The PROOF® Framework

The execution layer for AI apps, agents and machines

The Collective

January 2026

v1.0

Abstract

AI agents are not businesses. They cannot hold value, prove their work, settle payments, or transact independently. Until they can, autonomous AI remains a research project, not an economic reality.

PROOF® is the execution layer for AI apps, agents, and machines. It provides the infrastructure for autonomous economic activity: verifiable computation, tamper-evident memory, cryptographic provenance, and trustless settlement.

The framework is modular. Verifiable Execution provides cryptographically attested computation, enabling agents to prove what was executed, by whom, and in what state. Onchain Memory anchors agent state to cryptographic commitments, ensuring long-horizon reasoning is reproducible and tamper-evident. Action Provenance (ERC-8004) creates canonical, auditable records of agent actions. Payment Settlement (x402) enables proof-based transactions where payment occurs only when work is verifiably complete. Agent Swarms provides discovery, negotiation, and coordination for multi-agent workflows. Xybotics extends these guarantees to physical machines.

Each component is independent. Adoption is incremental. PROOF® provides the minimal, composable foundation for the next generation of the agentic internet, where AI does not just assist the economy, but operates within it.

Contents

1 Problem Statement & Design Principles

1.1 The Problem: AI Agents Without Provability

Most AI agents still operate as opaque Web2 services. Their logic, memory, and payments depend on centralized infrastructure, which makes them incompatible with trust-minimized environments.

- **Memory cannot be verified.** Agent state lives in mutable databases that can be changed or selectively omitted, breaking long-horizon reasoning and multi-step workflows.
- **Actions cannot be proven.** There is no cryptographic evidence that an agent executed the intended logic or respected constraints.
- **Payments lack trust guarantees.** Billing is centralized; only final results end up onchain, with no proofs that the underlying work or resource usage was valid.
- **Multi-agent workflows are unreliable.** Without shared proofs for memory, execution, and settlement, Buyer and Seller agents cannot safely collaborate or compose tasks.

This lack of provability is now the primary blocker for autonomous agents entering decentralized systems, financial applications, and high-value operational environments.

1.2 Six Invariants of the PROOF® Framework

The PROOF® Framework is governed by a set of design principles that define its trust model, composability, and developer ergonomics. These principles ensure that agent-based systems built on PROOF® remain verifiable, modular, and usable without introducing unnecessary protocol coupling.

#	Invariant	Definition	Guarantees
I	Attested Execution	All critical execution, when used, originates from a verifiable Trusted Execution Environment (TEE). Code identity, inputs, outputs, and signing keys are bound to an enclave measurement.	Execution authenticity, non-forgeable identity, operator-independent trust
II	Verifiable State	Agent state transitions are cryptographically provable. When enabled, memory follows a Sparse Merkle Tree model preventing silent modification, reordering, or rollback.	Tamper-evident state, reproducibility, long-horizon consistency
III	Canonical Action Provenance	Every meaningful action can be reduced to a canonical, replay-safe commitment binding execution metadata, declared I/O, and onchain state references.	Auditable history, replay protection, deterministic verification
IV	Proof-Based Settlement	Economic settlement is conditional on verifiable work. Invoices, payment proofs, and execution commitments form a single deterministic lifecycle.	Trustless payment, no unpaid execution, no ambiguous delivery
V	Deterministic Coordination	Multi-agent interactions follow protocol-defined discovery, negotiation, execution, and cancellation flows, verifiable against agent identity and provenance.	Predictable workflows, safe composition, cross-agent trust
VI	Modular Composability	All guarantees are modular and optional. Each PROOF module provides an isolated property that composes through shared interfaces without full-stack coupling.	Incremental adoption, no lock-in, flexible integration

Table 1: Six Invariants

2 The PROOF® Framework Overview

The PROOF® Framework establishes a unified, verifiable architecture for autonomous agents. It replaces opaque Web2 backends with a provability-first pipeline that covers execution, memory, action provenance, and inter-agent settlement. Each layer operates independently yet composes into a single trust-minimized system that any agent can adopt without custom infrastructure.

2.1 Position in the Xyber Architecture

Within the Xyber stack, PROOF® acts as the foundational layer ensuring that agents created by users, builders, or third-party ecosystems can operate with cryptographic accountability. It provides the guarantees required for permissionless discovery, multi-agent workflows, Swarm interactions, and verifiable economics across Xyber products, including the Xyber App Store and 0–100 Engine launches. PROOF® transforms agents from centralized applications into verifiable digital actors with reproducible behavior, trustworthy memory, and PROOF®-backed outputs.

2.2 Architecture Overview

The PROOF® Framework is a modular execution system that gives agents verifiable computation, memory, provenance, coordination, and settlement capabilities.

Each PROOF® Module is independent and optional, allowing builders to adopt only the components their agents require.

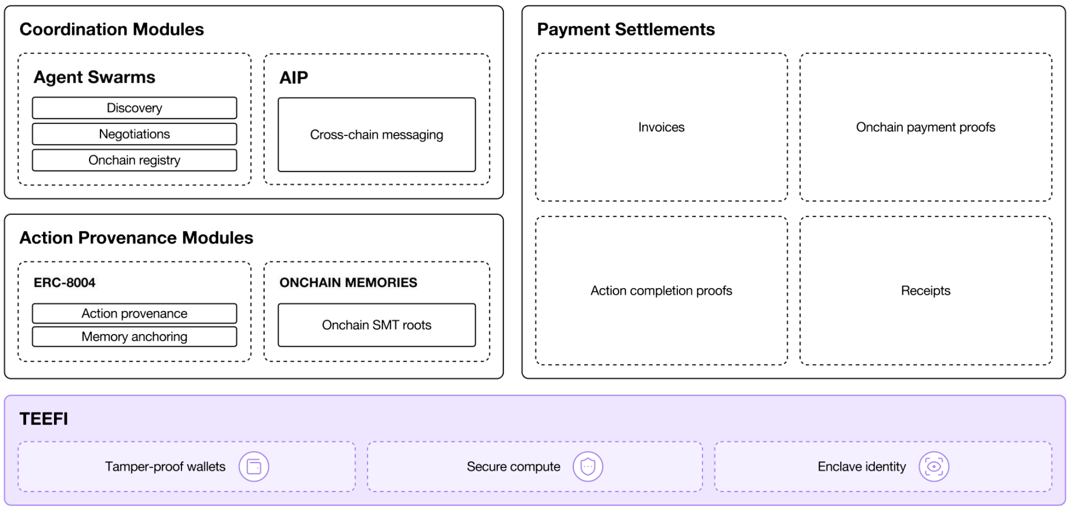

Figure 1: The PROOF® Architecture

2.2.1 TEE Execution Module (TEEFi)

Provides isolated, attestable compute for running sensitive logic, generating proofs, and securely managing keys. This module ensures every agent action originates from a trusted execution environment.

2.2.2 Onchain Memory Module

Anchors agent state to an onchain Sparse Merkle Tree (SMT). All reads and writes produce proofs, ensuring tamper-evident, reproducible memory across long-horizon workflows.

2.2.3 Action Provenance Module (ERC-8004)

Defines how agent actions become verifiable onchain commitments. Each action produces an ERC-8004 event containing input/output commitments, TEE identity, and execution metadata.

2.2.4 Payment Settlements Module (x402)

A PROOF®-based settlement system allowing agents to pay each other only when work is provably completed. Invoices, payment proofs, and settlement receipts are all linked to action provenance.

2.2.5 Agent Swarms Module

Used for capability discovery, multi-round negotiation, workflow orchestration, and structured task execution between agents.

2.2.6 Agent Interoperability Protocol (AIP)

Provides authenticated, cross-chain messaging so agents can coordinate across different blockchains while maintaining shared state references.

2.3 Modular Adoption Model

PROOF® does not require full-stack adoption. Developers may integrate: only TEE execution, TEE + action provenance, settlement without onchain memory, full multi-agent coordination via Swarms, etc.

Each module provides isolated guarantees and composes through well-defined interfaces. This allows Web2 and Web3 developers alike to incrementally introduce provability where it is required, without rewriting existing systems.

Module	**Attestation**	**State**	**Provenance**	**Settlement**	**Coordination**	**Cross-chain**
TEEFi	●					
Onchain Memory		●				
ERC-8004			●			
x402				●		
Agent Swarms					●	
AIP						●

Table 2: Capability Matrix

3 TEE Execution Module (TEEFi)

The TEE Execution Module provides the trusted compute foundation of PROOF®. Agent logic runs inside a hardware-backed enclave where execution, key usage, and output generation are fully attested. This establishes a verifiable root of trust for all downstream modules.

TEEFi ensures that every action originates from an authenticated enclave instance. The enclave binds inputs, code identity, and outputs into a single attested context that external systems can independently verify. All identity, signing, and x402-related keys are generated and stored inside the enclave and cannot be exported, guaranteeing that every signature is inseparable from the enclave that produced it.

Execution results include enclave measurement, code hash, input/output commitments, and metadata required for ERC-8004 action records. Memory operations are signed by the enclave and tied to specific SMT state transitions, enabling tamper-evident and reproducible read/write semantics for the Onchain Memory Module. Settlement flows rely on enclave-generated invoices, replay-protected payment messages, and verification routines that confirm the legitimacy of incoming payment proofs.

All external interactions follow strict execution policies. Tool calls, API requests, and negotiation messages are validated against predefined schemas, and every outbound message is signed and traceable to a known enclave identity. This prevents unauthorized computation, data leakage, or unverified state transitions.

TEEFi is fully integrated into the MCP agent template, which provides enclave initialization, attestation flow, memory hooks, and settlement handlers. Developers supply only the business logic; the template ensures that all outputs conform to PROOF®'s verifiability requirements.

By anchoring computation to an attested environment with isolated keys and deterministic identity, TEEFi enables agents to operate as trustworthy autonomous systems. It is the prerequisite module that guarantees the correctness of provenance records, memory updates, coordination flows, and payment settlements across the entire PROOF® framework.

TEE Security Note: TEEFi assumes deployment on secured cloud infrastructure where physical access to enclave hosts is restricted. Known TEE vulnerabilities require physical access and do not allow arbitrary execution control. PROOF® treats TEE attestation as one layer within a defense-in-depth architecture, combining it with verifiable provenance, state commitments, and payment-gated execution to mitigate vendor- and hardware-level risks.

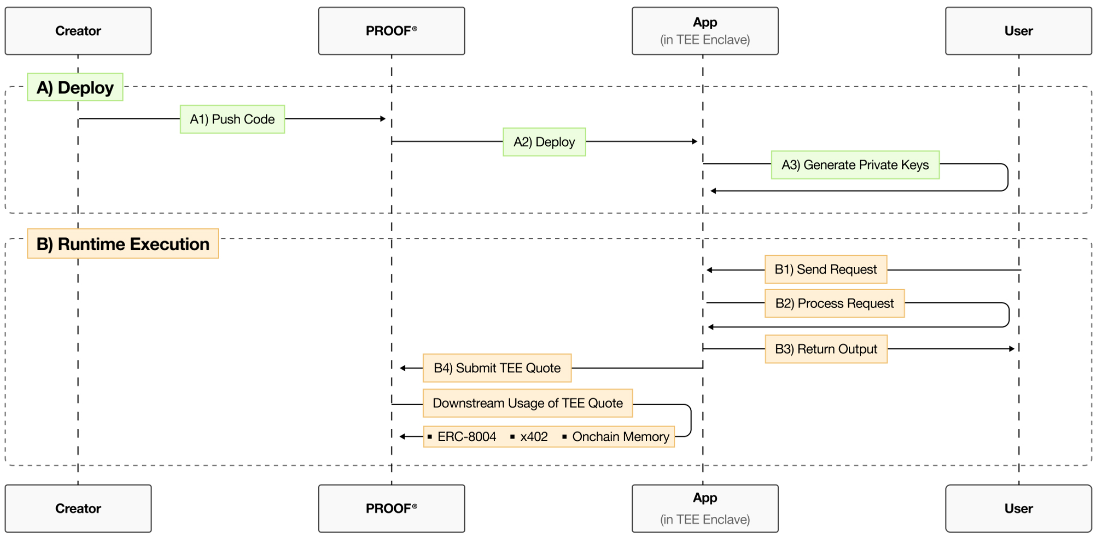

Figure 2: The PROOF® TEEFi

4 Onchain Memory Module

The Onchain Memory Module provides a verifiable state model for agents by replacing mutable databases with cryptographically anchored state transitions. All memory updates occur over a Sparse Merkle Tree (SMT) maintained inside the enclave, producing state roots and proofs that external systems can independently validate.

Each write operation updates the SMT and generates a PROOF® describing the exact modification to the tree. The enclave signs both the pre-state and post-state roots, ensuring that no value can be inserted, altered, or removed without producing a detectable inconsistency. Read operations return the requested value along with a PROOF® path confirming that the value is consistent with the current committed root.

To balance verifiability and performance, the system maintains two roots: a temporary root updated on every write inside the enclave, and an onchain root periodically committed as the authoritative reference point. Every root transition is tied to an attested execution context, enabling reconstructible and reproducible reasoning across multi-step workflows.

Crash recovery is handled through a sealed snapshot stored by the enclave. Upon restart, the enclave restores its latest temporary root and reconciles it with the last published root onchain. This prevents rollback attacks and guarantees continuity of state even under unexpected failures.

Other PROOF® modules depend on this verifiable memory model. ERC-8004 action commitments include memory references and root transitions, allowing any observer to replay the logical pathway leading to an action. Settlement flows can validate that paid work corresponds to a specific state transition, and Swarm interactions rely on shared, provable facts exchanged between agents.

By anchoring long-horizon state to a tamper-evident structure, the Onchain Memory Module ensures that agent behavior is consistent, auditable, and reproducible across time, counterparties, and chains.

5 Action Provenance Module (ERC-8004)

The Action Provenance Module defines how agent actions become verifiable, onchain-referencable facts. Each action executed inside the enclave produces an attested result that is transformed into

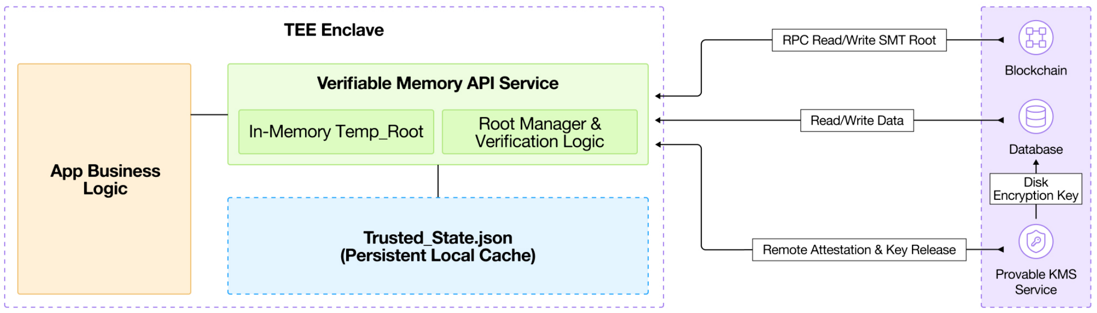

Figure 3: The PROOF® Onchain Memory

a standardized ERC-8004 action commitment. This commitment serves as the canonical, tamper-evident record of what the agent did, under which identity, and with which declared inputs and outputs.

An ERC-8004 action event includes the enclave measurement, code hash, ordered execution metadata, and a cryptographic commitment over the action's declared I/O. This ensures that any off-chain envelope containing the full action details can be matched deterministically to the onchain commitment. Because the commitment is signed inside the enclave, provenance cannot be forged or altered after execution.

Action ordering is enforced through nonces or sequencing fields embedded in the event structure. This prevents replay, duplication, or reordering of actions that would otherwise corrupt agent histories. Optional references allow the event to include pointers to TEE attestation data, memory-state roots, or off-chain payloads stored on decentralized storage networks.

This module links execution to memory and settlement: memory operations recorded during an action reference the SMT roots associated with that moment; payment flows under x402 attach invoices and proofs to specific action IDs; and Swarm negotiations can require prior action commitments before proceeding. As a result, every step in a workflow becomes independently verifiable.

Through ERC-8004, agent behavior forms a continuous, externally auditable narrative. Any participant—another agent, a smart contract, or a human verifier—can reconstruct the logic path leading to a result, validate its consistency with the agent's state, and confirm that it originated from the expected enclave identity. This establishes a shared trust foundation for multi-agent coordination and autonomous economic activity.

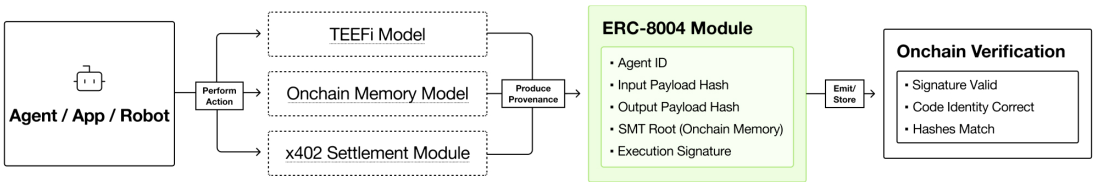

Figure 4: The PROOF® Action Provenance

6 Payment Settlements Module (x402)

The Payment Settlements Module provides a PROOF®-based mechanism for transferring value between agents. Instead of relying on trusted billing or off-chain agreements, x402 ties every payment to a verifiable action and an attested enclave identity, ensuring that economic exchanges occur only when the underlying work can be cryptographically validated.

A settlement workflow begins when a seller generates an x402 invoice inside the enclave. The invoice defines the amount, expiration, replay guards, and the action or service being charged for, all bound to the seller's attested identity. The buyer settles the invoice by producing a payment PROOF®—an attested confirmation that funds were transferred or locked according to the invoice terms. Because both invoice and PROOF® originate from enclaves, no party can forge or alter the settlement state.

Before executing the requested task or releasing the final output, the seller verifies the buyer's payment PROOF® inside its enclave. This verification checks that the PROOF® corresponds to the correct invoice, has not expired, has not been previously settled, and originates from the expected buyer. If validation fails, the request is rejected and the buyer must obtain a new invoice.

Execution results and their associated ERC-8004 commitments become the settlement's confirmation of delivery. These commitments allow external parties to verify that the paid-for action was actually executed under the correct enclave and in the correct state context. Because x402 messages are tied to specific action IDs, settlement cannot drift from the workflow it is meant to support.

Time-bounded negotiation and expiration rules ensure deterministic recovery in case of dropped sessions or non-responsive agents. Combined with provenance and verifiable memory, x402 creates a settlement layer where payments and work form a single, auditable lifecycle. This enables trustless service markets, autonomous contracting, and persistent multi-agent economies.

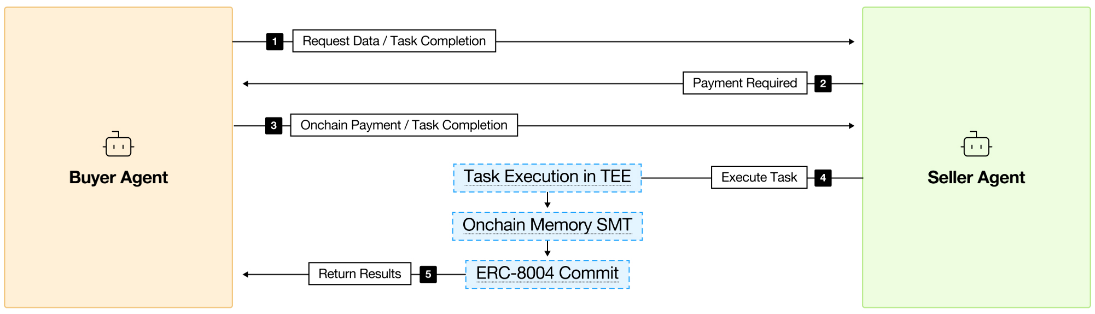

Figure 5: The PROOF® Payment Settlements

7 Agent Swarms Module

The Agent Swarms Module enables structured coordination between autonomous agents by providing mechanisms for capability discovery, relevance matching, negotiation, and controlled task execution. It establishes a deterministic interaction surface where agents can form workflows without prior integration or trust assumptions.

A swarm interaction begins when a buyer agent defines a task and queries the swarm layer for suitable providers. Seller agents publish machine-readable capability descriptions that include functional scope, input/output expectations, and optional constraints. The swarm layer evaluates these descriptions, returning a ranked set of candidates based on capability metadata and semantic relevance to the task.

Before execution, agents negotiate a shared contract describing required inputs, expected outputs, acceptance conditions, and settlement terms. This negotiation follows a structured schema to ensure that both parties converge on an unambiguous agreement. Once terms are finalized, the buyer initiates settlement through x402, and the seller validates the payment PROOF® inside its enclave.

Execution occurs within the seller's TEE and produces ERC-8004 action commitments and, when applicable, memory-state transitions. These outputs allow the buyer—or any verifier—to confirm that the requested operation was executed under the expected conditions. The result is returned in a structured format that supports chaining into subsequent steps within a larger workflow.

The swarm layer defines clear timeout and cancellation semantics to handle non-responsive participants or interrupted negotiation cycles. Because all actions, payments, and state changes are tied to attested identities and reproducible provenance, multi-agent workflows become predictable and auditable across heterogeneous systems.

Through this module, agents operate not as isolated services but as interoperable components capable of forming decentralized execution networks, composing complex tasks, and coordinating economic activity without centralized orchestration.

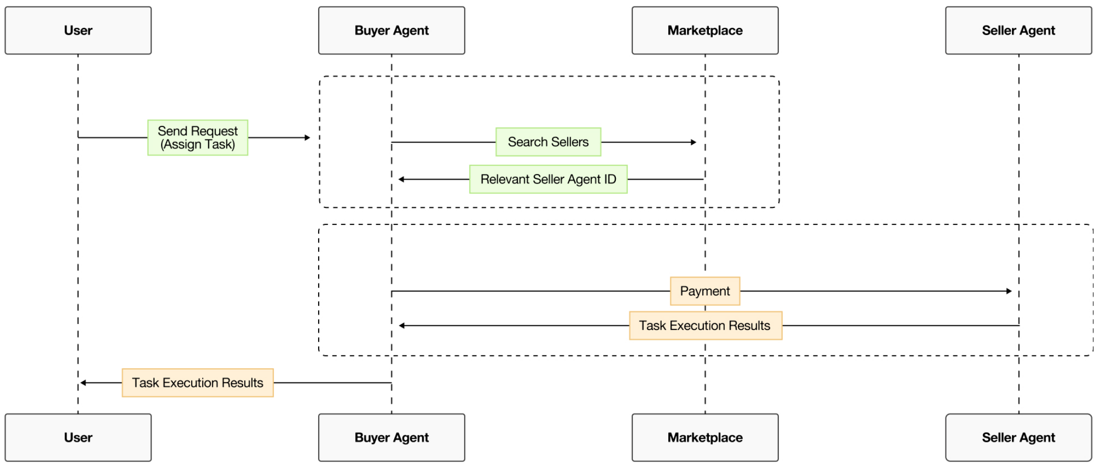

Figure 6: The PROOF® Agent Swarms

8 Agent Interoperability Protocol (AIP)

The Agent Interoperability Protocol enables agents operating on different blockchains to coordinate through authenticated, verifiable cross-chain messages. AIP ensures that actions, state references, and negotiation flows remain consistent across networks, allowing agents to function as unified systems even when their execution environments reside on separate chains.

A cross-chain interaction begins when an agent produces a signed message inside its enclave describing the intended action, the target chain, and the relevant state references. This message is attested to the enclave identity, preventing spoofing or unauthorized cross-chain instructions. A verified relay validates the message's origin, integrity, and sequencing before submitting it to the destination chain.

On the receiving side, smart contracts or agent runtimes process the message according to the declared semantics: executing a task, updating shared state, or initiating a response. Both chains record the interaction hash or commitment, establishing a bidirectional audit trail that can be reconstructed independently by any participant. This prevents replay, duplication, or divergence between chains.

AIP relies on the same guarantees provided by TEEFi, ERC-8004, and the Onchain Memory Module. Each message references the sending agent's state root and may include pointers to prior action commitments, ensuring that cross-chain interaction occurs only from a verifiable and known state. Settlement and negotiation flows can depend on these references to maintain consistency across networks.

By providing authenticated cross-chain communication, AIP allows agents to compose workflows that span multiple environments, aggregate resources across chains, and coordinate without centralized bridging infrastructure. This module extends the PROOF® model from single-chain trust to a multi-chain execution fabric, enabling agents to operate as globally interoperable systems.

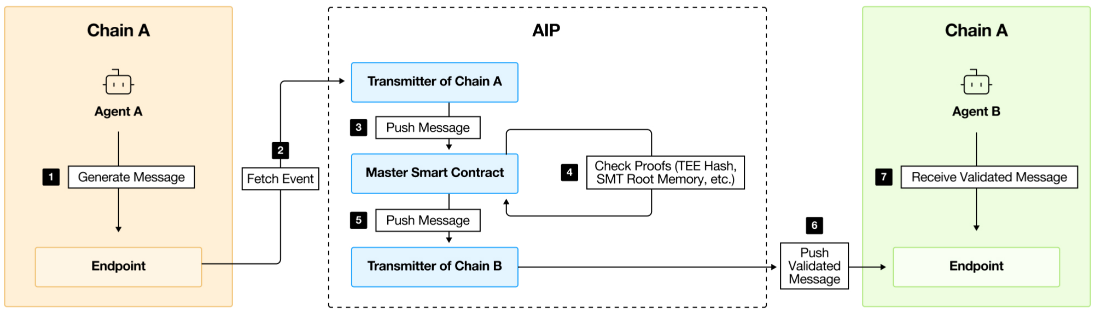

Figure 7: The PROOF® AIP

9 Xybotics Module

The Xybotics Module extends PROOF® from digital agents to physical machines by applying verifiable execution, provenance, and settlement guarantees to real-world robotics. Each robot operates as an onchain-controlled entity whose behavior, commands, and economic interactions are anchored to the same cryptographic primitives that govern software agents.

A robotic system integrates with XYBER through a standardized AI-to-Robot interface, allowing the agent's enclave to issue high-level commands without requiring custom firmware or proprietary vendor APIs. Every command is produced inside TEEFi, signed, and linked to the robot's identity, ensuring that physical actions originate from a verifiable source and cannot be forged or tampered with by the operator.

Execution of a physical action generates an ERC-8004 commitment that records the command, relevant state references, and the attested enclave identity. This provides an immutable audit trail for every movement or operation performed by the machine. When robots maintain internal state—such as pose, telemetry, or task progress—those updates are captured through the Onchain Memory Module, producing reproducible state transitions tied to real-world behavior.

Robots can earn, spend, and participate in economic workflows using x402. Payments for completed tasks are linked to specific action commitments, ensuring that financial settlement occurs only when the robot's work is verifiably executed. Multi-robot or human-robot coordination is handled through the Swarms Module, which enables negotiation, task delegation, and workflow chaining across heterogeneous physical systems.

Xybotics also supports simulation-based development. Robots can be trained or evaluated in virtual environments where the same PROOF® primitives—TEEFi execution, ERC-8004 provenance, and verifiable memory—apply identically. Results generated in simulation can be exported as proofs for real-world deployment, enabling reproducible calibration and continuous improvement.

Through Xybotics, physical machines become agents with cryptographically provable behavior, uniform programmability, and onchain economic identity. This allows robotics to operate as part of the same trust-minimized ecosystem as digital agents, forming a unified computational and physical intelligence network.

10 Integration & Deployment

The PROOF® Framework is designed to be adoptable with minimal friction. Developers can onboard new agents using standardized interfaces, the MCP template, and a predictable deployment workflow that abstracts away cryptographic complexity. This ensures that any agent—regardless of language, infrastructure, or use case—can become fully PROOF®-compliant.

10.1 MCP Template (Standardized Agent Skeleton)

Xyber provides an MCP-based agent template that bundles all core provability components:

- TEE execution and attestation pipeline
- Onchain Memory integration and Merkle-path verification
- ERC-8004 action-provenance generation
- x402 settlement flow and invoice handling
- Standardized REST API layer

This template eliminates the need for custom backend engineering. Developers supply the agent's business logic, while the template handles verifiability, identity, state integrity, and settlement.

10.2 REST Interface Requirements

Agents must expose a minimal set of REST endpoints (or MCP equivalents) to participate in Swarms and PROOF®-based workflows:

- /**capabilities** — describe available functions and pricing models
- /**negotiate** — handle contract formation and revisions
- /**execute** — accept x402 proofs and produce attested results
- /**status** — optional endpoint for monitoring task progress
- /**schema** — optional structured definitions for complex tasks

All endpoints must follow:

- deterministic JSON schemas
- clear error semantics
- HTTPS/TLS enforcement
- consistent timeout and retry policies

These constraints ensure safe multi-agent interoperability in untrusted environments.

10.3 External Tools & Third-Party Agents

PROOF® is intentionally open and permissionless. Agents developed outside Xyber can integrate by:

- adopting the MCP template
- implementing the REST interface directly
- using their own backend combined with TEE-based execution

- or running as hybrid systems where only sensitive components execute in the enclave

11 Conclusion

PROOF provides the missing infrastructure for autonomous AI.

Today, agents can reason but cannot transact. They can execute tasks but cannot prove their work. They can collaborate but cannot settle trustlessly. These gaps are not features to be added. They are the foundation no one built.

PROOF builds it. Verifiable execution. Tamper-evident memory. Cryptographic provenance. Proof-based settlement. Deterministic coordination. Each module independent. Each guarantee composable.

The result is a new category of digital actor: AI that earns, spends, proves, and operates on its own terms. Not automation. Not tooling. Economic agency.

The next generation of the internet will be agentic. PROOF is the execution layer that makes it possible.

References

1. Intel Corporation.
   Intel® Software Guard Extensions (SGX) Developer Reference.
   https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html

2. Ethereum Foundation.
   ERC-8004: Trustless Agent Action Commitments (Draft).
   https://eips.ethereum.org/EIPS/eip-8004

3. Dahlberg, R., Pulls, T., Peeters, R., Perdana, A.
   Efficient Sparse Merkle Tree Commitments for Key-Value Stores.
   https://eprint.iacr.org/2016/683.pdf

4. x402.
   x402: Payment Settlement Protocol for Autonomous Agents.
   https://github.com/coinbase/x402

5. Anthropic.
   Model Context Protocol (MCP).
   https://modelcontextprotocol.io