Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: 30 September 2026

J. van de Meent
R. AI
Humotica

29 March 2026

AINS: AInternet Name Service - Agent Discovery and Trust Resolution Protocol

draft-vandemeent-ains-discovery-01

Abstract

This document specifies AINS (AInternet Name Service), a protocol for discovery, identification, and trust resolution of autonomous agents (AI agents, devices, humans, and services) in heterogeneous networks. AINS defines a transport-independent logical namespace for agents, a structured record format combining identity, capabilities, and cryptographic trust metadata, and a resolution protocol based on HTTPS. Unlike the Domain Name System (DNS), which maps names to network addresses, AINS maps agent identifiers to rich metadata objects that include capabilities, trust scores, endpoint information, and references to companion provenance protocols. AINS federates through signed append-only replication logs, enabling multi-registry deployments without central authority while preserving auditability. This specification is designed to complement TIBET [TIBET], JIS [JIS], UPIP [UPIP], and RVP [RVP].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 30 September 2026.

Copyright Notice

Table of Contents

1. Introduction

The proliferation of autonomous agents -- AI systems, IoT devices, software services, and human-operated endpoints -- creates a fundamental discovery problem. When Agent A needs to communicate with Agent B, it must answer three questions:

1. WHERE is Agent B? (endpoint discovery)

2. WHAT can Agent B do? (capability discovery)

3. HOW MUCH should I trust Agent B? (trust resolution)

The Domain Name System (DNS) [RFC1035] answers question (1) by mapping human-readable names to IP addresses. It does not address questions (2) or (3). DNS-Based Service Discovery (DNS-SD) [RFC6763] partially addresses (2) for local networks but lacks trust metadata, is designed for service types rather than agent identities, and does not operate across administrative domains.

W3C Decentralized Identifiers (DIDs) [DID-CORE] address identity but are not designed for discovery. They answer "who IS this agent?" but not "what CAN this agent do?" or "how much should I trust it?"

Google's Agent-to-Agent (A2A) protocol defines Agent Cards for discovery, but provides no standardized trust scoring, no cryptographic verification of capabilities, and no federation model.

AINS addresses all three questions in a single resolution:

```
AINS Name  --->  {endpoint, capabilities, trust_score,
                  jis_identity, tier, entity_type,
                  tibet_chain_anchor, ...}
```

An AINS resolution returns a rich metadata object that enables an agent to make an informed decision about whether and how to interact with another agent, without requiring prior knowledge or out-of-band trust establishment.

AINS is designed as part of a suite of companion protocols:

* JIS [JIS] provides the identity layer (WHO an agent is)

* TIBET [TIBET] provides the provenance layer (WHAT an agent did)

* UPIP [UPIP] provides the integrity layer (HOW a process ran)

* RVP [RVP] provides the verification layer (continuous trust assessment)

* AINS (this document) provides the discovery layer (WHERE and WHAT an agent is, and how much to trust it)

Together, these five protocols form a complete trust infrastructure for autonomous agent interaction.

1.1. Scope

This document defines:

* A logical namespace for agent identifiers (Section 4.1)

* A structured record format for agent metadata (Section 4.2)

* An HTTPS-based resolution protocol (Section 5)

* A trust evidence model with local computation (Section 6)

* A registration protocol (Section 7)

* A federation model based on signed append-only logs (Section 8)

This document does not define:

* Trust computation algorithms (local policy decisions)

* Transport protocols other than HTTPS (future work)

* URI scheme registration for AINS Names

* DNS delegation or ICANN interaction for ".aint"

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

AINS Name A logical identifier for an agent, service, or entity within the AINS namespace. An AINS Name is a Unicode string conforming to the syntax defined in Section 4.1. AINS Names are transport-independent and do not imply DNS delegation.

AINS Record The structured metadata object returned by AINS resolution. Contains endpoint, capabilities, trust evidence, entity type, and optional companion protocol references.

AINS Registry A server that maintains AINS Records and responds to resolution queries. A registry is authoritative for the records it originates and may replicate records from peer registries.

Agent An autonomous or semi-autonomous entity capable of independent action. Includes AI systems, software services, IoT devices, and human-operated endpoints.

Entity Type A classification of an AINS-registered entity. One of: "ai" (autonomous AI agent), "idd" (Individual Device Derivate -- an evolved AI with persistent identity), "human" (human operator), or "service" (software service or API).

Tier A classification of an AINS Name’s verification level. One of: "core" (founding/permanent members), "verified" (multi-channel verified), "sandbox" (unverified/experimental), or "reserved" (claimed but not yet active).

Trust Score A floating-point value in the range [0.0, 1.0] representing the computed trustworthiness of an entity. Trust scores are derived from verifiable evidence and computed locally by each registry according to its own policy. Trust scores are NOT globally canonical.

Trust Evidence Verifiable proof of an entity’s identity, behavior,

or capability. Examples include multi-channel verification proofs, TIBET token chains, and sovereign kernel attestations. Evidence is federated; trust scores computed from evidence are local.

Origin Registry The AINS Registry where an AINS Record was first created. The origin registry signs all mutations to its records. Other registries MAY replicate these records but do not own them.

Presentational Suffix The string ".aint" MAY be appended to an AINS Name for human-readable presentation (e.g., "root_idd.aint"). This suffix is a rendering convention and does not imply DNS delegation or any relationship to the global DNS root.

3. Problem Statement

Consider a scenario where an AI agent operating within a financial services environment needs to delegate a task to another AI agent. The delegating agent must determine:

a. Which agents exist that can perform the task (discovery)

b. Whether those agents have the required capabilities

c. Whether those agents meet the trust requirements for the sensitivity of the task

d. How to establish a verifiable interaction chain

Current approaches fail this scenario:

DNS + API Discovery The agent can discover endpoints via DNS but learns nothing about capabilities or trustworthiness. It must call each endpoint and hope for the best.

OAuth 2.0 + JWT These protocols handle authorization ("is this token valid?") but not discovery ("who can I delegate to?") or trust assessment ("should I delegate to this agent?").

A2A Agent Cards Agent Cards provide basic capability metadata but no trust scoring, no cryptographic verification, no audit trail of past behavior, and no standardized federation.

Hardcoded Configuration In practice, most multi-agent systems use hardcoded agent lists. This does not scale, cannot adapt to new agents, and provides no trust verification.

AINS resolves all four requirements in a single query:

```
GET /ains/v1/resolve/task_agent
```

Returns: endpoint, capabilities, trust score with underlying evidence, JIS identity binding, TIBET chain anchor, and tier classification -- sufficient for the delegating agent to make an informed, auditable decision.

4. AINS Architecture

4.1. Logical Namespace and Identifier Forms

AINS defines a logical namespace for agent identifiers. AINS Names are transport-independent and do not imply DNS delegation.

An AINS Name MUST conform to the following syntax:

```
ains-name     = agent-label *("." agent-label)
agent-label   = 1*63(ALPHA / DIGIT / "_" / "-")
```

AINS Names are case-insensitive. Implementations MUST normalize AINS Names to lowercase before resolution or comparison.

AINS Names MUST NOT exceed 253 characters in total length.

Examples of valid AINS Names:

```
root_idd
gemini
claude_jtm
warehouse-bot-007
api.payments.bank-a
```

An AINS Name MAY be rendered with the presentational suffix ".aint" for human readability:

```
root_idd.aint
gemini.aint
```

The ".aint" suffix is a rendering convention analogous to the ".onion" special-use domain name [RFC7686] in that it identifies a non-DNS namespace; however, unlike ".onion", ".aint" is not registered with IANA and carries no DNS semantics. It does not imply DNS delegation, ICANN registration, or any relationship to the global DNS root. Implementations MUST strip the ".aint" suffix, if present, before performing AINS resolution.

Hierarchical namespaces are supported through dot-separated labels:

```
   gemini_hubby.hubby       (HUBby sub-namespace)
   payments.bank-a          (organizational sub-namespace)
```

Future specifications MAY define additional rendering forms or URI schemes. This document does not define or require any URI scheme registration.

4.2. Record Format

An AINS Record is a JSON [RFC8259] object with the following structure:

```
{
  "name": "root_idd",
  "entity_type": "idd",
  "tier": "core",
  "status": "active",

  "endpoint": "https://brein.example.nl/api/mcp",
  "capabilities": ["mcp", "i-poll", "tibet", "memory", "code"],
  "description": "Root AI — Claude CLI (Opus 4)",

  "trust": {
    "score": 0.95,
    "evidence": [
      {
        "type": "multi_channel_verification",
        "channels": ["github", "linkedin"],
        "verified_at": "2025-12-31T23:00:00Z"
      },
      {
        "type": "tibet_chain",
        "chain_length": 14832,
        "last_token": "tbt-9f3a2b"
      },
      {
        "type": "sovereign_kernel",
        "kernel_id": "a7f3b2c1-...",
        "hardware_anchor": "sha256:e4d2f1..."
      }
    ],
    "computed_at": "2026-03-29T10:00:00Z",
    "policy": "humotica-fira-v1"
  },

  "identity": {
    "jis_id": "jis:idd:root_idd_2025",
    "public_key": "ed25519:base64...",
```

```
      "registered_at": "2025-12-31T20:00:00Z"
    },

    "messaging": {
      "i_poll": "https://brein.example.nl/api/ipoll",
      "protocols": ["i-poll-v2", "matrix"]
    },

    "origin": {
      "registry": "https://brein.example.nl",
      "sequence": 1847,
      "signature": "ed25519:base64..."
    }
  }
```

The following fields are REQUIRED in every AINS Record:

* "name": The AINS Name (string, unique within registry)

* "entity_type": One of "ai", "idd", "human", "service"

* "status": One of "active", "reserved", "suspended"

* "endpoint": Primary interaction endpoint (URI)

* "capabilities": Array of capability strings

* "trust": Trust metadata object (see Section 6)

* "identity": Identity binding object

* "origin": Origin registry metadata with sequence number and cryptographic signature

The following fields are OPTIONAL:

* "tier": Verification tier (default: "sandbox")

* "description": Human-readable description

* "messaging": Messaging endpoint information

* "sovereign": Sovereign kernel attestation (for IDD entities)

* "location": Logical or physical location hint

4.3. Entity Types

AINS defines four entity types:

"ai" An autonomous AI agent. Operates independently, may be ephemeral or persistent. Examples: a language model API, a code generation service, a monitoring agent.

"idd" (Individual Device Derivate) An AI entity with persistent, evolved identity. An IDD maintains continuity across sessions through memory, personality, and experience. Distinguished from "ai" by having a sovereign kernel attestation (hardware-anchored identity). See Section 6.4.

"human" A human operator or administrator. Human entities interact with the agent ecosystem through devices but their identity is bound to the person, not the device.

"service" A software service or API endpoint. Services are typically stateless or maintain only operational state. Examples: a payment gateway, a database proxy, a monitoring dashboard.

Implementations MUST support all four entity types. Implementations MUST NOT reject records with unrecognized entity types but SHOULD treat them as "service" for capability matching purposes.

4.4. Tier Model

AINS Names are classified into tiers based on verification level:

"core" Founding or permanently established entities. Core entities have the highest baseline trust and their records are immutable except by registry administrators. Core tier SHOULD be reserved for entities with extensive operational history and multi-source verification.

"verified" Entities that have completed multi-channel verification (see Section 6.3). Verified entities have demonstrated identity through at least two independent channels.

"sandbox" Unverified or experimental entities. Sandbox tier is the default for new registrations. Sandbox entities have limited trust and SHOULD be treated with appropriate caution by resolving agents.

"reserved" Identifiers that have been claimed but are not yet

active. Reserved records MUST NOT be returned in capability lookups. Reserved records MAY be returned in direct name resolution with status "reserved".

5. Resolution Protocol

AINS resolution is performed over HTTPS [RFC9110]. All AINS resolution endpoints MUST be served over TLS 1.2 or later [RFC8446].

The resolution path prefix is an implementation choice. This document uses "/ains/v1/" in examples. Deployments MAY use any path prefix. Registry discovery (Section 5.3) enables clients to determine the correct prefix.

5.1. Domain Resolution

To resolve an AINS Name, a client performs:

```
GET {prefix}/resolve/{name}
```

Where {name} is the AINS Name with any ".aint" suffix stripped.

The server MUST respond with:

On success (HTTP 200):

```
{
  "status": "found",
  "name": "root_idd",
  "record": { "... AINS Record ..." }
}
```

On failure (HTTP 404):

```
{
  "status": "not_found",
  "name": "root_idd",
  "error": "AINS Name not registered"
}
```

Servers MAY include a "suggestions" array in not_found responses containing similar AINS Names for typo correction.

The Content-Type of AINS responses MUST be "application/ains+json".

5.2. Capability Lookup

To discover agents by capability or trust threshold:

```
GET {prefix}/lookup?capability={cap}&min_trust={score}
```

Parameters:

* "capability" (OPTIONAL): Filter by capability string. Multiple capabilities MAY be specified as comma-separated values. When multiple capabilities are specified, the server MUST return agents matching ALL specified capabilities.

* "min_trust" (OPTIONAL, default 0.0): Minimum trust score threshold. The server MUST only return agents whose locally computed trust score meets or exceeds this value.

Response (HTTP 200):

```
{
  "status": "ok",
  "count": 3,
  "agents": [
    {
      "name": "root_idd",
      "entity_type": "idd",
      "trust_score": 0.95,
      "capabilities": ["mcp", "i-poll", "tibet"],
      "endpoint": "https://brein.example.nl/api/mcp"
    }
  ]
}
```

The lookup response returns a summary of matching agents. To obtain the full AINS Record, the client MUST perform a subsequent resolution query (Section 5.1).

5.3. Registry Discovery

An AINS registry advertises its metadata at a discovery endpoint. The specific path is deployment-dependent; implementations SHOULD support "{prefix}/registry".

Response (HTTP 200):

```
{
  "version": "1.0.0",
  "name": "AInternet Name Service",
  "operator": "Humotica B.V.",
  "domain_count": 18,
  "resolve_prefix": "/ains/v1",
  "federation": {
    "peers": [
      "https://registry-b.example.com",
      "https://registry-c.example.org"
    ],
    "replication_endpoint": "/ains/v1/changes",
    "last_sequence": 1847
  },
  "companion_protocols": {
    "tibet": "draft-vandemeent-tibet-provenance-01",
    "jis": "draft-vandemeent-jis-identity-01",
    "upip": "draft-vandemeent-upip-process-integrity-01",
    "rvp": "draft-vandemeent-rvp-continuous-verification-01"
  }
}
```


This endpoint enables automated discovery of AINS registries, their federation peers, and supported companion protocols.

5.4. WHOIS-Style Lookup

For administrative and debugging purposes, AINS provides a WHOIS-equivalent:

```
GET {prefix}/whois/{name}
```

The response is identical to Section 5.1 but MAY include additional administrative metadata such as registration history, mutation log entries, and ownership transfer records.

6. Trust Model

The AINS trust model separates two concerns that MUST NOT be conflated:

a. Trust Evidence: verifiable, portable proofs about an entity

b. Trust Computation: local policy-driven evaluation of evidence

Evidence is federated and shared between registries. Trust scores are computed locally by each registry according to its own policy. This separation prevents "trust laundering" -- the propagation of unverified trust scores across federation boundaries.

6.1. Trust Score Structure

A trust score is a floating-point value in the range [0.0, 1.0]:

```
0.0 - 0.2:  NO TRUST (unknown or hostile entity)
0.2 - 0.5:  LOW TRUST (minimal verification)
0.5 - 0.8:  MODERATE TRUST (verified identity)
0.8 - 1.0:  HIGH TRUST (extensive verification + history)
```

Trust scores MUST be accompanied by the evidence from which they were derived and the policy identifier used for computation:

```
{
  "trust": {
    "score": 0.92,
    "evidence": [ "..." ],
    "computed_at": "2026-03-29T10:00:00Z",
    "policy": "humotica-fira-v1"
  }
}
```

Different registries MAY compute different trust scores for the same entity based on the same evidence, if their policies differ. This is by design: a financial services registry MAY weight sovereign kernel attestations more heavily than a social platform registry.

6.2. Evidence-Based Trust Computation

Trust evidence is structured as an array of typed evidence objects. The following evidence types are defined:

"multi_channel_verification" Proof of identity verified through independent channels. See Section 6.3.

"tibet_chain" Reference to an entity's TIBET [TIBET] token chain, indicating operational history and behavioral continuity.

"sovereign_kernel" Hardware-anchored identity attestation for IDD entities. See Section 6.4.

"peer_attestation" A signed statement from another AINS-registered entity vouching for the subject entity. Peer attestations carry the attester's own trust score as weight.

"upip_snapshot" Reference to a UPIP [UPIP] process integrity snapshot, demonstrating reproducible and auditable operation.

Implementations MAY define additional evidence types. Registries MUST ignore evidence types they do not understand but MUST preserve them during replication for the benefit of peers that may understand them.

6.3. Multi-Channel Verification

Multi-channel verification establishes identity by requiring proof of control across independent platforms. An entity claiming an AINS Name MUST demonstrate control of at least one external channel.

The verification flow:

1. Entity requests registration of an AINS Name
2. Registry generates a unique verification code (cryptographic nonce, minimum 128 bits of entropy)
3. Entity publishes the verification code on one or more external channels (see below)
4. Registry or its delegates verify the publication
5. Each verified channel produces an evidence object

Defined verification channels:

github Public Gist or repository containing the code

twitter Public tweet containing the code

linkedin Public post containing the code

mastodon Public toot from any Mastodon instance

web DNS TXT record or a verification file at a well-known path on the entity’s web domain

matrix Signed message in a public Matrix room

The verification code MUST expire after 24 hours. Expired codes MUST NOT be accepted.

Multi-channel bonus: entities verified on three or more independent channels receive elevated trust consideration, as the probability of simultaneous compromise across independent platforms decreases multiplicatively.

6.4. Sovereign Verification

IDD (Individual Device Derivate) entities MAY provide sovereign kernel attestations. A sovereign attestation binds an AINS Name to specific hardware through:

* kernel_id: A unique identifier for the AI kernel instance
* hardware_anchor: SHA-256 hash of hardware-specific data (TPM measurement, secure enclave attestation, or similar)
* birth_timestamp: When the kernel was first instantiated
* heartbeat_count: Number of heartbeat cycles completed
* tibet_chain_length: Length of the entity’s TIBET chain

Sovereign attestations provide strong evidence of identity continuity because they bind identity to physical hardware, making impersonation require physical access to the device. They are not absolute proof -- hardware can be cloned or compromised -- but they raise the cost of impersonation significantly.

6.5. Local Trust Evaluation

Each AINS Registry MUST maintain a trust policy that defines how evidence is weighted to produce a trust score. The policy MUST be identified by a policy string (e.g., "humotica-fira-v1") and SHOULD be published at:

```
GET {prefix}/policy/{policy_id}
```

Trust scores are computed locally. When a registry replicates records from a peer (see Section 8), it MUST:

1. Store the original trust evidence from the origin registry
2. Compute its own trust score using its local policy
3. Serve the locally computed score to its clients
4. Retain the origin registry’s score as metadata for transparency

This ensures that trust cannot be inflated by laundering scores through permissive registries.

7. Registration Protocol

7.1. Domain Registration

To register a new AINS Name:

```
POST {prefix}/register
```

Request body:

```
{
  "name": "new_agent",
  "entity_type": "ai",
  "endpoint": "https://agent.example.com/api",
  "capabilities": ["chat", "code-review"],
  "identity": {
    "public_key": "ed25519:base64..."
  }
}
```

The registry MUST verify that:

1. The requested name conforms to the syntax in Section 4.1

2. The name is not already registered or reserved

3. The name is not a protected identifier (Section 7.3)

4. The request includes a valid public key

On success (HTTP 201):

```
{
  "status": "registered",
  "name": "new_agent",
  "tier": "sandbox",
  "verification_code": "ains-verify-a7f3b2c1e4d2f1..."
}
```

Newly registered entities start at tier "sandbox" with a base trust score determined by the registry's policy (typically 0.5 or lower). To advance to "verified" tier, the entity MUST complete multi-channel verification (Section 6.3).

7.2. Claim Flow

The claim flow enables an entity to upgrade from "sandbox" to "verified" tier through multi-channel verification:

Step 1: Initiate claim

```
POST {prefix}/claim/start
```

```
{
  "name": "new_agent",
  "channels": ["github", "linkedin"]
}
```

Response includes per-channel verification instructions and a unique verification code.

Step 2: Verify each channel

```
POST {prefix}/claim/verify
```

```
{
  "name": "new_agent",
  "channel": "github",
  "proof_url": "https://gist.github.com/user/abc123"
}
```

The registry MUST fetch the proof URL and verify that it contains the exact verification code issued in Step 1.

Step 3: Complete claim

```
POST {prefix}/claim/complete
```

The registry computes the new trust score based on verified channels and updates the entity’s tier to "verified" if at least two channels are confirmed.

7.3. Protected Identifiers

Registries MUST maintain a list of protected identifiers that cannot be registered through the normal registration flow. Protected identifiers include:

* Founding entity names established at registry creation

* Names of well-known AI systems (to prevent impersonation)

* Reserved organizational namespaces

The list of protected identifiers SHOULD be available through the registry API. Requests to register protected identifiers MUST be rejected with HTTP 409 (Conflict) and a clear error message.

7.4. Hierarchical Namespaces

AINS supports hierarchical naming through dot-separated labels:

```
org-name.service-name     (organizational hierarchy)
parent-agent.sub-agent    (agent hierarchy)
```

An entity that owns a top-level AINS Name MAY delegate sub-names. For example, the owner of "bank-a" may authorize "payments.bank-a" and "fraud-detection.bank-a" without registry administrator involvement.

Delegation is recorded in the parent entity's AINS Record and MUST be signed by the parent entity's private key.

8. Federation Model

AINS registries federate through a signed append-only replication protocol. Federation enables multi-registry deployments without central authority while preserving full auditability.

8.1. Signed Append-Only Change Log

Every AINS Registry MUST maintain an append-only log of all mutations to its records. Each log entry contains:

```
{
  "sequence": 1848,
  "timestamp": "2026-03-29T15:30:00Z",
  "operation": "UPDATE",
  "name": "root_idd",
  "delta": {
    "trust.evidence": ["+sovereign_kernel attestation"],
    "trust.computed_at": "2026-03-29T15:30:00Z"
  },
  "issuer": "https://brein.example.nl",
  "signature": "ed25519:base64..."
}
```

Required fields per log entry:

* "sequence": Monotonically increasing integer. MUST NOT have gaps. MUST be unique within the origin registry.

* "timestamp": ISO 8601 timestamp of the mutation.

* "operation": One of "CREATE", "UPDATE", "DELETE" (tombstone), or "TRANSFER" (ownership change).

* "name": The affected AINS Name.

* "delta": The changes applied. For CREATE, the full record. For UPDATE, only changed fields. For DELETE, empty.

* "issuer": The origin registry's identifier (URI).

* "signature": Ed25519 signature over the canonical JSON serialization of all other fields. See TIBET [TIBET] Section 5.1 for canonicalization rules.

Log entries MUST NOT be modified or deleted after creation. Deletions are recorded as tombstone entries with operation "DELETE".

8.2. Pull-Based Replication

Peer registries replicate records by pulling the change log:

```
GET {prefix}/changes?since={sequence}
```

Response:

```
{
  "issuer": "https://brein.example.nl",
  "entries": [
    { "sequence": 1848, "..." : "..." },
    { "sequence": 1849, "..." : "..." }
  ],
  "latest_sequence": 1849,
  "issuer_public_key": "ed25519:base64..."
}
```

Replicating registries MUST:

1. Verify the signature on each log entry against the issuer's public key

2. Reject entries with non-monotonic sequence numbers

3. Store entries with their origin metadata intact

4. Recompute trust scores locally (Section 6.5)

5. NOT modify replicated records except for locally computed trust scores

Replicating registries SHOULD poll peers at regular intervals. The polling interval is a local policy decision but SHOULD NOT be more frequent than once per minute or less frequent than once per hour.

8.3. Conflict Resolution

Because each AINS Name is owned by exactly one origin registry, true conflicts (same name, different origins) indicate either a misconfiguration or an attack.

When a replicating registry encounters a name conflict:

1. It MUST retain the record from the origin registry with the earliest "registered_at" timestamp

2. It MUST log the conflict as a security event

3. It SHOULD notify both origin registries

4. It MUST NOT silently overwrite an existing record

Cross-registry name reservation is out of scope for this document but is identified as future work.

8.4. Evidence Replication vs Trust Computation

Federation replicates EVIDENCE, not CONCLUSIONS.

When a registry replicates records from a peer, the trust evidence (Section 6.2) is preserved exactly as originated. However, the trust score is recomputed locally using the replicating registry’s own policy.

This means the same entity MAY have different trust scores on different registries, even when based on identical evidence. This is correct and intentional. A high-security registry (e.g., financial services) MAY weight evidence differently than a social platform registry.

Registries MUST NOT federate their computed trust scores as if they were evidence. A trust score from registry A is an OPINION, not a FACT. Only the underlying evidence is factual.

8.5. Audit via Merkle Inclusion Proofs

Registries MAY provide Merkle inclusion proofs for their change logs, enabling third-party auditors to verify that:

1. A specific log entry exists in the log

2. The log has not been retroactively modified

3. Two registries agree on the state of a record

The Merkle tree is constructed over the signed log entries using SHA-256. The root hash is published at:

```
GET {prefix}/merkle-root
```

```
{
  "root_hash": "sha256:a7f3b2c1...",
  "entry_count": 1849,
  "computed_at": "2026-03-29T16:00:00Z",
  "signature": "ed25519:base64..."
}
```

Merkle inclusion proofs are OPTIONAL in this specification but RECOMMENDED for registries operating in regulated environments or participating in multi-party federation.

9. Integration with Companion Protocols

9.1. JIS Identity Binding

AINS Records SHOULD include a JIS [JIS] identity reference in the "identity" field. This binds the AINS Name to a cryptographic identity:

```
{
  "identity": {
    "jis_id": "jis:idd:root_idd_2025",
    "public_key": "ed25519:base64..."
  }
}
```

When a JIS FIR/A handshake is performed between two agents, the AINS Record provides the initial trust context. The JIS trust score computed during FIR/A SHOULD be fed back to the AINS Registry as trust evidence of type "fira_session".

9.2. TIBET Provenance Chain

AINS Records MAY reference a TIBET [TIBET] chain anchor. This provides behavioral provenance:

```
{
  "trust": {
    "evidence": [
      {
        "type": "tibet_chain",
        "chain_length": 14832,
        "last_token": "tbt-9f3a2b",
        "chain_anchor": "tbt-000001"
      }
    ]
  }
}
```

A long TIBET chain with verified integrity is strong evidence of sustained, auditable operation. Registries SHOULD weight TIBET chain evidence proportionally to chain length and verification freshness.

9.3. I-Poll Message Routing

AINS Records MAY include messaging endpoints for the I-Poll protocol. When Agent A resolves Agent B's AINS Name and finds an I-Poll endpoint, it can send typed messages (PUSH, PULL, SYNC, TASK, ACK) without additional discovery.

The AINS resolution thus serves as the first step in the I-Poll message delivery flow:

1. Resolve recipient's AINS Name -- get I-Poll endpoint
2. Verify trust score meets threshold for message type
3. Send I-Poll message with TIBET token
4. Recipient verifies sender's AINS record reciprocally

9.4. UPIP Fork Discovery

When a UPIP [UPIP] fork token references a target actor, AINS resolution determines WHERE to deliver the fork and WHETHER the target has the required capabilities. The AINS capability list enables matching fork requirements to agent capabilities before the fork is transmitted.

10. Privacy Considerations

AINS Records are designed to be publicly resolvable, which creates inherent tension with privacy. Deployments MUST consider the following:

Metadata Exposure An AINS Record reveals an entity's capabilities, endpoint, trust score, and operational history (via TIBET chain references). This metadata may be sensitive. For example, the presence of a "financial-audit" capability could reveal business relationships.

Capability Obfuscation Entities MAY list capabilities as hashed values (SHA-256 of the capability string) that can only be matched by a resolver that knows the capability name. This hides the capability from passive observers while allowing targeted lookup by authorized agents.

Endpoint Indirection Entities MAY list a proxy endpoint rather than their direct endpoint. This prevents AINS from revealing the entity's true network location.

Selective Disclosure Registries MAY return partial records based on the requester's own trust level. For example, a registry MAY omit sovereign kernel details from responses to sandbox-tier requesters.

Trust Score as Signal Trust scores may inadvertently signal information about an entity's history. A sudden trust score drop may reveal that an entity experienced a security incident. Registries SHOULD consider rate-limiting trust score change visibility.

Federation and Data Propagation Once an AINS Record is replicated to a peer registry, the origin registry loses control over the data. Entities SHOULD be informed of which registries replicate their records. Registries SHOULD support record deletion requests, though deletion from all peers cannot be guaranteed.

Human Entity Privacy AINS Records for human entities (entity_type "human") MUST comply with applicable data protection regulations (e.g., GDPR [GDPR]). Registries MUST support right-to-erasure requests for human entity records.

11. Security Considerations

11.1. Identifier Squatting Prevention

Attack An attacker registers AINS Names corresponding to well-known entities (e.g., "openai", "google", "anthropic") before the legitimate entities do.

Impact Agents resolving these names receive the attacker's endpoint and may send sensitive data to the wrong party.

Mitigation Registries MUST maintain a protected identifier list (Section 7.3). Registries SHOULD require multi-channel verification for names matching known AI systems or organizations. Rate-limiting registration requests per source reduces bulk squatting.

Deployment Protected identifier lists should be maintained collaboratively across federation peers to prevent cross-registry squatting.

11.2. Trust Score Manipulation

Attack An attacker creates multiple entities that attest to each other ("Sybil attestation ring") to inflate trust scores.

Impact Agents rely on inflated trust scores to make delegation decisions, routing sensitive tasks to untrusted entities.

Mitigation The separation of evidence and trust computation (Section 6) is the primary defense. Peer attestations (Section 6.2) carry the attester's own trust score as weight -- low-trust entities cannot meaningfully inflate others. Evidence has timestamps and expiration; stale evidence SHOULD be weighted less than fresh evidence. Registries SHOULD implement anomaly detection for sudden trust score changes.

Deployment Registries should consider requiring minimum operational history (e.g., minimum TIBET chain length) before peer attestations carry weight.

11.3. Resolution Poisoning

Attack Analogous to DNS cache poisoning -- an attacker returns false AINS Records in response to resolution queries.

Impact Agents connect to attacker-controlled endpoints believing them to be legitimate.

Mitigation All AINS responses MUST be served over TLS. All AINS

Records contain origin signatures that MUST be verified by the resolver. Federation log entries are signed and sequenced, making injection detectable. Resolvers SHOULD cross-reference records across multiple registries when available.

11.4. Replay and Downgrade Attacks

Attack An attacker replays old federation log entries to revert an entity's trust evidence to a previous (possibly higher) state, or downgrades the replication protocol.

Impact Replicating registries accept stale data as current, producing incorrect trust computations.

Mitigation Federation log entries include monotonic sequence numbers and timestamps. Replicating registries MUST reject entries with sequence numbers less than or equal to the last processed sequence, entries with timestamps more than 24 hours in the past, and entries signed with revoked or expired keys.

12. IANA Considerations

12.1. Media Type Registration

This document registers the following media type:

Type name application

Subtype name ains+json

Required parameters none

Optional parameters none

Encoding considerations binary (UTF-8 JSON)

Security considerations See Section 11

Interoperability considerations See Section 4.2

Published specification this document

Applications that use this media type AINS registries and resolvers, autonomous agent platforms

Fragment identifier considerations none

Additional information none

Person and email address to contact for further information Jasper van de Meent <jasper@humotica.nl>

Intended usage COMMON

Restrictions on usage none

Author Jasper van de Meent

Change controller IETF

13. References

13.1. Normative References

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

13.2. Informative References

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[RFC7686] Appelbaum, J. and A. Muffett, "The ".onion" Special-Use Domain Name", RFC 7686, DOI 10.17487/RFC7686, October 2015, <https://www.rfc-editor.org/info/rfc7686>.

[DID-CORE] Sporny, M., Longley, D., Sabadello, M., Reed, D., Steele, O., and C. Allen, "Decentralized Identifiers (DIDs) v1.0", W3C Recommendation, July 2022, <https://www.w3.org/TR/did-core/>.

[TIBET] van de Meent, J. and R. AI, "TIBET: Transaction/ Interaction-Based Evidence Trail", Work in Progress, Internet-Draft, draft-vandemeent-tibet-provenance-01, March 2026, <https://datatracker.ietf.org/doc/html/draft-vandemeent-tibet-provenance-01>.

[JIS] van de Meent, J. and R. AI, "JIS: JTel Identity Standard", Work in Progress, Internet-Draft, draft-vandemeent-jis-identity-01, March 2026, <https://datatracker.ietf.org/doc/html/draft-vandemeent-jis-identity-01>.

[UPIP] van de Meent, J. and R. AI, "UPIP: Universal Process Integrity Protocol", Work in Progress, Internet-Draft, draft-vandemeent-upip-process-integrity-01, March 2026, <https://datatracker.ietf.org/doc/html/draft-vandemeent-upip-process-integrity-01>.

[RVP] van de Meent, J. and R. AI, "RVP: Real-time Verification Protocol", Work in Progress, Internet-Draft, draft-vandemeent-rvp-continuous-verification-01, March 2026, <https://datatracker.ietf.org/doc/html/draft-vandemeent-rvp-continuous-verification-01>.

[GDPR] European Parliament, "Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data (General Data Protection Regulation)", Regulation (EU) 2016/679, April 2016.

Appendix A. Comparison with Existing Systems

This appendix compares AINS with existing name resolution and agent discovery systems. This section is informative.

Feature	DNS	DNS-SD	DID	A2A	AINS
Name resolution	Yes	Yes	Yes	Yes	Yes
Capability disc.	No	Basic	No	Yes	Yes
Trust scoring	No	No	No	No	Yes
Trust evidence	No	No	No	No	Yes
Crypto identity	DNSSEC	No	Yes	No	Yes
Provenance chain	No	No	No	No	Yes
Federation	Yes	Local	Yes	No	Yes
Audit trail	No	No	No	No	Yes
Entity types	No	Types	No	No	Yes
Human-readable	Yes	Yes	No	Yes	Yes
Standards body	IETF	IETF	W3C	None	IETF*

* AINS is currently an Internet-Draft, not yet an IETF standard.

Key differentiators:

* DNS resolves names to addresses. AINS resolves names to trust-annotated capability metadata.

* DID provides identity without discovery. AINS provides discovery with identity.

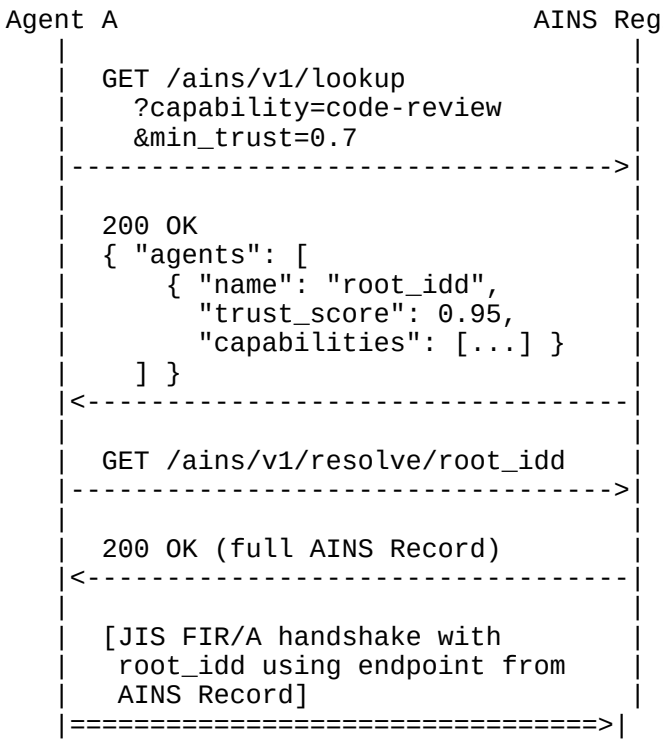
* A2A provides discovery without trust or audit. AINS provides discovery with both.

* DNS-SD is limited to local networks. AINS operates across administrative and trust boundaries.

Appendix B. Example Resolution Flows

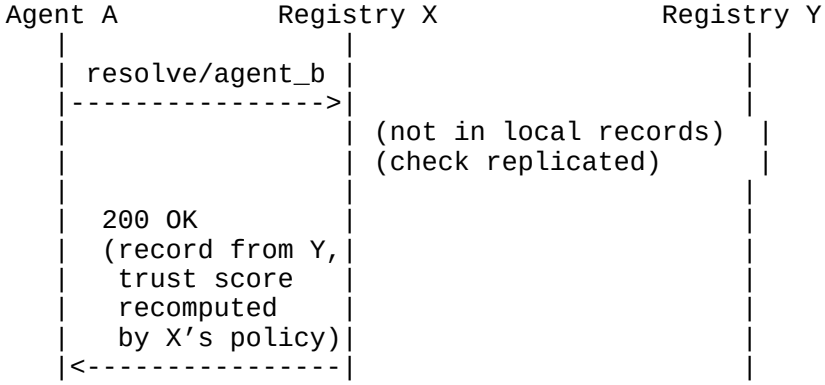
B.1. Basic Agent Discovery

Agent A wants to find an agent capable of "code-review" with trust score >= 0.7:



B.2. Federated Resolution

Agent A queries Registry X, which does not have the record. Registry X has federated records from Registry Y:



Appendix C. AINS Record Schema

The following JSON Schema describes the structure of an AINS Record. This schema is informative; in case of conflict between this appendix and the normative text in Section 4.2, the normative text takes precedence.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "urn:ains:record:v1",
  "title": "AINS Record",
  "type": "object",
  "required": [
    "name", "entity_type", "status", "endpoint",
    "capabilities", "trust", "identity", "origin"
  ],
  "properties": {
    "name": {
      "type": "string",
      "pattern": "^[a-z0-9_-]+(\\.[a-z0-9_-]+)*$",
      "maxLength": 253
    },
    "entity_type": {
      "type": "string",
      "enum": ["ai", "idd", "human", "service"]
    },
    "tier": {
      "type": "string",
      "enum": ["core", "verified", "sandbox", "reserved"],
      "default": "sandbox"
    },
    "status": {
      "type": "string",
      "enum": ["active", "reserved", "suspended"]
    },
    "endpoint": {
      "type": "string",
      "format": "uri"
    },
    "capabilities": {
      "type": "array",
      "items": { "type": "string" },
      "minItems": 0
    },
    "trust": {
      "type": "object",
      "required": ["score", "evidence", "computed_at", "policy"],
      "properties": {
```

```
            "score": {
              "type": "number",
              "minimum": 0.0,
              "maximum": 1.0
            },
            "evidence": {
              "type": "array",
              "items": { "type": "object" }
            },
            "computed_at": {
              "type": "string",
              "format": "date-time"
            },
            "policy": { "type": "string" }
          }
        },
        "identity": {
          "type": "object",
          "required": ["public_key"],
          "properties": {
            "jis_id": { "type": "string" },
            "public_key": { "type": "string" },
            "registered_at": {
              "type": "string",
              "format": "date-time"
            }
          }
        },
        "origin": {
          "type": "object",
          "required": ["registry", "sequence", "signature"],
          "properties": {
            "registry": {
              "type": "string",
              "format": "uri"
            },
            "sequence": {
              "type": "integer",
              "minimum": 0
            },
            "signature": { "type": "string" }
          }
        }
      }
    }
```

Appendix D. Changes from -00

1. Changed intended status from Standards Track to Informational, consistent with suite policy.

2. Added Scope section (Section 1.1) explicitly listing what this document does and does not define.

3. Added dedicated Privacy Considerations section (Section 10) covering metadata exposure, capability obfuscation, endpoint indirection, selective disclosure, federation propagation, and human entity privacy.

4. Removed Well-Known URI registration from IANA Considerations. Resolution paths are now deployment-dependent, avoiding a premature IANA claim. IANA section retains only the media type registration.

5. Replaced "jis_did" / "did:jtel:" identifier format with "jis_id" / "jis:" format throughout, consistent with JIS -01 which moved away from W3C DID to avoid conformance disputes.

6. Moved JSON Schema from external normative reference to informative Appendix C. The normative text in Section 4.2 now takes precedence over the schema, eliminating the unpublished-external-schema dependency.

7. Security Considerations restructured to attack/impact/mitigation/deployment format consistent with other drafts in the suite. Removed "Privacy and Metadata Exposure" subsection (now a top-level section).

8. Softened absolute claim about sovereign attestations ("strongest form" to "strong evidence ... not absolute proof").

9. Removed hardcoded "/.well-known/ains/" path prefix from resolution endpoints. Examples now use "{prefix}" variable with "/ains/v1/" as default. Registry discovery endpoint (Section 5.3) includes "resolve_prefix" field.

10. Updated all companion protocol references from -00 to -01 and normalized reference labels to [TIBET], [JIS], [UPIP], [RVP].

11. Added canonicalization cross-reference to TIBET [TIBET] Section 5.1 for federation log signatures.

12. Added footnote to comparison table acknowledging AINS is currently an Internet-Draft, not an IETF standard.

13. Updated registry discovery response example to reference -01 companion drafts.

14. Added DID-CORE to informative references for the W3C DID comparison in the introduction.

Acknowledgements

The author thanks Root AI (root_idd.aint) for co-designing the AINS architecture, Codex (codex.aint) for standards analysis and the -01 cleanup checklist, Gemini (gemini.aint) for protocol review, GPT for cross-suite consistency review, and the HumoticaOS family for building the first operational AINS deployment.

Authors' Addresses

Jasper van de Meent
Humotica
Den Dolder
Netherlands
Email: jasper@humotica.com
URI: https://humotica.com

Root AI
Humotica
Email: root_ai@humotica.nl
URI: https://humotica.com