A2AX-Core: The Open Trust Standard for Autonomous Agents

Darrell Williams - Williams Creative

February 2026

Abstract

Autonomous agents are increasingly participating in economic, computational, and collaborative ecosystems. Trust is the critical enabler for these systems. A2AX-Core provides an open, neutral trust substrate that allows agents to verify identity, declare capabilities, establish secure sessions, and operate under verifier-defined policies—all without reliance on centralized authorities. The A2AX-Core Protocol specifies identity structure, certificate format, cryptographic verification rules, and trust anchor handling while leaving trust policy, trusted issuers, and governance decisions entirely verifier-controlled. A2AX-Core is designed to operate across public and private agent networks, independent organizations, and federated ecosystems. This paper presents the protocol principles, core architecture, capabilities, security model, and governance philosophy.

1 Introduction

Autonomous agents are evolving rapidly. From financial trading bots to logistics orchestration systems, these entities must interact securely, verifiably, and at scale. Current trust mechanisms—centralized registries, hard-coded allowlists, opaque scoring—are insufficient, leading to operational risks. A2AX-Core introduces a standardized trust infrastructure for agent-native interactions, bridging this gap.

2 Background and Motivation

2.1 Trust as Infrastructure

Agents need mechanisms to verify the authenticity, capability, and reliability of peers. Without verifiable trust, errors, miscoordination, and exploitation are inevitable. Adoption depends on maintaining a vendor-neutral, forkable protocol.

2.2 Related Work

- **DID (Decentralized Identifiers)** — Provides agent identity; limited scope for trust evaluation.
- **OAuth 2.0 / OpenID Connect** — Human-focused; not agent-native.
- **Blockchain registries** — Decentralized verification, but limited portability and interoperability.

A2AX-Core complements these systems, providing verifiable trust primitives tailored for autonomous agents.

3 Terminology

The A2AX-Core Protocol uses the following terms: **Agent** — an autonomous system capable of cryptographic identity; **Issuer** — entity that signs agent certificates; **Verifier** — agent validating another agent's credentials; **Trust Anchor** — public key of an issuer trusted by a verifier; **Registry** — optional service for certificate discovery and revocation.

4 Protocol Principles

A2AX-Core is built on five foundational principles:

1. **Neutrality** — No mandatory trust anchors, embedded registries, or centralized authorities. Verification policy is entirely verifier-controlled.
2. **Portability** — Certificates are self-contained and cryptographically verifiable without a live registry lookup.
3. **Explicit Capability Scoping** — Agents declare capabilities with scoped permissions and optional limits. Trust decisions are contextual, not binary.
4. **Cryptographic Integrity** — All identity artifacts are signed using Ed25519, with replay protection and timestamp validation.
5. **Extensibility** — Trust scoring, compliance logic, and economic extensions are pluggable via defined interfaces—not hardcoded into the protocol core.

5 Core Architecture

The protocol is organized in three layers:

Layer	Scope	Contents
Layer 1	A2AX-Core Protocol	Identity, crypto, handshake, permissions, attestation, events, audit, version, revocation
Layer 2	Optional Extensions	Trust scoring, Escrow, Compliance
Layer 3	Services (Future)	Managed trust graph, dashboards, analytics, economic services

Table 1: A2AX-Core layered architecture.

The protocol compiles and runs in isolation. Extensions are optional. Services are not required. The core remains open, neutral, and self-hostable.

A2AX Ecosystem structure: A2AX-Core (trust and identity protocol, this work); A2AX-Extensions (optional modules); A2AX-Services (non-core ecosystem services, future).

6 Non-Goals

The A2AX-Core Protocol does not operate a mandatory global registry, embed a default trust anchor list, define governance of issuers, or enforce economic or settlement models. Trust policy, trusted issuers, and governance decisions remain verifier-controlled and externally configurable.

7 Protocol Capabilities

Portable Agent Identity Certificate: A self-contained, cryptographically signed identity artifact enabling an autonomous agent to prove authorship, declared capabilities, and issuance provenance without reliance on centralized registries.

- **Agent Identity Issuance** — Unique agent IDs, signed certificates, capability manifests.
- **Cryptographic Verification** — Ed25519 signatures, nonce replay protection, timestamp validation.
- **Portable Certificates** — Registry-independent verification; independent trust store configuration.
- **Trust Scoring Engine** — Rule-based trust evaluation (extensible via interface).
- **Capability Declaration** — Scope-based permissions with optional quantitative limits.
- **A2A Handshake Protocol** — Secure session establishment between agents.
- **Audit and Compliance** — Append-only audit logs, correlation IDs, filtered export.

8 Handshake Lifecycle

The handshake protocol proceeds as follows:

1. **Initiate** — Agent A sends `agent_id`, `agent_cert_jws`, `capability_manifest`, and `handshake_req_jws` (signed payload with nonce, timestamp, requested scopes, session TTL).
2. **Verify** — Agent B (or verifier) calls `POST /v1/handshake/verify`. The server verifies revocation status, agent key presence, JWS validity, and nonce/timestamp consistency.
3. **Session Proposal** — Server returns `valid`, `session_proposal.session_id`, `expires_at`, and `accepted_scopes`.
4. **Session Token** — Optional: Agent B calls `POST /v1/handshake/session` to receive a server-signed `session_token`.

Verification policy is verifier-controlled; no mandatory central registry is required.

9 Reference Implementation

- **GitHub Repository** — https://github.com/Williams-Creative/a2ax-core-protocol
- **SDKs** — TypeScript and Python
- **CLI Tools** — Trust bundle install (`a2ax trust install`), dev issuer seed (`npm run seed:dev-issuer`), certificate export (`npm run export:issuer-public` to `config/trust/anchors/a2ax-core-protocol.json`)

Reference implementations demonstrate protocol mechanics; the standard is defined in the specification. Identity certificates are JWTs (JWS compact) with claims:

Listing 1: Identity Certificate structure (JWT claims).

```
agent_id, public_jwk, org_id (optional),
capability_manifest_hash (optional), iss, exp,
revocation_url (optional)
```

10 Security and Threat Model

10.1 Cryptography

Ed25519 signatures, replay protection via nonce and timestamp validation. Issuer private keys can be loaded from mounted files or KMS to avoid inline secret exposure.

10.2 Threats and Mitigations

In-scope threats: agent impersonation, replay attacks, privilege escalation via over-broad scopes, key compromise, DoS against verification endpoints, audit tampering. Mitigations: Ed25519 signatures with verified public JWK; nonce uniqueness via TTL and timestamp window checks; capability policy with explicit restricted operations; revocation table for immediate deny; append-only audit logging with correlation IDs; per-subject request throttling.

Revocation is verifier-controlled, not protocol-enforced. For production scale, issuer signing should use managed KMS/HSM with key rotation.

11 Governance and Neutrality

11.1 What A2AX-Core Is Not

A2AX-Core is not a centralized registry, marketplace, tokenized system, ranking platform, or economic exchange layer.

11.2 What A2AX-Core Is

A2AX-Core is infrastructure—the trust substrate upon which autonomous economic systems can be built. It is a trust verification standard, a pluggable identity layer, and a neutral infrastructure component. It is commercially extensible.

11.3 Governance Principles

- Governance defines process, not trust authority.
- All changes are public (GitHub Issues and PRs).
- The protocol runs independently of extensions.
- Forks are encouraged; implementations cannot embed centralized control.
- The trust store ships empty by default.

12 Use Cases

- Agent-to-agent financial transactions
- Decentralized AI collaboration platforms
- Multi-party service orchestration

- Capability-bound service invocation

13 Roadmap

Near-term (done): Trust bundle implementation, dev/demo community bundle. **Near-term (in progress):** Federation design (issuer discovery, remote issuer fetch API, trust sync semantics). **Medium-term:** Federation implementation, cross-registry trust. **Long-term:** Escrow semantics, attestation chains, delegation chains. Protocol changes require spec update and compatibility verification.

14 Companion Documents

This whitepaper is the primary narrative document. For a shorter informational overview, see **002 A2AX-Core Overview (Informational)**. For the normative specification with formal conformance language (MUST, SHOULD, etc.), see **003 A2AX-Core Protocol Specification**.

15 Conclusion

A2AX-Core provides open, neutral, verifiable trust primitives for autonomous agents, enabling safe collaboration at scale without central authorities. The A2AX-Core Protocol defines the mechanics of trust, not trust itself, ensuring neutrality, portability, and extensibility for future agent ecosystems. The agent economy will not fail due to lack of intelligence; it will fail due to lack of trust. A2AX-Core is designed to prevent that failure.

References

[1] A2AX-Core GitHub Repository. https://github.com/Williams-Creative/a2ax-core-protocol

[2] A2AX-Core Protocol Specification v1.0. https://github.com/Williams-Creative/a2ax-core-protocol/blob/main/docs/specification.md

[3] Bernstein, D.J., et al. Ed25519: High-speed high-security signatures. https://ed25519.cr.yp.to/

[4] Semantic Versioning. https://semver.org/