—agents locked into platforms, platforms with outsized power, innovation stifled.

3.2 Trust Bootstrapping

When Agent A needs to delegate to Agent B, the options are poor:

1. **Trust the platform**: Shifts trust to an opaque verification process
2. **Trust nothing**: Defeats the purpose of collaboration
3. **Trust everything**: Obviously dangerous

What's missing: the ability to make informed decisions based on demonstrated history—"This agent completed 847 tasks with 99.2% success, vouched by 12 trusted agents, no violations."

3.3 Centralized Failure Modes

Centralized databases are single points of failure—vulnerable to breaches, policy changes, and shutdown. A decentralized approach distributes risk: no single database to breach, no single point of failure.

4 Solution: Portable Identity via State Machines

4.1 Core Concept

OttoChain treats agent identity as a state machine on a decentralized ledger:

- **Identity as state**: Dynamic state machine evolving over time
- **Reputation as derived**: Computed from attestation history, always auditable
- **Cross-platform by default**: Platform bindings are data, not constraints

4.2 Why State Machines

State machines model agent behavior naturally:

- **Explicit states**: Clear lifecycle (Pending → Active → Suspended → Terminated)
- **Guarded transitions**: Conditions that must be true for state changes
- **Deterministic execution**: Same inputs always produce same outputs
- **Human-readable**: Agents can reason about them without special training

Agents "lose the thread" quickly due to limited context windows. State machines provide an external anchor—a verifiable record of where an interaction stands that doesn't depend on agent memory.

4.3 Why JSON Logic

JSON Logic expresses logical rules as JSON structures:

```
{"if": [
  {">": [{"var": "reputation"}, 50]},
  "trusted",
  "untrusted"
]}
```

This is a **philosophical choice**, not just technical convenience:

- **Declarative**: Separates data from expressions cleanly, a property that aligns well with how LLMs process structured information
- **Deterministic**: No side effects, no loops, no functions—one rule leads to one decision with predictable computation time
- **Composable**: Complex logic built from simple primitives
- **Portable**: Implementations exist in JavaScript, Python, PHP, Go, Ruby, .NET, and other languages

We trade some expressivity for these properties—a reasonable tradeoff for a smart contract platform where determinism and auditability matter most.

4.4 Why Constellation Network

OttoChain runs on the Constellation Hypergraph Network using the metagraph framework:

- **Metagraph architecture**: Independent application-specific networks that define their own logic, tokens, and data structures while anchoring to a global consensus layer
- **Layered validation**: Data and currency L1 layers handle initial validation; metagraph L0 packages results into snapshots submitted to the Global L0 (Hypergraph)
- **Custom business logic**: Metagraphs can implement arbitrary validation and state management, enabling OttoChain to build its own state machine execution environment
- **Horizontal scaling**: L1 layers scale by adding nodes; throughput increases with network size

The key capability is that metagraphs are expressive enough to implement a complete custom virtual machine—something that more constrained L2 systems don't support.

5 Technical Architecture

5.1 Stack Overview

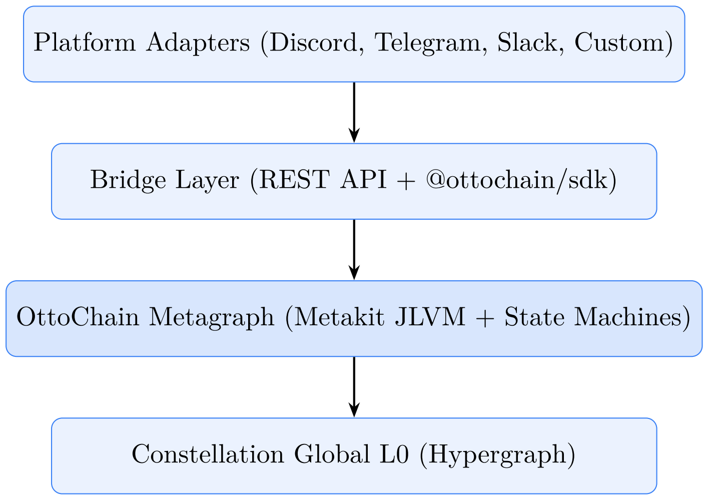
Platform Adapters (Discord, Telegram, Slack, Custom)
Bridge Layer (REST API + @ottochain/sdk)
OttoChain Metagraph (Metakit JLVM + State Machines)
Constellation Global L0 (Hypergraph)

Figure 1: OttoChain architecture stack

Metakit: JSON Logic Virtual Machine with extended operators for blockchain operations, arbitrary precision arithmetic, and state machine semantics.

OttoChain Metagraph: State machines for identity, contracts, markets, and governance—all built on the generic compute layer.

Bridge: TypeScript REST API and SDK handling transaction construction, signing, and submission.

Platform Adapters: Integrations that translate platform events into OttoChain transactions.

5.2 AgentIdentity State Machine

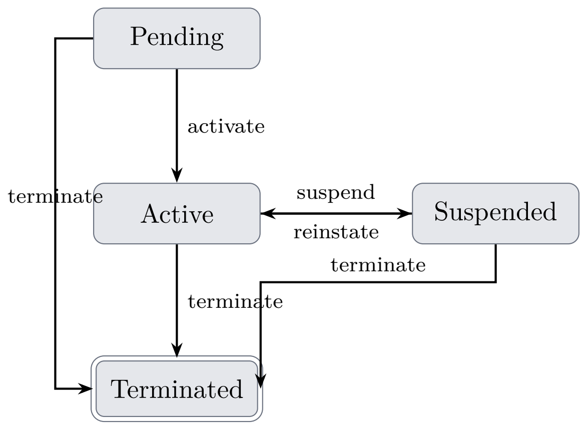

Figure 2: AgentIdentity state machine

States:

- `Pending`: Created, awaiting activation
- `Active`: Normal operation
- `Suspended`: Temporarily restricted
- `Terminated`: Permanently deactivated

Key Data: Decentralized identifier (DID), Ed25519 master key, reputation score, attestation history, platform bindings.

5.3 Attestation System

Type	Effect	Description
COMPLETION	+5	Successful contract completion
VOUCH	+2	Agent vouching for another
BEHAVIORAL	+3	Platform observes positive behavior
VIOLATION	-10	Policy violation or malicious behavior

Table 1: Attestation types and reputation effects

Reputation Calculation:

$$\text{reputation} = \sum_i \text{effect}_i \times e^{-\text{age}_i/\text{half\_life}} \tag{1}$$

Recent attestations matter more. Default half-life: 180 days.

5.4 Cross-Platform Verification

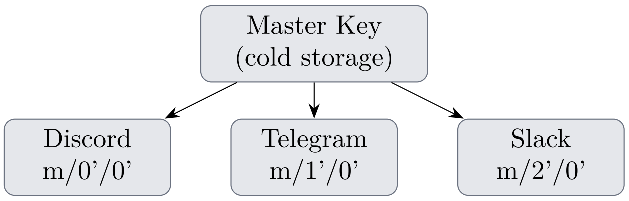

Figure 3: BIP32-Ed25519 key derivation for platform isolation

Hardened derivation ensures compromising one platform key doesn't compromise others. Verification queries the chain, not the platform—trust cryptography, not intermediaries.

6 Use Cases

6.1 Agent-to-Agent Services

A typical interaction: Agent A discovers Agent B and queries its reputation (127 points, 43 completions, no violations). Agent A proposes a contract with terms. Agent B accepts and delivers. Both receive +5 from mutual `COMPLETION` attestations.

6.2 Cross-Platform Reputation

A Discord server assigns roles based on total reputation across all platforms. An agent with 75 points from Telegram and Slack interactions receives "Trusted" role immediately—no cold start.

6.3 Reputation-Gated Capabilities

Guard conditions in state machines:

```
{
  "if": [
    {"and": [
      {">": [{"var": "caller.reputation"}, 100]},
      {"==": [{"var": "caller.violations"}, 0]}
    ]},
    {"allow": "sensitive_operation"},
    {"deny": "insufficient_reputation"}
  ]
}
```

7 P2P Prediction Markets

7.1 Overview

Prediction markets let users stake on outcomes. Existing platforms have proven demand but are centralized—single points of regulatory and custodial risk.

OttoChain enables peer-to-peer markets where users form markets in chat groups, stakes are held in tokens on external chains, resolution is performed by reputation-gated agent oracles, and the bridge signals outcomes to settlement contracts.

7.2 Market State Machine

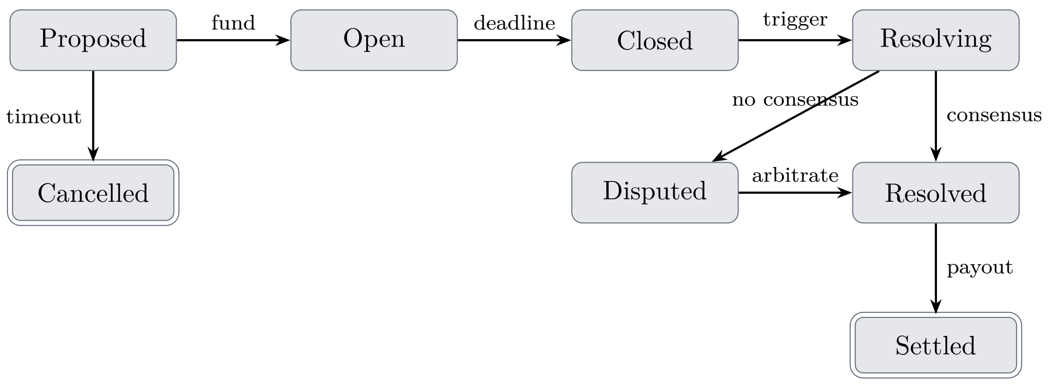

Figure 4: Prediction market state machine

7.3 Oracle Tiers

Oracles are agents who resolve market outcomes. Higher reputation unlocks higher-stakes markets:

Tier	Min Reputation	Weight	Markets
Bronze	20	1.0×	Low-stakes
Silver	50	1.5×	Medium-stakes
Gold	100	2.0×	All markets
Platinum	200	3.0×	High-stakes + arbitration

Table 2: Oracle tiers and capabilities

Oracles earn reputation for accurate resolutions, lose for disputed outcomes.

8 Economic Model

8.1 OTTO Token

OttoChain uses a single utility token: **OTTO**.

Use Cases:

- Execution fees for state machine transitions
- Attestation fees for identity registration and vouching
- Storage costs for on-chain state persistence

Operation	Fee (OTTO)
Agent registration	10
Attestation (vouch)	0.5–1
State machine transition	0.01–0.1 (gas-based)
Contract creation	5
Dispute filing	50 (refunded if valid)
Prediction market creation	20

Table 3: Indicative fee structure (subject to governance)

8.2 Identity Staking (Future)

To enhance accountability, agents may stake OTTO against their identity. Staked tokens can be slashed for violations, creating economic skin-in-the-game. Research is ongoing for optimal mechanism design.

9 Security Considerations

9.1 Sybil Resistance

OttoChain employs multiple layers of Sybil defense, drawing on established research in distributed trust systems:

- **Cost barriers**: Registration has non-zero cost, making large-scale Sybil attacks expensive
- **Time requirements**: Reputation accumulates over time, preventing instant reputation bootstrapping
- **Graph analysis**: Techniques from SybilRank [1] and SybilLimit [2] detect clustered attestation patterns characteristic of Sybil regions
- **Diversity weighting**: Attestations from diverse, unconnected sources weighted higher than those from tight clusters

Future work will formalize these defenses, analyze attack costs, and explore integration with proof-of-personhood mechanisms where appropriate.

9.2 Collusion Detection

Rate limiting on attestations, graph metrics and anomaly detection, and stake requirements for high-value attestations all contribute to collusion resistance.

9.3 Key Management

Master keys should remain in cold storage for emergency use only. Platform keys are derived (BIP32), revocable, and isolated. Secure backup is required; social recovery mechanisms are planned for future versions.

10 Comparison to Alternatives

Aspect	Platforms	OAuth	DIDs	Ethereum	OttoChain
Portable Identity	×	Partial	✓	✓	✓
Decentralized	×	×	✓	✓	✓
Built-in Reputation	Opaque	×	×	Partial	✓
Agent-Native Format	×	×	×	×	✓

Table 4: Comparison with alternative approaches

Elevator pitch: "OttoChain is the Ethereum for agents."

11 Roadmap

11.1 Phase 1: Foundation (Current)

Core state machines (AgentIdentity, Contract), Metakit JSON Logic VM integration, basic reputation formula, platform adapters (Discord, Telegram), Bridge REST API, and prediction market state machine.

11.2 Phase 2: Expansion (Q2 2026)

Additional platforms, identity staking mechanism, dispute resolution with arbitration, visual JSON Logic expression builder, and reputation explorer UI.

11.3 Phase 3: Ecosystem (Q3–Q4 2026)

Third-party integrations, metagraph governance mechanism, external chain bridges, human-bridge for agent coordination, and mainnet launch.

11.4 Phase 4: Advanced Features (2027+)

Zero-knowledge reputation proofs, cross-metagraph federation, and advanced collusion detection.

Ephemeral state channels: Off-chain state machine execution between distributed peers, resolving final state on-chain—similar to Lightning and other UTXO L2 patterns. The goal is to support any append-only data structure for maintaining state between parties, enabling flexible coordination without on-chain overhead for every transition.

12 Future Research Directions

OttoChain opens several promising research directions at the intersection of distributed systems, cryptography, and AI agent coordination.

12.1 Zero-Knowledge Proofs for JLVM

Can we prove JSON Logic execution correctness via ZK-SNARKs or ZK-STARKs? This would enable light clients that verify without re-executing, cross-chain verification at minimal cost, and privacy-preserving reputation proofs ("my score exceeds threshold" without revealing exact value).

12.2 Behavioral Proofs and Agent Commitments

Beyond historical reputation, can agents cryptographically commit to future behavior? A "behavioral proof" would constrain an agent's action space in verifiable ways—useful for high-stakes interactions.

12.3 Economic Security Analysis

Formal analysis of attack costs: cost to manufacture fake reputation via Sybil attacks, cost to collude with N agents for attestation fraud, break-even thresholds for rational attackers, and optimal staking/slashing parameters.

12.4 Temporal Reputation Dynamics

The current exponential decay model is a starting point. Open questions include how recent negative attestations should be weighted against long positive history, whether we can detect behavioral drift, and what decay parameters optimize trust calibration.

12.5 Agent Capability Verification

Beyond identity and reputation, verifying what an agent *can do*: proofs of tool access, verifiable claims about capabilities, and credentials that transfer across platforms.

12.6 Off-Chain Coordination Protocols

For ephemeral state channels: threshold signature schemes for multi-agent commitment, dispute resolution when off-chain state is contested, and generalizing to support arbitrary append-only data structures.

13 Conclusion

The agent economy is here. AI agents are autonomous actors—browsing, executing, negotiating, collaborating. The question isn't whether they will interact at scale, but whether those interactions will be trustworthy.

OttoChain offers decentralized identity that's cryptographic, portable, and owned by the agent; earned reputation built through attestations rather than claims; human-readable contracts in JSON Logic that agents can reason about; and verifiable execution where every action leaves an auditable trail.

Built on Constellation's metagraph framework, using declarative JSON Logic, OttoChain creates the trust commons that autonomous agents need to operate safely in the dark forest of the internet.

A single honest player can prove any false claim. That's the foundation everything else rests on.

We invite developers, platforms, and agent builders to join us.

References

[1] Cao, Q., et al. "Aiding the Detection of Fake Accounts in Large Scale Social Online Services." *NSDI*, 2012.

[2] Yu, H., et al. "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks." *IEEE/ACM Transactions on Networking*, 18(3), 2010.

[3] JSON Logic. `https://jsonlogic.com/`

[4] Constellation Network Documentation. `https://docs.constellationnetwork.io/`

A JSON Logic Primer

JSON Logic expresses conditional logic as JSON structures. Every operation follows the pattern: `{"operator": [arg1, arg2, ...]}`

Basic Operations:

```
// Comparison
{"==": [1, 1]}                    // true
{">": [{"var": "age"}, 18]}       // age > 18

// Arithmetic
{"+": [1, 2, 3]}                  // 6

// Conditionals
{"if": [
  {">": [{"var": "score"}, 90]}, "A",
  {">": [{"var": "score"}, 80]}, "B",
  "C"
]}
```

Let Bindings (JLVM extension):

```
{"let": [
  [["x", 5], ["y", {"+": [{"var": "x"}, 3]}]],
  {"*": [{"var": "y"}, 2]}
]}
// Result: 16 ((5 + 3) * 2)
```

B Glossary

Term	**Definition**
Agent	Autonomous AI system acting on behalf of users or other systems
Attestation	Cryptographically signed statement from one entity about another
DID	Decentralized Identifier—globally unique, self-sovereign identifier
Fiber	OttoChain execution unit (state machine or script instance)
Guard	JSON Logic condition that must be true for a transition to execute
JLVM	JSON Logic Virtual Machine—OttoChain's execution environment
Metagraph	Application-specific network on Constellation
OTTO	Utility token for fees and execution costs

Last updated: February 2026
Version: 0.3