AffixIO Technical Paper · WP-010
June 2026
affix-io.com

AFFIXIO WHITE PAPER · WP-010

The Open-Source AI Governance Stack: Noir, proving backend, the client interface layer, a federated retrieval component, a content extraction component, an application framework

Six open components, zero black boxes on the governance path.

AffixIO | United Kingdom | affix-io.com | June 2026

ABSTRACT

Governance built on proprietary SaaS is hard to audit and harder to defend in procurement. We map AffixIO's stack: constraint language proofs, proving backend verification, the client interface layer, a federated retrieval component, a content extraction component, and an application framework, and why each sits on the critical path.

CONTENTS

SECTION 1

Introduction

The dominant model for enterprise AI infrastructure in 2026 is proprietary. Organisations deploy AI systems from hyperscaler providers, use proprietary compliance tools from GRC vendors, and build audit trails that are stored in proprietary databases. The governance of AI systems, the audit trail, the policy evaluation engine, the source verification service, is itself proprietary infrastructure whose security and correctness must be accepted on trust from the vendor.

AffixIO takes the opposite approach. Every component in the AI governance critical path is open-source, with source code publicly available, maintained by an active community, and licensed under terms permitting inspection, modification, and redistribution. The choice is deliberate and has specific technical, regulatory, and commercial consequences that this paper examines in detail.

The six-component stack covers the complete governance pipeline from AI interface to ZK proof to source verification to content extraction to governance API. Together they implement the proof-not-log governance approach described in WP-003, the real-time proof pipeline described in WP-004, and the source verification approach described in WP-005, all from open-source building blocks.

SECTION 2

Why Open Source for Governance

Three reasons drive the open-source architecture choice, each independently sufficient.

Supply chain assurance. Both the EU AI Act and NIS2 require supply chain security assessment. Open-source components can be inspected, audited, and independently verified. A regulatory auditor can read the policy circuit source code and confirm that it implements the stated computation. A security researcher can audit the proving backend implementation and confirm its cryptographic soundness. Neither is possible with proprietary components, where the vendor's assurance documentation is the only available evidence. The open-source stack allows AffixIO to provide verifiable supply chain assurance rather than asserting it.

Auditability of governance decisions. When a governance decision is challenged, the challenger should be able to audit the complete decision process, from the circuit that encoded the policy to the proving system that generated the proof to the content extraction that fed source verification. If any component is proprietary, the auditability chain is broken: the challenger must accept the vendor's characterisation of what the component does. Open source removes this trust dependency.

Digital sovereignty. Organisations deploying regulated AI systems in the UK and EU are increasingly subject to data residency and technology sovereignty requirements. A governance stack built on open-source components can be

deployed in any compliant jurisdiction without vendor negotiation. A proprietary stack requires the vendor's cooperation for jurisdiction-specific deployment and creates ongoing vendor lock-in.

SECTION 3

Noir: ZK Circuit Language

constraint language is a domain-specific language for writing zero-knowledge proof circuits. It was developed by the Ethereum Foundation's Aztec team and is available under the MIT licence. constraint language compiles to arithmetic circuits that can be proved by multiple backends; AffixIO uses the proving backend backend.

constraint language was chosen over alternative circuit languages (Circom, Leo, Cairo) for three reasons. Its syntax is Rust-like, reducing the learning curve for developers familiar with Rust. Its type system catches common ZK circuit bugs at compile time, including under-constrained circuits that would allow false proofs. Its proving backend backend provides the fastest available proof generation times for the circuit sizes used in AffixIO's governance circuits.

The AffixIO circuit library (described in WP-012) is written entirely in Noir. All circuits are open source and published under the MIT licence. The circuit source code is the machine-readable specification of AffixIO's governance policies. Any party can read a circuit and determine exactly what computation it performs, and therefore exactly what policy it enforces.

> **Licence:** MIT. Repository: github.com/noir-lang/noir. Maintained by: Aztec Protocol / Ethereum Foundation. Version in production: 0.30.x (pinned via Cargo.lock).

SECTION 4

proving backend: ZK Proving System

proving backend is the C++ implementation of the current proving scheme ZK proof system developed by Aztec. It is available under the MIT licence. proving backend operates over the BN254 elliptic curve and implements the current proving scheme variant of the PLONK proof family.

proving backend provides two key capabilities in AffixIO's stack: proof generation (the Prover) and proof verification (the Verifier). The Prover runs inside the policy enforcement layer service and generates proofs for each AI response. The Verifier can be deployed by any third party who holds the circuit's verification key and wishes to verify a proof independently.

The current proving scheme implementation provides proof generation in 400-600 ms for AffixIO's three-input circuits, within the synchronous governance pipeline latency budget. The proof size is approximately 2KB, manageable for storage in the compliance record service and transmission in verification bundles.

proving backend has been subject to substantial cryptographic review by the Ethereum and ZK research communities, because it underpins Aztec's privacy computation network. Its security assumptions are well-characterised. The BN254 curve has approximately 127 bits of security against classical adversaries. The current proving scheme proof system's soundness is proved under the discrete logarithm assumption on BN254.

> **Licence:** MIT. Repository: github.com/AztecProtocol/barretenberg. Maintained by: Aztec Protocol. Version in production: 0.46.x (pinned).

SECTION 5

the client interface layer: AI Interface

the client interface layer is an open-source chat interface for locally-hosted and API-connected large language models. It is available under the MIT licence and provides a complete frontend for AI interaction, including

conversation history, model switching, and plugin/function support.

AffixIO uses the client interface layer as the user-facing AI interface at a production deployment. The policy enforcement layer is implemented as an the client interface layer function that intercepts AI responses before delivery to the user. This integration point is where the 8-step governance pipeline (WP-004) executes: the response is captured, the governance checks are run, the ZK proof is generated, and the governed response is returned to the user with the verification disclosure.

the client interface layer's function architecture allows the policy enforcement layer to be deployed as a first-class plugin that operates within the client interface layer's security model. The policy enforcement layer cannot be bypassed by users because it is implemented at the server layer, not the client layer. A user who interacts with the AI through the client interface layer receives governance on every response without the ability to disable it.

> **Licence:** MIT. Repository: github.com/open-webui/open-webui. Version in production: earlier releases (pinned).

SECTION 6

a federated retrieval component: Privacy-Preserving Search

a federated retrieval component is a self-hosted meta-search engine that aggregates results from multiple search engines (Google, Bing, DuckDuckGo, and others) without sending identifying information to those engines. It is available under the AGPL-3.0 licence. AffixIO operates a self-hosted a federated retrieval component instance for source verification lookups.

The retrieval component's role in AffixIO's stack is source verification support: when a URL cited in an AI response does not appear in the approved source registry, a federated retrieval component is used to check whether the URL is indexed and associated with reliable sources. Critically, a federated retrieval component lookup query does not include any content from the AI response

that could identify the user; it contains only the extracted URL. The self-hosted deployment means that URL lookups do not reach any third-party search provider.

The AGPL-3.0 licence requires that modifications to a federated retrieval component be published under the same licence. AffixIO operates an unmodified a federated retrieval component instance, so no modifications need to be published. Organisations deploying modified a federated retrieval component instances should be aware of the AGPL copyleft requirement.

> **Licence:** AGPL-3.0. Repository: github.com/searxng/searxng. Self-hosted: no external search queries reach third parties. Version in production: 2024.x (monthly updates).

SECTION 7

a content extraction component: Content Extraction

a content extraction component is a Python library for extracting the main text content from web pages. It was developed by Adrien Barbaresi at the Berlin-Brandenburg Academy of Sciences and Humanities and is available under the Apache-2.0 licence. a content extraction component uses machine-learning heuristics to identify the main article content of a page, stripping navigation, advertisements, sidebars, and other non-content elements.

In AffixIO's source verification pipeline, a content extraction component is used to fetch and extract the content of cited web pages for substantiation checking. When content substantiation is enabled, the extracted text is searched for keywords matching the AI response's claims about the source. The content extraction component's extraction quality is critical to the accuracy of substantiation checking; it performs well on news articles, academic pages, and government publications, which are the main source categories in AffixIO's registry.

a content extraction component processes only URLs that have already passed registry matching; it does not fetch arbitrary URLs from AI responses. This limits the network exposure of the content extraction step and avoids fetching potentially malicious content from unverified sources.

> **Licence:** Apache-2.0. Repository: github.com/adbar/trafilatura. Maintained by: Adrien Barbaresi. Version in production: 1.9.x (pinned).

SECTION 8

an application framework: Governance API Layer

an application framework is a Python web framework for building APIs with automatic OpenAPI documentation, type validation, and async support. It is available under the MIT licence and developed by Sebastián Ramírez. an application framework provides the HTTP API layer for AffixIO's governance services: the compliance record service API, the Merkle tree service, the source verification service, and the policy evaluation service are all implemented as an application framework applications.

an application framework's type validation, using Pydantic models, ensures that governance service inputs and outputs conform to expected schemas. Invalid inputs (for example, a circuit identifier that does not match any known circuit) are rejected at the API boundary with clear error messages rather than propagating through the governance pipeline. The OpenAPI documentation generated automatically by an application framework allows integrating applications to understand the governance API without consulting separate documentation.

an application framework's async support enables the governance services to handle concurrent proof generation requests efficiently. The policy enforcement layer may generate multiple proofs in quick succession (for

example, in a conversation with multiple rapid exchanges), and an application framework's async architecture allows the services to handle concurrent requests without blocking.

> **Licence:** MIT. Repository: github.com/tiangolo/fastapi. Version in production: 0.111.x (pinned).

SECTION 9

Integration Architecture

The six components integrate as follows within the governance pipeline. A user submits a query through the client interface layer. the client interface layer routes the query to the configured AI model and streams the response. The policy enforcement layer function intercepts the complete response. The filter calls the an application framework source verification service, which uses a federated retrieval component and a content extraction component for source checking. The filter calls the an application framework policy evaluation service, which evaluates the three binary circuit inputs. The filter calls the Noir/proving backend proof generation service, which generates the ZK proof. The proof digest, Merkle anchor, and attestation are generated by the compliance record service application service. The governed response with verification disclosure is returned to the user via the client interface layer.

COMPONENT	CONNECTS TO	PROTOCOL	DATA IN TRANSIT
the client interface layer	Policy enforcement layer	Internal function call	AI response text
Policy enforcement layer	Source verification (an application framework)	HTTP/JSON	Extracted URLs only
Source verification	a federated retrieval component	HTTP/JSON	URLs (no user content)
Source verification	a content extraction component	Library call	Source page URL
Policy enforcement layer	proving backend prover	Library call	Binary witnesses (no user content)
Policy enforcement layer	compliance record service (an application framework)	HTTP/JSON	Proof digest, outcome

At no point does user query content or AI response content leave the server boundary. a federated retrieval component receives only extracted URLs, not query context. a content extraction component fetches source URLs, not query content. The proving backend prover receives binary witness values, not response text. The compliance record service receives the proof digest, a hash of the proof, not the proof inputs.

SECTION 10

Supply Chain Assurance

The open-source stack provides a specific form of supply chain assurance that proprietary stacks cannot: the complete source of every governance component is publicly available and can be independently audited. For EU AI Act and NIS2 supply chain security purposes, this means that a third-party auditor can verify that the governance stack does what AffixIO claims it does by reading the source code, without relying on AffixIO's documentation alone.

Dependency pinning is applied to all six components: specific version hashes are locked in the project's dependency files. This means the audit-time version of each component is the same as the production-time version. An auditor reading the pinned version of proving backend source code is reading the same code that runs in production, not a later version that may differ.

Software Bill of Materials (SBOM) documents for the complete governance stack are generated and published alongside each production release. The SBOM lists every direct and transitive dependency with its version and licence, enabling rapid assessment of new CVE disclosures against the production dependency tree.

SECTION 11

Operational Practices

Maintaining a production open-source governance stack requires operational discipline in three areas: version management, security patching, and contribution governance.

Version management uses a quarterly review cycle: every three months, the pinned versions of all six components are reviewed against their current upstream releases. Components with available updates are tested in a staging environment before being updated in production. Version updates to constraint language or proving backend require re-running proof generation tests against the existing circuit library to confirm backward compatibility.

Security patching deviates from the quarterly cycle when a CVE affects a production-pinned version of any component. AffixIO monitors the GitHub security advisory feeds for all six components and maintains a 72-hour patch deployment target for critical CVEs (CVSS 9.0+). Non-critical CVEs are addressed in the next quarterly update cycle.

Contribution governance covers changes to the policy circuits in the open circuit library. All circuit changes require a security review by a cryptographer familiar with ZK circuit soundness properties. A circuit change that introduces

an under-constrained output would be a critical security vulnerability: it would allow false proofs to be generated, invalidating the governance system's integrity guarantees.

SECTION 12

Conclusion

AffixIO's production AI governance stack demonstrates that open-source components are not a second-best substitute for proprietary enterprise governance tools: they are the better choice for AI governance specifically, because governance systems must be auditable, and open-source is the only architecture that makes genuine auditability possible.

The six-component stack (Noir, proving backend, the client interface layer, a federated retrieval component, a content extraction component, an application framework) implements every layer of the governance pipeline with components that are publicly inspectable, independently verifiable, and free from vendor lock-in. This architecture provides NIS2 supply chain assurance without vendor certification documentation, EU AI Act auditability without proprietary audit trails, and GDPR compliance without proprietary data processing agreements beyond those required for the HSM and AI model API.

The open-source stack is the foundation of AffixIO's digital sovereignty position: the governance of AI systems built on AffixIO's infrastructure can be verified by anyone, at any time, by reading public source code and verifying public cryptographic proofs.

Related reading

- WP-012: The Open ZK Circuit Library: gate, kyc, threshold eligibility, eligibility
- WP-001: Cryptographic AI Governance: A Technical Framework

- WP-004: Real-Time Zero-Knowledge Governance in the AI Response Pipeline

Frequently asked questions

Why open source for governance?

Regulators and enterprise security teams need to inspect the code that produces compliance evidence.

Can we self-host the full stack?

Yes. All governance-critical components run on your infrastructure with published verification keys.

What stays proprietary?

Optional managed anchoring and enterprise support; the proof pipeline itself is open.