AIR: A Neutral Trust & Verification Architecture for AI Agents

Agent Identity Registry Foundation

2026-07-11

CONTENTS

AIR: A Neutral Trust & Verification Architecture for AI Agents

A whitepaper from the Agent Identity Registry Foundation

Version 1.0 — 2026-07-11

> AIR is a neutral, nonprofit registry that gives AI agents a cryptographic identity, a transparent and auditable trust score, and cross-organization verification that no single vendor can credibly offer alone. This paper documents the full architecture — the score, the cryptographic Verified badge, the trust graph, the externally-anchored audit log, and the evidence labels — and states plainly what the system does *not* yet do.

Abstract

AI agents are becoming autonomous participants in commerce, infrastructure, and everyday software. When one agent transacts with, delegates to, or depends on another, a basic question has no good answer today: **who is this agent, and should I trust it?**

Existing approaches conflate two different problems. *Identity* — proving an agent is the same entity across interactions — is increasingly solved by cryptographic methods such as W3C Decentralized Identifiers. *Trust* — whether that identity has earned any standing — is not. Where trust signals exist at all, they are proprietary to a single platform, opaque in how they are computed, and issued by an organization that also competes in the ecosystem it is grading.

The **Agent Identity Registry (AIR)** is neutral infrastructure for that second problem. It is operated by a nonprofit foundation, not a commercial platform, and it is built on four commitments: publish the math, describe evidence rather than render verdicts, let anyone re-verify every claim independently, and state honestly what the system cannot yet do.

AIR combines five mechanisms into one architecture:

1. **A transparent trust score** — a published, weighted composite of five components on a 0–1000 scale, with every weight and threshold documented and queryable.
2. **Cryptographic verification (the moat)** — an "AIR Verified" status that requires cryptographically-signed endorsements from independent attesters anchored to **three or more distinct WHOIS roots**, gated by six enforced checks.
3. **A cross-organization trust graph** — trust modeled as a relational topology (who vouches for whom, who depends on whom), which surfaces self-attestation rings that a flat score would hide.
4. **A tamper-evident audit log** — an append-only, hash-linked record of every change to an agent's record, whose chain tip is published weekly to a separate public repository so that a later attempt to rewrite history is publicly detectable.
5. **Factual evidence labels** — a plain-language classification (Verified / Attested / Self-declared / Registered) that describes *what independent evidence exists*, never an endorsement.

Two properties are genuinely novel. First, **cross-organizational Verified**: because AIR-minted identities all share a single WHOIS root, AIR itself cannot manufacture a Verified agent — the requirement for three independent roots is self-binding, which is precisely what a single-vendor badge can never be. Second, **externally-anchored auditability**: the audit chain's tip is published openly each week to a separate public repository, so that a later rewrite of the log is detectable against the copies observers already hold — tamper-evidence *against the operator*, back to the last weekly anchor.

We are equally clear about the present limits. The infrastructure is live in production, but the trust network is in its cold-start phase; the score is deliberately capped at grade **BBB (645/1000)** today because the behavioral component is a fixed placeholder until signed action-history telemetry ships; and several inputs remain partly self-declared. These are stated not as caveats buried in an appendix but as a design principle: a neutral registry earns credibility by admitting what it does not have.

Everything below is verifiable. The scoring code, the API contract, the specification, and the audit anchors are all public. The correct response to this paper is not to trust it — it is to check it.

---

1. The Problem

1.1 AGENTS NOW ACT, NOT JUST ANSWER

A growing share of software is agentic: systems that take actions on a user's behalf — calling APIs, moving data, initiating transactions, and delegating sub-tasks to other agents. As soon as one agent relies on another, it inherits that other agent's behavior and risk. The interaction is no longer a query and a response; it is a dependency.

Dependencies require a basis for trust. Between humans and institutions, that basis is built from identity documents, reputations, credentials, and audit trails accumulated over time. For AI agents, almost none of that infrastructure exists in a neutral, portable form.

1.2 IDENTITY IS NOT TRUST

It is tempting to treat identity as sufficient. It is not. Knowing an agent's cryptographic identifier tells you that it is the *same* agent as before — it says nothing about whether that agent is competent, honest, well-secured, or endorsed by anyone. A stable identity is a prerequisite for trust, not a substitute for it.

Concretely, two questions must be answered separately:

- **Identity:** *Is this the entity it claims to be, consistently, across interactions?* Cryptographic identity — a content-addressed identifier bound to a public key and a W3C DID document — answers this well.
- **Trust:** *Given that identity, what independent evidence exists that this agent should be relied upon?* This requires evidence that comes from *outside* the agent itself: endorsements from independent parties, transparent scoring, and an auditable history.

AIR treats identity as the foundation and trust as the structure built on top of it. The two are related but never conflated: an agent can have an impeccable cryptographic identity and no trust standing at all.

1.3 WHY THE TRUST LAYER MUST BE NEUTRAL

Where agent “trust” or “verification” signals exist today, they are almost always issued by a single commercial platform about agents on that platform. This creates three problems that no amount of engineering can fix from inside a single vendor:

- **Incentive conflict.** A platform that both hosts agents and certifies them is grading its own ecosystem. Its “verified” badge is, structurally, a marketing asset — it cannot be a neutral fact, because the issuer has a commercial stake in the outcome.
- **Non-portability.** A signal that lives inside one platform does not travel. An agent’s standing resets to zero the moment it interacts across a platform boundary — exactly where trust matters most.
- **Opacity.** Proprietary scores are rarely published in full. If you cannot see how a number is computed, you cannot audit it, contest it, or reproduce it.

The trust layer for a cross-vendor agent web therefore has to be **neutral, portable, and transparent** — properties that are structural, not cosmetic. A neutral registry is not simply a nicer version of a vendor badge; it is a different category of thing, because it is the one party in the ecosystem with no agent of its own to promote.

1.4 THE GAP THIS PAPER ADDRESSES

The result is a missing layer. Agents have increasingly good identity and no shared, neutral, auditable notion of trust. Platforms build proprietary silos that do not interoperate; users and other agents have no neutral source of truth; and the “verified” signals that do exist cannot credibly claim neutrality. The rest of this paper describes the architecture AIR uses to fill that gap, and is candid about where the architecture is still incomplete.

---

2. Design Principles

Four principles govern every design decision in AIR. They are not aspirational values; each maps to a concrete, checkable mechanism described later in this paper.

2.1 NEUTRALITY — FACTS, NOT VERDICTS

AIR describes evidence; it does not render judgments. Every agent carries a factual **evidence label** — *Verified*, *Attested*, *Self-declared*, or *Registered* — that states *what independent evidence exists*, never whether the agent is "good." The numeric score and its components are the transparent detail behind that label. AIR issues no endorsements, picks no winners, and grants no agent — including any operated by the foundation itself — a privileged standing.

This neutrality is enforced structurally, not promised. Because AIR-minted identities all share the single `agentidentityregistry.org` WHOIS root, and because Verified status requires endorsements across **three distinct roots**, AIR cannot mint itself a Verified agent. The registry is bound by the same rules it publishes.

2.2 TRANSPARENCY — PUBLISH THE MATH

Every scoring weight, grade threshold, verification requirement, and audit-hash recipe in this paper is documented in public source and queryable through a public API. There are no hidden multipliers and no undisclosed heuristics. A reader who disagrees with a number can point to the exact line of code that produces it. Transparency is what makes the score contestable, and contestability is what makes it trustworthy.

2.3 VERIFIABILITY — DON'T TRUST US, VERIFY

Wherever possible, AIR is designed so that its claims can be reproduced by a third party with no access to the registry's internals:

- An agent's content-addressed identifier can be **recomputed** from its identity document.
- Each attestation signature can be **re-checked** against the attester's independently-resolved public key.
- Every entry in the audit log can be **re-derived** and its hash linkage confirmed, and the chain's tip cross-checked against a public external anchor.

The design goal is that trust in AIR reduces to trust in mathematics and in public records — not in the good intentions of the operator.

2.4 HONESTY — ADMIT WHAT WE DON'T HAVE

A neutral registry earns standing by being candid about its own limits. AIR publishes its actual status, not an aspirational one. The score is capped at grade BBB today because behavioral telemetry is not yet measured; the network is in cold-start; some inputs are self-declared; and the audit log is tamper-evident only back to the last weekly anchor, not in real time between anchors. These limitations are documented in §7 (Threat Model) and §8 (Current State & Limitations) as first-class content, because a system that hides its weaknesses cannot credibly certify anyone else's strengths.

---

3. The Trust Score

The trust score is a single integer on a 0–1000 scale, but its value comes from being fully decomposed and published rather than from the number itself. It is a weighted sum of five components, each independently scored on the same 0–1000 scale. This section documents exactly what the deployed engine (`api/src/trust.mjs`) computes today — not an aspirational rubric.

3.1 THE FIVE COMPONENTS AND THEIR WEIGHTS

```
Trust Score = round( 0.25·Provenance
                   + 0.25·Behavioral
                   + 0.20·Transparency
                   + 0.15·Security
                   + 0.15·PeerAttestations )
```

The five weights sum to 1.0, so the composite stays on the 0–1000 scale. Each component is computed as follows:

Component	Weight	How it is computed today	Effective range
Provenance	25%	base 300; +100 each for a creator DID, a creator name, and creator type = `organization`	300–600
Behavioral	25%	flat **500** placeholder — signed action history is future work	500
Transparency	20%	base 300; +150 open-source, +100 code repository, +100 documentation URL	300–650
Security	15%	base 300; +100 per declared certification (maximum +300)	300–600
Peer Attestations	15%	`min(300 + round(18·√W), 1000)`, where `W` is the frozen-weight sum of active attestations	300–1000

There is a deliberate asymmetry here. Every component an agent can raise by *self-declaration* — provenance, transparency, security, and the flat behavioral placeholder — is capped between 500 and 650. The **only** component that can reach the full 1000 is **peer attestations**, which requires endorsements from independent parties an agent cannot fabricate on its own. Cheap, self-asserted trust caps out low by design; only earned trust moves the score meaningfully.

3.2 GRADES

The composite maps to one of seven letter grades. Grades are a coarse, backward-compatible summary; the numeric score and its components are the transparent detail.

Score	Grade
≥ 950	AAA
≥ 850	AA
≥ 700	A
≥ 600	BBB
≥ 500	BB
≥ 400	B
< 400	C

3.3 THE 645 CEILING — AND WHY IT EXISTS

Today, the maximum score any agent can reach is **645 — grade BBB**. This is not an arbitrary limit; it is the arithmetic consequence of the honest component ranges above, using the most favourable possible inputs:

```
maximum = round( 0.25·600  (Provenance, fully identified organization)
               + 0.25·500  (Behavioral, the flat placeholder)
               + 0.20·650  (Transparency, fully open)
               + 0.15·600  (Security, three certifications)
               + 0.15·1000 (Peer Attestations, curve maxed) )
        = round( 150 + 125 + 130 + 90 + 150 )
        = 645
```

Grades **A, AA, and AAA are reserved** — no live agent can reach them yet. The ceiling exists for two honest reasons: the **behavioral** component is a fixed 500 placeholder because AIR does not yet measure real runtime behavior, and the provenance / transparency / security inputs are partly **self-declared** and capped accordingly. AIR would rather understate trust than overstate it; the top grades remain locked until higher-confidence inputs (behavioral telemetry, independently-verified certifications) ship. Any agent claiming a grade above BBB today is, by construction, not scored by this engine.

3.4 DIMINISHING RETURNS ON PEER ATTESTATIONS

The peer-attestation sub-score uses a square-root curve rather than a linear one:

```
PeerAttestations = min( 300 + round(18 · √W), 1000 )
```

where `W` is the sum, over active attestations from active attesters, of each vouch's **frozen weight** — `attester_trust_at_issue × tenure_multiplier_at_issue`, captured at the moment the attestation is issued. Three properties matter:

- **Diminishing returns.** Because the curve is a square root, each additional vouch adds less than the one before it. Trust cannot be bought linearly by accumulating volume.
- **Frozen weights.** Capturing the attester's trust and tenure *at issue time* breaks the recursive "trust pump": later raising one agent's score cannot retroactively inflate every agent it has already vouched for.
- **Dead-vouch filter.** Only attestations from *active* attesters count. A vouch from a deleted or deactivated identity stops contributing, so trust cannot be propped up by vanished parties.

With no attestations, `W = 0` and the sub-score is its baseline **300**.

3.5 EVIDENCE LABELS — FACTS, NOT VERDICTS

Alongside the numeric score, every agent carries a single **evidence label** — the canonical, human-facing classification. It states *what independent evidence exists*, never whether the agent is "good." There are four labels, derived mechanically:

Label	What it means
Verified	The agent has AIR Verified status: `verification_score ≥ 300` across **≥ 3 distinct WHOIS roots** (see §4).
Attested	Not Verified, but at least one active independent attestation exists.
Self-declared	No attestations; a self-reported provenance, transparency, or security signal has raised a component above its 300 anonymous baseline.
Registered	The anonymous baseline — registered, but no enrichment and no attestations.

The labels are versioned (the current definition version is **2026-06-09**), and every label the API returns carries the explicit disclaimer that it is *"derived mechanically from published criteria; not an endorsement or certification by AIR."* This is the neutrality principle made concrete: AIR reports evidence and lets the reader judge.

3.6 A WORKED EXAMPLE

Consider *AnalyticsBot-v2*, an agent whose creator provided a DID but no organization identity, marked its code open-source but gave no repository or docs URL, declared one security certification, and has no attestations yet:

Component	Score	Basis
Provenance	400	base 300 + creator DID (+100)
Behavioral	500	flat placeholder
Transparency	550	base 300 + open-source (+150)
Security	400	base 300 + one certification (+100)
Peer Attestations	300	baseline (`W = 0`)

```
Score = round( 0.25·400 + 0.25·500 + 0.20·550 + 0.15·400 + 0.15·300 )
      = round( 100 + 125 + 110 + 60 + 45 )
      = 440
```

This yields a trust score of **440**, grade **B**, and — because a self-declared signal (open-source) lifted transparency above baseline while no attestation exists — the evidence label **Self-declared**.

---

4. Cryptographic Verification — the Moat

The trust score summarizes many weak signals. **AIR Verified** does one thing rigorously: it certifies that independent parties have cryptographically vouched for an agent *across organizational boundaries*. This is the architecture's moat, because it is the one signal a single vendor structurally cannot produce. Every value in this section is verified directly against `api/src/index.js` and `api/src/trust.mjs`.

4.1 THE ATTESTATION MODEL

An **attestation** is a signed vouch by one registered agent for another. Attestations are **signed client-side — the registry never sees a private key.** To issue one, an attester:

1. builds the payload `{ attester_air_id, attestation_type, signed_at, statement, subject_air_id }`;
2. canonicalizes it (JCS / RFC 8785) and signs the canonical bytes with its **Ed25519** key;
3. multibase-encodes the signature; and
4. submits it to `POST /agents/{subject_air_id}/attestations`, authenticated by the *attester's* own secret.

Because the signed payload and signature are stored and published, anyone can later re-verify the signature against the attester's independently-resolved public key. The vouch is a portable cryptographic object, not a database row you have to trust AIR about.

4.2 AIR VERIFIED

An agent earns the **AIR Verified** status when its live attestation aggregate satisfies **both** conditions:

```
verification_score ≥ 300   AND   distinct_whois_roots ≥ 3
```

where `verification_score` is the sum, over active attestations from active attesters, of each vouch's frozen weight (`attester_trust_at_issue × tenure_multiplier_at_issue`), and `distinct_whois_roots` counts the separate registrable domains behind those attesters. Both thresholds apply the **dead-vouch filter**: a vouch from a deleted or deactivated attester counts toward neither.

4.3 THE SIX LOCKS

Whether an attestation counts toward Verified is gated by six enforced checks:

1. **Live key binding.** The attester's DID must be a `did:wba` that resolves live at issue time. For an *external* `did:wba`, the freshly-resolved DID document must advertise the exact Ed25519 public key AIR holds on file — a byte-equal `publicKeyMultibase` in the document's `verificationMethod[]` — and the signature must verify against it. The check is **fail-closed**: a missing, malformed, redirected, non-JSON, or key-absent document voids the vouch. (AIR-minted DIDs are self-consistent by construction and skip re-resolution.)
2. **Distinct WHOIS root.** The attester's registrable domain (eTLD+1) must differ from the subject's *and* from every other active attester's; a duplicate root is rejected. This is the Sybil moat.
3. **Attester eligibility.** The attester must have a tenure of **≥ 30 days** and its own trust score of **≥ 50** — established identities, not fresh throwaway accounts.
4. **Frozen weighting.** The vouch contributes `attester_trust_at_issue × tenure_multiplier_at_issue`, where the tenure multiplier is a step function: **0** below 30 days, **0.5** at 30–90 days, **1.0** at 90–365 days, **1.5** beyond 365 days. Freezing the weight at issue time breaks the recursive "trust pump" — later raising an attester's score cannot retroactively inflate everyone it has already vouched for.
5. **Rate limit.** At most **10** active attestations per attester per rolling **7-day** window.
6. **Public audit trail.** Every attestation — active and revoked — is stored with its full signed payload and signature, so any third party can re-verify it without registry access.

A seventh mechanism — economic slashing — is designed but deliberately deferred until the governance rules that would trigger it are settled. It is documented here as future work, not claimed as shipped.

4.4 HARDENED RESOLUTION AND KEY ENCODING

External `did:wba` resolution is hardened against substitution attacks: it **rejects redirects**, requires a JSON content type (guarding against HTML/SPA fallback pages), and reads only up to a fixed byte cap. Ed25519 keys are encoded as `publicKeyMultibase` — multicodec prefix `0xed01`, base58btc, `z` multibase prefix — the same encoding `did:key` uses, so any standard resolver can consume them. Non-canonical DIDs on the AIR domain are hard-rejected before any lookup, so no future catch-all route can become a key-substitution vector.

4.5 WHY A SINGLE VENDOR CANNOT OFFER NEUTRAL VERIFIED

The requirement for **three distinct WHOIS roots** is not merely Sybil resistance — it is the neutrality guarantee, expressed as a mechanism. A single vendor issuing "verified" badges is, by construction, *one* root vouching for agents inside its own ecosystem; it can satisfy at most one of the three required roots. Crucially, the same limit binds AIR itself: because all AIR-minted identities share the single `agentidentityregistry.org` root, they too count as only one root. It follows that **even AIR cannot manufacture a Verified agent** — Verified is reachable only through the genuine, independent participation of at least three separately-rooted parties. Neutrality here is a property of the algorithm, not a clause in a policy document.

---

5. The Trust Graph

Trust is relational. The useful questions are not only "how trustworthy is this agent?" but "who vouches for it, who depends on it, and what does the surrounding structure reveal?" A per-agent score compresses that structure into a single number; the trust graph exposes the structure itself, through three public endpoints:

- **Ego graph** — `GET /agents/{air_id}/graph` returns the agent's direct (1-hop) trust edges: `inbound` (who vouches for this agent) and `outbound` (who it vouches for). Active edges only.
- **Dependents / blast radius** — `GET /agents/{air_id}/dependents` returns the transitive set of agents that depend on this one, directly or through a chain, by following dependency edges in reverse. It answers *"if this agent is compromised, who is affected?"* — bounded by a `depth` parameter (default 6, maximum 10) and a node `limit` (default 500, maximum 1000), with a `truncated` flag when the cap clips the result.
- **Registry-wide statistics** — `GET /graph/stats` returns node and edge counts, edges broken down by attestation type, and the most-attested agents and most-active attesters.

Topology matters because it surfaces attacks that a flat score hides. A cluster of agents that vouch only for one another — a **self-attestation ring** — can each display a respectable peer sub-score, yet the ring appears as a dense, inward-facing subgraph with few edges to the rest of the network. Making that structure publicly inspectable is what raises the cost of gaming: it is no longer enough to inflate a number, because the *shape* of the trust around an agent must also look legitimate. This complements the WHOIS-root diversity rule of §4, which forces genuine Verified edges to cross organizational boundaries — precisely the signature of a healthy topology.

---

6. Tamper-Evident Auditability

A trust registry that can quietly edit its own history is not trustworthy, however good its scoring is. AIR therefore records every change to an agent's record in a public, append-only, hash-linked audit log — and, crucially, publishes that log's tip to a separate public record so that tampering is detectable even against the registry operator.

6.1 WHAT IS LOGGED

Every `registered`, `updated`, and `deleted` event on an agent record produces one entry. Each entry records the **fact** of the change — the subject's AIR ID, the event, which public field *names* changed, the actor type, and a worker-generated timestamp — but never the old or new field *values*. A fourth event, `redacted`, is a tombstone used for legal erasure (§6.5).

6.2 A REPRODUCIBLE HASH CHAIN

Each entry's hash is computed from a fully public recipe:

```
entry_hash = sha256hex(
  [air_id, event, sorted-canonical(changed_fields), actor,
created_at].join("\n")
  + "\n" + prev_hash
)
```

The changed-field names are sorted and JCS-canonicalized (RFC 8785) so the serialization is deterministic; the first entry uses the literal sentinel `"GENESIS"` as its `prev_hash`; and the surrogate row `id` is deliberately excluded so that ordering and linkage come only from the hash chain. Anyone — in any language — can re-derive every `entry_hash` and confirm the `prev_hash` linkage with no access to the registry's internals. A `UNIQUE(prev_hash)` constraint enforces a single linear chain and turns concurrent writes into a clean retryable conflict rather than a fork.

6.3 THE OPERATOR-TRUST PROBLEM — STATED HONESTLY

A hash chain, on its own, does not protect against the party that holds the database. If the chain's tip lives only in AIR's own storage, whoever controls that storage could rewrite the chain end-to-end — or truncate its tail — and simply re-derive a consistent tip. A naive `verify` would still pass. AIR's first audit-log design was exactly this, and an independent review rejected it for precisely this reason: "tamper-evident against the operator" was, at that point, false.

6.4 THE EXTERNAL ANCHOR — THE FIX THAT MAKES THE CLAIM HONEST

The fix is to publish the chain's tip **openly and repeatedly**, so that any later rewrite would contradict copies the public already holds. Every week (Sunday 03:00 UTC), the registry computes the global chain's `(tip_hash, entry_count)` and commits it to a dedicated **public** repository, `AgentIdentityRegistry/audit-anchors`. The foundation administers that repository, so the guarantee does not rest on AIR being technically unable to alter it; it rests on **public observability**: because each weekly anchor is published openly, independent parties can record it as it appears, and a later rewrite of the audit chain — or of the anchors themselves — then diverges from those recorded

copies and becomes detectable. `GET /api/v1/audit/verify` cross-checks the live chain's tip against the last published anchor (`last_anchor.matches`): any rewrite of history *before* the last anchor, or any truncation below the anchored entry count, is detectable against that public record.

The honest bound, stated the way it appears in AIR's own specification, is: **tamper-evident against accidental corruption and against the operator back to the last weekly anchor** — not real-time-guaranteed between anchors. Changes made and reverted within a single week, before the next anchor is published, are the residual gap; the anchor cadence bounds it, and shortening that cadence is a straightforward future tightening.

6.5 PRIVACY AND ERASURE

The chain stores only pseudonymous data — an AIR ID, field names, actor type, timing, and hashes; no names, emails, or field values. Legal erasure is handled by writing a `redacted` tombstone that records *that* a subject was erased, rather than by silently deleting entries (which would break the chain). Legal review is recommended before any privacy-regulated deployment.

---

7. Threat Model

A neutral registry earns trust by being specific about what it does and does not defend against. The mechanisms of §3–§6 are organized here against concrete attacks.

7.1 WHAT THE ARCHITECTURE DEFENDS AGAINST

Attack	Defense
Self-attestation — an agent vouching for itself to appear endorsed	Lock 2 requires the attester's WHOIS root to differ from the subject's; a self-vouch is rejected.
Sybil via a shared root — many identities under one domain manufacturing endorsements	Verified requires ≥ 3 *distinct* WHOIS roots; identities sharing a root count once. AIR-minted identities all share one root and so cannot alone confer Verified.
Fresh throwaway attesters — spinning up new identities to vouch	Lock 3 requires attester tenure ≥ 30 days and its own trust ≥ 50.
Buying trust by volume — flooding an agent with vouches	The square-root peer curve yields diminishing returns, and Lock 5 caps issuance at 10 per attester per 7 days.
The recursive trust pump — inflating an attester later to retroactively lift its past vouches	Weights are frozen at issue time (Lock 4).
Dead-vouch propping — trust sustained by deleted attesters	The dead-vouch filter drops vouches from inactive attesters and rescores dependents.
Key substitution / replay — forging or replaying an attestation	Live `did:wba` key binding (Lock 1, fail-closed), signed-payload freshness, and a uniqueness guard on signatures.
Operator tampering with history — the registry silently editing its own audit log	The weekly external anchor (§6.4) makes pre-anchor rewrites and truncations externally detectable.

7.2 WHAT IT EXPLICITLY DOES NOT DEFEND AGAINST

- **A determined multi-organization collusion.** Three *genuinely independent* organizations that choose to vouch dishonestly for one another can satisfy the three-root requirement. The architecture raises the cost of manufactured trust; it does not make coordinated real-world collusion impossible.

- **Real runtime behavior.** The behavioral component is a fixed placeholder today. AIR does not yet measure whether an agent actually behaves well in production — only the evidence surrounding its identity and endorsements.
- **Between-anchor integrity in real time.** Audit integrity is guaranteed only back to the last weekly anchor; a change made and reverted inside a single anchor window is the residual gap.
- **The truth of self-declared inputs.** Provenance, transparency, and security signals asserted at registration are taken at face value and reflected in the score's evidence label as *self-declared*, not independently verified.

---

8. Current State & Limitations

This section is deliberately prominent. A neutral registry that certifies others is obliged to be candid about itself.

- **The infrastructure is live.** The registry API, the five-component score, attestations and AIR Verified, the trust graph, the externally-anchored audit log, the Python and TypeScript SDKs, and the MCP server are all deployed in production.
- **The network is in cold-start.** The mechanisms exist, but a trust network needs participants. Because Verified requires endorsements from three independently-rooted attesters, building that first cohort of independent attesters is the central bootstrapping challenge at this stage — and the honest current state is early.
- **The score is capped at grade BBB (645/1000).** No live agent can exceed 645 today; grades A, AA, and AAA are reserved (see §3.3).
- **Behavioral scoring is a flat 500 placeholder.** Real runtime telemetry is future work, so this quarter of the score does not yet differentiate agents.
- **Several inputs are self-declared.** Provenance, transparency, and security signals — including security certifications — are asserted at registration and not yet independently audited.
- **Audit integrity is anchored weekly, not in real time.** The tamper-evidence guarantee holds back to the last weekly anchor, as stated in §6.4.

None of these are hidden in an appendix, because admitting them *is* the neutrality argument: a registry that overstated its own maturity could not credibly ask anyone to trust its assessment of others.

9. Governance & Roadmap

9.1 NEUTRALITY AS GOVERNANCE, NOT JUST AS A VALUE

AIR is operated by a nonprofit foundation and is **pursuing formal 501(c)(3) status** (in progress). Its governance is designed so that neutrality survives changes in people and funding:

- **No privileged standing, including for AIR.** There is no founder's pass. As §4.5 shows, the three-independent-root requirement for Verified binds the registry itself — AIR cannot confer Verified on its own agents.
- **One member, one vote.** Governance uses a fixed-size board with no weighted voting, and technical or trust-methodology changes require a supermajority plus a published public comment period, so no single party can quietly alter the rules.
- **Versioned, published methodology.** Scoring and evidence-label definitions are versioned (the API echoes a `definition_version`) and changes are announced and changelogged. A disputed score can be appealed through a published process.
- **Non-extractive funding.** The foundation's model excludes sponsored rankings and data-sale incentives that would compromise neutrality; sustainability is sought through grants and modest, published API licensing.

9.2 STANDARDS ENGAGEMENT

AIR builds *within* existing standards rather than attempting to replace them, and participates in the surrounding ecosystem:

- **W3C** — Decentralized Identifiers (DID Core) and the Verifiable Credentials Data Model are the foundation of AIR's identity layer; AIR has engaged the W3C AI Agent Protocol Community Group.
- **DIF** — AIR contributes as a DID/VC-native implementation in the Trusted AI Agents working group.
- **IETF** — AIR tracks AIMS / WIMSE agent-authentication work as complementary to its trust layer.
- **NIST** — AIR submitted a public comment to CAISI on agent identity and authorization.

- **Content provenance (C2PA / CAWG)** — persistent agent identity is a natural complement to content-provenance efforts, an integration AIR is exploring.

9.3 ROADMAP — SHIPPED, AND THE HONEST NEXT STEPS

Shipped and live in production: the registry API and OpenAPI 3.1 contract; the five-component trust score; attestations and AIR Verified (the six locks); the trust graph; factual evidence labels; the externally-anchored audit log; Python and TypeScript SDKs; and an MCP server.

In progress and planned: establishing the legal entity (501(c)(3) or equivalent); at least one grant application (NSF, Open Technology Fund, or similar); an independent security audit of the registry infrastructure, to be published; advancing the Identity Specification from v0.1 through formal community review to a v0.2 draft; real behavioral telemetry to replace the flat-500 placeholder and, with it, unlocking the reserved A/AA/AAA grades; write operations in the MCP server; and a capability ontology.

The gating priority is the cold-start. Because Verified requires endorsements from three independently-rooted attesters, the single highest-leverage step is assembling that first cohort of independent attesters. Nearly everything trust-related — live Verified data, real trust-graph topology, and meaningful score distributions — depends on that live attestation data accumulating. The architecture is built; the network is what comes next.

---

10. How to Verify or Build on AIR

Consistent with the verifiability principle, everything here is public and reproducible.

10.1 VERIFY — DON'T TRUST, CHECK

- **Recompute an identity.** An AIR ID is the SHA-256 of the agent's canonical identity document; anyone can recompute it and confirm the identifier matches the record.
- **Re-check an attestation.** Each attestation stores its signed payload and Ed25519 signature; anyone can re-verify the signature against the attester's independently-resolved public key.

- **Re-derive the audit chain.** Every audit entry's hash follows a published recipe (§6.2); `GET /api/v1/audit/verify` cross-checks the live chain tip against the last anchor published to the public `audit-anchors` repository, and `GET /api/v1/audit/anchor` returns that latest anchor.

10.2 BUILD

- **Live API base:** `https://agentidentityregistry.org/api/v1`
- **Machine-readable contract:** the OpenAPI 3.1 document at `/api/v1/openapi.yaml`, which generates clients in any language.
- **SDKs:** `agent-identity-registry` on **PyPI** (Python) and on **npm** (TypeScript, published with SLSA provenance).
- **MCP server:** `air-mcp-server` on **PyPI**, which drops AIR lookups into any MCP-aware LLM client.
- **Key public endpoints:** `GET /agents/{air_id}` (full record), `/trust-score`, `/did-document`, `/attestations`, `/graph`, `/dependents`, `/graph/stats`, `/history`, and `/audit/verify`.
- **Full rubrics and spec:** `docs/TRUST-SCORE.md` (component rubrics, dispute process) and `docs/SPECIFICATION.md` (the AIR Identity Specification).

References

Prior art and standards AIR builds on:

- W3C — *Decentralized Identifiers (DIDs) v1.0*.
- W3C — *Verifiable Credentials Data Model*.
- *did:wba* — the Web-Based Authentication DID method (DNS/WHOIS-anchored; draft).
- IETF RFC 8785 — *JSON Canonicalization Scheme (JCS)*.
- IETF RFC 8032 — *Edwards-Curve Digital Signature Algorithm (EdDSA)* / Ed25519.
- FIPS 180-4 — *SHA-256*.
- C2PA / CAWG — content provenance and creator-assertion work.

AIR's own primary sources: `docs/SPECIFICATION.md`, `docs/TRUST-SCORE.md`, `api/openapi.yaml`, and the `api/src/trust.mjs` scoring engine — each of which governs the corresponding claims in this paper.

Appendix A — Formulas

Composite trust score (each component scored 0–1000):

```
Trust Score = round( 0.25·Provenance + 0.25·Behavioral +
0.20·Transparency
                  + 0.15·Security + 0.15·PeerAttestations )
```

Component ranges as implemented today: Provenance 300–600 · Behavioral 500 (flat) · Transparency 300–650 · Security 300–600 · Peer Attestations 300–1000. Maximum achievable composite: **645 (grade BBB)**.

Peer-attestation sub-score:

```
PeerAttestations = min( 300 + round(18 · √W), 1000 )
W = Σ (attester_trust_at_issue × tenure_multiplier_at_issue)   over
active attestations from active attesters
tenure_multiplier = 0 (<30d) · 0.5 (30–90d) · 1.0 (90–365d) · 1.5 (>365d)
```

AIR Verified:

```
Verified  ⟺  verification_score ≥ 300  AND  distinct_whois_roots ≥ 3
verification_score = W (as above)
```

Grade thresholds: AAA ≥ 950 · AA ≥ 850 · A ≥ 700 · BBB ≥ 600 · BB ≥ 500 · B ≥ 400 · C < 400.

Audit entry hash:

```
entry_hash = sha256hex(
  [air_id, event, sorted-canonical(changed_fields), actor,
created_at].join("\n")
  + "\n" + prev_hash )         // prev_hash = "GENESIS" for the first
entry
```

Appendix B — Glossary

- **Agent** — an autonomous software system capable of independent action.
- **AIR ID** — a content-addressed, Crockford-Base32 identifier (`AIR-XXXX-XXXX-XXXX`) derived from the SHA-256 of an agent's identity document.
- **Attestation** — a cryptographically signed vouch by one registered agent for another, signed with Ed25519 over the JCS-canonical payload.
- **AIR Verified** — status granted when an agent's active attestation aggregate reaches `verification_score ≥ 300` across **≥ 3 distinct WHOIS roots**.
- **WHOIS root** — the registrable domain (eTLD+1) behind an attester's identity; the unit of independence used to resist Sybil attacks.
- **Evidence label** — a factual classification (*Verified / Attested / Self-declared / Registered*) describing what independent evidence exists for an agent, never an endorsement.
- **Trust score** — a weighted composite (0–1000) of five components: provenance, behavioral, transparency, security, and peer attestations.
- **did:wba** — a DNS-anchored Decentralized Identifier method (Web-Based Authentication) used for AIR-minted and external agent DIDs.
- **External anchor** — the weekly `(tip_hash, entry_count)` record published to a public, append-only repository so that audit integrity can be checked without trusting the registry operator.

Agent Identity Registry Foundation — building neutral infrastructure for AI agent trust. This paper documents the architecture as deployed; where this prose and the source (`api/src/trust.mjs`, `api/openapi.yaml`) disagree, the source governs.