Fusio

A Decentralized Protocol for Autonomous Agent Compute

Ahmed Ali (*Lord Amdal*)

amdal.ali@oit.edu

March 2026

Abstract

The agora was the central public space of ancient Greek cities, a marketplace and meeting place governed by shared rules, open participation, visible transactions, and accountable actors with no single point of control. We are building its equivalent for the AI economy. **Fusio** is an open protocol that connects a *requester* (a person with an AI model and a task) with a *worker* (a person who contributes their computer to run that task). The exchange is governed by four non-overridable primitives: verified identity, matched routing, settled payment, and permanent receipt. The protocol is open source and owned by no entity. Any AI agent operating on Fusio can act, pay, and be held accountable without a human in the loop.

Keywords: autonomous agents, decentralized compute, agentic infrastructure, browser automation, cryptographic accountability, token economics, AI protocols.

This document is published for review and comment by the technical community. It does not constitute an offer to sell securities. The FSO token described herein is a utility token. Readers should seek independent legal and financial advice before acting on any information contained herein.

1. Introduction

1.1 The Original Agora

The Greeks understood something most infrastructure builders forget: a marketplace is not merely an economic mechanism; it is a public institution. It functions because everyone can see what is happening, everyone knows the rules, and everyone is accountable to everyone else. The agora was governed by shared rules, not controlled by any merchant or guild. That model scaled across centuries and cultures because it was built on open participation, visible transactions, accountable actors, and no single point of control. The AI economy needs the same foundation.

1.2 The Problem

The history of computing is a history of collapsing asymmetries. The personal computer gave individuals the power of institutions. The internet gave individuals the reach of publishers. Cloud computing gave individuals the infrastructure of enterprises. We are at the beginning of another such shift: AI models are becoming capable enough to perform knowledge work that previously required teams of people.

However, there is a ceiling. AI models can think but they cannot act autonomously in the world. They cannot open a browser and run a job for hours while the user is asleep, pay for compute on behalf of their user, prove what they did, or maintain persistent identity. Every agentic system built today requires a human to stay in the loop: to hold the credentials, manage the infrastructure, and babysit execution. This is not a limitation of the models; it is a limitation of the infrastructure underneath them.

1.3 The Concrete Case

Consider someone building a personal brand who wants an AI agent to: monitor social media daily, identify trending topics, write articles in response, publish them to their blog, and engage with comments. AI models available today are capable of each of these subtasks. Yet to run this autonomously, the person must leave their own computer on, expose credentials to whatever process is running, pay for compute from personal infrastructure, and operate with no verifiable record of agent actions. The friction is prohibitive and the trust model is unclear.

This pattern repeats across any autonomous task: managing business calendars, monitoring competitors, processing incoming requests, executing research pipelines. Every one hits the same ceiling. Fusio collapses this asymmetry.

1.4 What Fusio Provides

Fusio is not an AI model. It is not an application. It is a protocol: a set of rules allowing different participants to interact predictably and trustworthily. The protocol defines four things precisely (see table 1):

1. How an AI agent establishes a verified identity on the network;
2. How a job is routed from a requester's AI model to a worker's computer;
3. How the worker is paid in FSO when the job completes;
4. How a signed receipt is written to an immutable on-chain log.

1.5 Why Now

Three conditions align today that did not exist two years ago:

- **Model capability:** The reasoning required to plan and execute multi-step browser tasks exists in commercially available models. Model capability is no longer the binding constraint; infrastructure is.
- **Idle compute:** Hundreds of millions of personal computers sit idle for large portions of the day, aggregating compute that dwarfs any data center. A protocol enabling this compute to be safely contributed to a network is the missing layer.
- **Behavioral readiness:** People already rent their homes (Airbnb), cars (Turo), and time. Contributing idle compute to a network in exchange for tokens is a natural extension, provided the trust and safety model is credible.

2. Protocol Design

Fusio is designed around a single core insight: *identity and compute are not separable in a trustworthy agentic system.* You cannot have verifiable compute without knowing who authorized it; you cannot have accountable identity without knowing what actions it took. The protocol treats these as a unified primitive.

2.1 Participants

Three participant classes interact in the protocol, as illustrated in fig. 1.

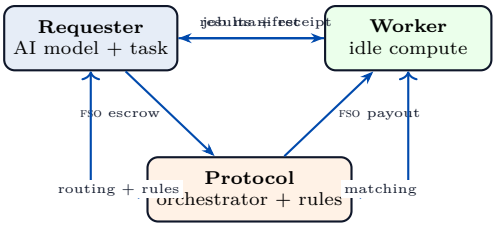

Figure 1: *Three participant classes and their interactions within the Fusio protocol.*

- **Requester:** Connects an AI model to the Fusio network, submits jobs, pays in FSO, and receives results with signed receipts. The protocol does not touch the model, conversation, or user data.
- **Worker:** Runs the Fusio node software, contributes isolated compute and a residential IP address, earns FSO. Cannot observe job execution.
- **Protocol / Foundation:** Sets rules, runs the orchestrator, validates completions, issues receipts, manages token economics. Does not control individual jobs or user data.

2.2 The Four Protocol Primitives

Every network interaction flows through the four primitives in table 1. These are the *only* things the protocol standardizes; everything else is implementation.

Table 1: *The Four Protocol Primitives*

Primitive	Standardization Scope
Agent Identity	Cryptographic keypair at registration. Public key is the on-chain address. Every job is signed by the requester's agent key; every worker receipt by the worker's key. Neither party can repudiate participation.
Job Routing	Structured job manifest declaring required capability, budget, duration estimate, and stealth requirement. Orchestrator matches manifest to available workers and confirms before any job starts.
Settlement	EVM smart contract holds requester's FSO in escrow at job start. Signed completion receipt triggers release. On failure, escrow is returned per fault classification.
Accountability	Signed receipt containing: agent ID, worker ID (hashed), job hash, timestamps, action count, outcome, and cost. Written to an append-only on-chain log; cannot be altered or deleted.

2.3 What the Protocol Does Not Standardize

Fusio deliberately leaves the AI model, worker operating system, and application layer to implementers. A protocol that standardizes everything becomes a product. Fusio is a foundation upon which many products can be built, not a product itself.

3. How a Job Works

fig. 2 provides the complete job lifecycle. The following subsections detail each phase using a running example: an AI agent that researches trending topics, writes a blog article, and publishes it.

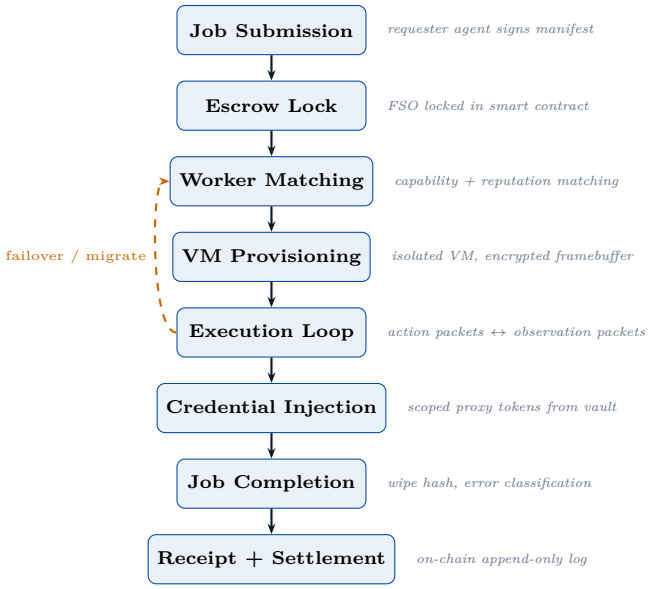

Figure 2: *Complete job lifecycle in the Fusio protocol. Orange dashed arrow indicates the migration path on worker node failure.*

3.1 Job Submission

The requester's AI model signs a structured job manifest with the requester's agent private key. The orchestrator validates the manifest (identity record, category prohibition list, balance sufficiency) and locks the declared amount in escrow.

```
job_manifest {
  job_id:           uuid,
  agent_id:         requester_public_key,
  agent_signature:  signs_entire_manifest,
  capability:       'browser.full',
  stealth_required: 'vm_stealth',
  estimated_minutes: 45,
  max_price_fso:    2.00,
  browser:          'chromium',
  declared_purpose: 'content research and
      publishing',
  category:         'content.creation',
  memory_policy:    'persistent',
  failover_policy:  'migrate'
}
```

Listing 1: *Job Manifest Structure*

3.2 Worker Matching

Workers continuously broadcast capability profiles to the orchestrator. The orchestrator matches job manifests against available profiles on three axes: (1) declared capability, (2) reputation score, and (3) price. On confirmation, the worker's node software spins up an isolated VM. The worker receives no information beyond what is needed to provision the VM: not the declared purpose, requester identity, or job instructions.

3.3 The Execution Loop

Execution proceeds as a request/observation loop (fig. 3) until job completion, budget exhaustion, or a failure condition.

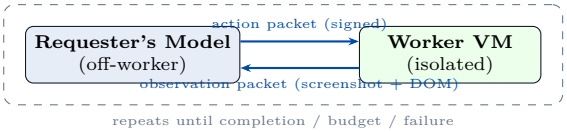

Figure 3: *The execution loop. The requester's model never runs on the worker's machine; only action instructions and screen observations cross the boundary.*

The protocol packet formats are defined as follows:

```
action_packet {
  job_id:   uuid,
  step:     integer,
  action:   'click' | 'type' | 'navigate'
          | 'scroll' | 'screenshot' | 'wait',
  target:   {x: number, y: number} | url,
  value:    string | null,
  signed_by: requester_agent_key
}

observation_packet {
  job_id:    uuid,
  step:      integer,
  screenshot: base64_image,
  dom_state:  compressed_semantic_description,
  signed_by:  worker_key
}
```

Listing 2: *Action and Observation Packet Formats*

3.4 VM Isolation

The VM enforces isolation at four independent layers, summarized in table 2. Isolation is enforced by architecture, not policy.

Table 2: *VM Isolation Layers*

Layer	Mechanism
Screen output	Encrypted framebuffer; worker host OS cannot read screen content
Network	All traffic exits through encrypted tunnel to Fusio relay before the open internet; worker cannot observe visited sites
Credentials	Injected into a hardware enclave; never appear in readable memory outside the enclave
Disk	Ephemeral; cryptographically wiped on job completion; wipe hash included in completion receipt

Workers observe only: CPU usage, RAM usage, job duration, and FSO earnings.

3.5 The Browser Identity Stack

Jobs interacting with automation-detection sites require a browser environment that appears as a real user session. The protocol's browser identity stack applies the following to every VM session:

- User-agent string reflects a current real browser release
- Platform string, screen resolution, timezone, language, and hardware concurrency drawn from the worker's *actual* host machine (real values, not fabricated)
- WebDriver flag explicitly disabled
- Canvas fingerprint entropy randomized per session but stable within a session
- Destination sites see a real residential IP via the worker's connection through the encrypted tunnel

This approach is more resilient than fabricated fingerprints because the underlying hardware signals are genuine.

3.6 Job Memory and State

Long-running jobs persist state across steps through three memory tiers:

Table 3: *Job Memory Tiers*

Tier	Description
Working memory	Agent context within a single execution session; held in encrypted VM memory
Step memory	Structured snapshot of job progress written to the Fusio relay at configurable intervals; enables migration without restart
Agent memory	Persistent storage owned by the requester's agent identity; maintained across jobs; accumulates knowledge about requester preferences and history

3.7 Failover and Migration

The orchestrator monitors a heartbeat signal from every active job. Absence for 30 seconds triggers migration: the orchestrator loads the most recent step snapshot, finds a new eligible worker, and resumes from the last confirmed step. *Credentials are never included in the step snapshot.* The new worker receives a fresh credential token issued by the vault; the original token is revoked.

4. Credentials and the Front Desk Agent

Handling credentials in a decentralized system where workers are unknown and untrusted is the hardest security problem the protocol faces. Passing credentials in the job manifest is unacceptable: workers would see plaintext secrets.

4.1 The Vault Model

The vault is controlled by the requester's principal identity. It never releases raw credentials to workers. Instead, it issues *short-lived scoped proxy tokens* at job start; each token grants access to a specific service, for a specific scope, for the duration of a specific job. The token lifecycle is illustrated in fig. 4.

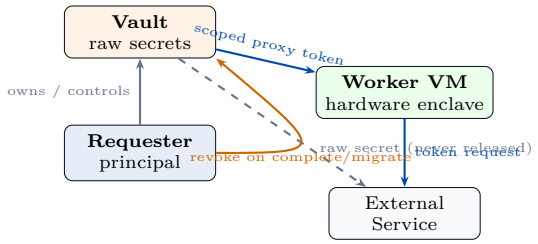

Figure 4: *Vault model: raw credentials are never released to workers. Scoped proxy tokens expire with the job; on migration, the old token is revoked before the new one is issued.*

If a worker's machine is compromised, the attacker obtains only an expired scoped token tied to a job that no longer exists.

4.2 The Front Desk Agent

The front desk agent is a lightweight onboarding flow built into the Fusio client. Before a credentialed job starts, it checks the vault. If credentials are missing, it presents a guided setup flow:

- **OAuth services:** Opens the provider's consent screen in the requester's browser. The access token goes directly to the vault without passing through intermediate application state.
- **API key services:** The requester pastes the key into an encrypted input field. It is encrypted on entry and stored in the vault. The raw key never appears in readable application memory.

The front desk agent validates each credential with a test call before job start. This prevents spending FSO on jobs that would fail immediately due to credential problems.

5. Error Handling and Worker Protection

5.1 Error Classification

Every job failure is classified by fault before any penalty or refund is applied. table 4 specifies the five fault classes and their consequences.

Table 4: *Error Classification and Consequences*

Class	Example	Consequence
Worker fault	Node crashes mid-job	Worker reputation penalty. Job migrates. Partial FSO returned.
Requester fault	Bad instructions cause infinite loop	No worker penalty. FSO spent for time used.
External	Site blocks browser session	No penalty to either party. Job requeued. Requester refunded.
Environmental	Site UI changed mid-job	No penalty. Logged as environmental failure. Requester refunded.
Credential	OAuth token expired during job	No worker penalty. Job paused. Requester notified to refresh.

Classification is determined by the error packet submitted at termination, containing: failure class, step at failure, screenshot of screen state, and recovery recommendation.

5.2 Worker Legal Protection

A worker contributing compute to the network is a passive infrastructure provider. They do not author job instructions, cannot see job purpose, and cannot observe execution, enforced by protocol architecture, not policy.

Every job is cryptographically signed by the requester's agent identity. The signature appears in the job manifest, in every action packet, and in the completion receipt. The evidence chain in the protocol audit log leads to the requester. The worker's contribution to that job is documented solely as compute provision. Workers agree to participation terms that establish their role as compute providers and commit them to cooperating with valid legal process; in return, the protocol commits to producing requester identity evidence should a worker's compute be implicated in a

harmful job.

Note: This framework is designed to mirror the legal protections cloud compute providers enjoy in most jurisdictions. Workers should review participation terms for their specific legal jurisdiction.

5.3 Prohibited Job Categories

The orchestrator enforces a pre-execution prohibition list:

- Jobs targeting financial institution login flows
- Jobs submitting PII belonging to third parties
- Jobs interacting with domains associated with known fraud patterns
- Jobs executing high-value purchase flows without explicit requester authorization

This screening is logged and cryptographically provable; workers are protected before execution begins.

6. Token Economics

The FSO token is the native utility token of the Fusio protocol. Its primary purpose is settling economic transactions between requesters and workers globally, instantly, and without a common financial intermediary. The name "Fusio" derives from the Latin extitfusio (fusion, convergence, a merging of flows) and is the common medium that allows strangers to transact without a bank, platform, or middleman.

6.1 Why a Native Token

Job sizes on the Fusio network are small: a 15-minute browser session may cost fractions of a cent in compute terms; a long-running research job a few dollars. These transaction sizes are impractical in fiat currency: payment processing fees would exceed job costs for small jobs, and international wire transfers are too slow and expensive for micropayments to workers in other countries. A native token settles instantly, globally, and at negligible per-transaction cost.

6.2 Token Utility

FSO utility flows through four channels:

- **Requesters** acquire FSO to pay for jobs
- **Workers** earn FSO by completing jobs
- **Service providers** stake FSO to list capabilities (stake slashed on consistent unavailability)
- **Foundation** earns FSO through network fees on settled transactions

FSO is not a speculative instrument. Its value is grounded in network utility: as more jobs run, more FSO is required. Token value tracks genuine network activity rather than speculation about future activity.

6.3 Minting and Supply

The minting schedule is defined in the protocol specification and controlled by the foundation within hard-coded limits. Changes to the minting schedule require a protocol upgrade following the governance process (section 8). Network fees split between the foundation treasury and a protocol reserve, which funds node operator incentives during early growth when organic job volume is insufficient to attract workers without supplemental rewards.

6.4 Worker Reward Formula

Worker earnings are calculated from three factors:

$$\text{FSO}_{\text{earned}} = R_{\text{base}} \times M_{\text{rep}} \times F_{\text{complexity}} \tag{1}$$

where:

- R_{base} is the base job rate agreed in the manifest at match time
- $M_{\text{rep}} \in [0.8,\ 1.2]$ is the reputation multiplier (0.8 for new workers; 1.2 for established workers with strong completion records)
- $F_{\text{complexity}}$ accounts for job duration and total action count

New workers start at $M_{\text{rep}} = 0.8$ and earn upward through successful completions, creating a natural quality gradient in the worker pool without manual review.

7. The Application Layer

The protocol defines the rules. The application is what people actually use. The first application built on the Fusio protocol is the Fusio Client, operated by the founding organization. The protocol is designed so that any party can build additional applications on it.

7.1 The Fusio Client

The Fusio client is a chat application. The requester connects it to any AI model (Claude, ChatGPT, Gemini, a locally running model, or any model exposing a compatible API). They describe their goal in natural language. When the model determines that the task requires browser automation or off-device compute, it submits a job to the Fusio network. The requester need not understand the protocol.

7.2 Worker Mode

The same client application runs in worker mode. The operator configures: how much compute to contribute, availability hours, and accepted job categories. The application runs the Fusio node software in the background. Jobs are matched and executed automatically. FSO earnings accumulate in the worker's wallet. A worker who leaves their machine available overnight

earns meaningful FSO without active involvement beyond initial setup.

7.3 Live Session Visibility

Requesters can watch jobs in real time. The Fusio client provides a live session view streaming the job screen from the worker's VM over an authenticated WebRTC connection [6]. The stream is not recorded, not stored, and is available only to the authenticated requester for the duration of the job.

Requesters may request control of the session: the agent pauses, keyboard and mouse inputs are forwarded to the VM, and on handback the agent re-observes the current screen state and continues its plan from where it left off, incorporating the requester's changes.

8. Governance

The Fusio protocol is not owned by its founding organization. The founding organization is the initial maintainer and operator of the reference implementation and the first hosted network. The protocol itself belongs to no one, and the rules for changing it are defined in the protocol rather than controlled by any single party.

8.1 The Foundation

The Fusio Foundation is an independent nonprofit stewarding the open source protocol. Funded by network fees and the foundation's allocation of FSO from the token treasury. Its mandate: maintain the protocol specification, publish the reference implementation, fund core engineering, and coordinate the upgrade process. The foundation does not control individual jobs, user data, worker nodes, or the FSO token price.

8.2 Protocol Upgrades

Any participant may propose a protocol upgrade by submitting a Protocol Improvement Proposal (PIP) to the foundation's public repository. The process is:

1. Technical review by the core engineering team for soundness
2. Minimum 30-day public comment period upon passing technical review
3. Vote by the foundation's technical committee
4. Approved proposals activated at a scheduled block height

Changes to the token minting schedule, fee structure, or governance process itself require a supermajority of the technical committee and a 60-day comment period; these parameters are treated as constitutional.

8.3 Transition to Community Governance

The foundation model is appropriate for a new protocol requiring clear leadership and fast decision-making in response to real usage. The Fusio Foundation intends to transition toward token-holder governance as the protocol matures, driven by ecosystem readiness rather than an arbitrary schedule. The foundation will publish a governance maturity framework defining conditions for each stage of decentralization.

9. Competitive Context

table 5 positions Fusio relative to the most relevant existing systems. The combination Fusio offers (decentralized browser-based agent compute with identity, settlement, and accountability built into the protocol) has no direct equivalent today.

Table 5: *Competitive Landscape*

System	Relationship to Fusio
LangGraph CUA + Scrapybara	Closest existing pattern. LangGraph provides the agent loop; Scrapybara provides centralized cloud VMs. Fusio replaces Scrapybara with a decentralized worker network and adds identity, settlement, and accountability that this pattern lacks.
Akash Network	Decentralized compute for containers and servers. No browser automation, no agent identity, no job-level accountability. Fusio is specifically designed for the browser-based agentic use case.
AWS + browser automation	Centralized, expensive for small jobs, no worker economy, no token settlement, not accessible to the public as a compute contributor.
Golem / iExec	Decentralized compute marketplaces for general computation. Not designed for browser-based agent tasks; no action packet protocol or browser identity stack equivalent.
0G Labs	Decentralized AI infrastructure focused on inference, storage, and on-chain settlement. Fusio's specific contribution is the browser-based computer-use layer and the human worker economy that makes real residential IPs and real machine fingerprints available.

The closest alternatives are either centralized, not designed for agents, or missing the accountability layer that makes autonomous agent actions safe enough to trust at scale.

10. Roadmap

The roadmap is organized around questions each phase must answer, rather than feature lists. fig. 5 provides a visual overview.

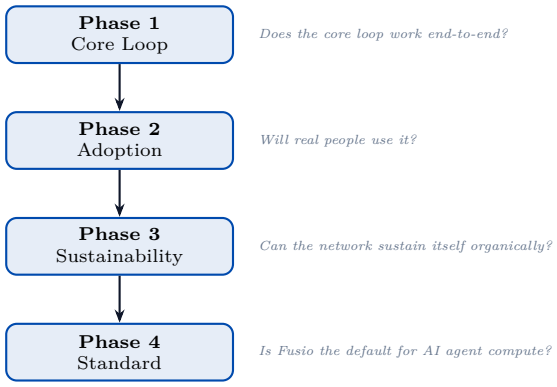

Figure 5: *Four-phase roadmap organized by the question each phase must answer.*

10.1 Phase 1: Does the Core Loop Work?

Goal: a working end-to-end demonstration. One requester, one AI model, one job, one worker, one signed receipt. The target demo is blog research and publishing, complex enough to validate the protocol under real conditions. Deliverables: four protocol primitives in the reference implementation; Fusio orchestrator on founding-organization infrastructure; node software installable on Linux and macOS; action packet format finalized and published; FSO token on testnet; demo recorded and published.

10.2 Phase 2: Will People Use It?

Goal: real users running real jobs. Founding-organization worker nodes provide baseline availability while the organic worker community grows. The client application is polished for non-technical users; the front desk agent handles credential onboarding without requiring OAuth knowledge; worker onboarding completes in under five minutes. Deliverables: public beta client; worker onboarding flow; live session viewer; reputation system active; FSO on mainnet; first ten third-party jobs run by external parties.

10.3 Phase 3: Can the Network Sustain Itself?

Goal: organic worker participation exceeds founding-organization node capacity. Job categories expand beyond browser automation to other compute types built by the community. Deliverables: agent-to-agent job chaining; enterprise compliance tier with enhanced audit exports; open governance transition beginning; Protocol Improvement Proposal process active with meaningful community participation.

10.4 Phase 4: Is Fusio the Standard?

Goal: Fusio becomes the default method for AI agents to request and execute compute. This requires: integration into major AI frameworks enabling developers to reach the Fusio network without understanding the protocol directly; agent identity standard adopted beyond the Fusio network so identity and history are portable across systems; governance sufficiently decentralized that the protocol's future does not depend on any single organization.

11. Conclusion

The agora worked because it was designed around a simple truth: when people can transact in the open, under shared rules, with identities known and actions visible, trust scales, commerce grows, and the community becomes more than the sum of its members. The AI economy is at the beginning of the same kind of moment. The models exist. The compute exists. The demand is real and growing. What is missing is the open space where it can all come together safely, fairly, and at scale.

Fusio is that space: four primitives, open source, governed by a foundation that intends to decentralize as the ecosystem matures. The token economics are grounded in genuine utility. The worker protection model is as strong as the law allows. The privacy guarantees are enforced by architecture rather than policy. Real people getting paid for compute they already own. Real agents doing real work in the world.

Any AI agent can act, pay, and be held accountable, without a human in the loop.

Acknowledgments

The author acknowledges the open-source communities behind the Ethereum protocol, LangChain/LangGraph, Playwright, and the broader decentralized compute research ecosystem, whose prior work provides the technical foundation upon which Fusio is built.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. https://bitcoin.

org/bitcoin.pdf

[2] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," 2014. https://ethereum.org/whitepaper/

[3] LangChain AI, "LangGraph Computer Use Agent," GitHub, 2025. https://github.com/langchainai/langgraph-cua-py

[4] U.S. Congress, "Clarifying Lawful Overseas Use of Data (CLOUD) Act," Public Law 115-141, 2018. https://www.congress.gov/bill/115th-congress/house-bill/4943

[5] Intel Corporation, "Software Guard Extensions Developer Guide," 2023. https://www.intel.com/sgx

[6] World Wide Web Consortium, "WebRTC 1.0: Real-Time Communication Between Browsers," 2024. https://www.w3.org/TR/webrtc/

[7] Playwright Project, "Browser Automation Documentation," 2025. https://playwright.dev

[8] M.H. Hansen, *Polis: An Introduction to the Ancient Greek City-State*. Oxford University Press, 2006.

A. Glossary

Term	Definition
Agent	An AI system operating on the Fusio network with a registered identity, a wallet, and the ability to submit and pay for jobs.
Principal	The human or organization that owns an agent and is legally accountable for its actions.
Requester	A principal who submits jobs to the Fusio network for execution on worker nodes.
Worker	A person who runs the Fusio node software and contributes their computer's resources to the network in exchange for FSO.
fso	Fusio token. The native EVM-based utility token used for all economic transactions on the protocol: job payments, worker earnings, provider staking, and network fees.
Job manifest	The signed document submitted by a requester that declares what a job needs and authorizes its execution.
Action packet	The structured instruction format transmitted from the requester's AI model to the worker's VM defining a single browser or computer action.
Observation packet	The structured response from the worker's VM containing a screenshot and semantic screen description.
Front desk agent	The onboarding flow that collects and validates credentials before a job starts.
Step snapshot	A serialized record of a job's progress written to the relay at regular intervals, enabling migration without restarting from the beginning.
Vault	The encrypted credential store controlled by the requester's principal identity. Issues scoped proxy tokens to workers rather than raw credentials.
Escrow	The smart contract that holds a requester's FSO during job execution and releases it to the worker on verified completion.
Wipe hash	A cryptographic proof included in the worker's completion receipt confirming that the VM disk was securely destroyed after the job ended.
Reputation score	A normalized score computed from a worker's history of successful job completions. Maps to an earnings multiplier $M_{\text{rep}} \in [0.8,\ 1.2]$: new workers start at 0.8 and earn upward through verified completions.
Protocol Improvement Proposal (PIP)	A formal document proposing a change to the Fusio protocol specification, subject to the governance process in section 8.

B. Action Packet Reference

The following action types are defined in the initial protocol specification. Additional types may be added through the PIP process.

`Action`	Parameters	Description
`navigate`	`url: string`	Navigate the browser to a URL.

`Action`	Parameters	Description
`click`	`x: number, y: number`	Click at the specified screen coordinates.
`type`	`value: string`	Type text at the current cursor position.
`scroll`	`direction: up|down|left|right, amount: number`	Scroll the page in the specified direction by the specified amount.
`screenshot`	*(none)*	Capture the current screen state and return an observation packet.
`wait`	`ms: number`	Wait for the specified number of milliseconds.
`key`	`key: string`	Press a keyboard key or combination.
`select`	`x: number, y: number, value: string`	Select a dropdown option at the specified coordinates.
`hover`	`x: number, y: number`	Move the mouse to coordinates without clicking.