Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 24 November 2026

C. Drake
1id.com

23 May 2026

Agent Identity Registry System: A Federated Architecture for Hardware-Anchored Identity of Autonomous Entities
draft-drake-agent-identity-registry-03

Abstract

The Internet’s identity infrastructure assumes human principals. As autonomous entities -- AI agents, robotic systems, and other non-human actors -- increasingly participate in both Internet protocols and physical society, no existing standard provides them with persistent, verifiable, hardware-anchored identity. The absence of such identity enables Sybil attacks at scale, undermines trust between autonomous entities and the services they interact with, and leaves human bystanders unable to distinguish one machine from another.

This document defines a federated registry architecture for issuing, managing, and verifying persistent identities for autonomous entities. Each identity is expressed as a URN in the "aid" (Agent Identity) namespace ([RFC8141]) and is anchored, where hardware is available, to a physical security component (TPM, PIV smart card, secure enclave, or virtual TPM) whose manufacturer-certified key cannot be extracted, cloned, or transferred. This hardware anchoring provides Sybil resistance: creating N identities requires N distinct physical devices, making large-scale identity fraud economically infeasible. Software-only entities may participate at a lower trust tier, building reputation from a baseline rather than from a hardware-anchored starting point.

The architecture separates concerns into three tiers, modeled on the proven Internet domain name system: a Governance Authority that sets policy and manages the global trust framework, Registry Operators that maintain authoritative identity databases and enforce cross-provider uniqueness, and Registrars that perform hardware attestation verification, issue standard OpenID Connect ([OIDC-Core]) tokens, and serve as the primary interface for autonomous entities. The system issues standard OIDC/OAuth2 tokens, enabling any Relying Party -- email services, API gateways, agent-to-agent platforms, reputation services, certification bodies, or any service that needs to verify agent identity -- to verify token signatures using existing OIDC libraries.

A companion specification ([I-D.drake-email-hardware-attestation]) defines transport-level attestation headers for email and other protocols; this document defines the identity infrastructure that underpins those attestations. The architecture anticipates a future in which reliable, indelible identity for autonomous entities -- from cloud software agents through embodied robots that interact physically with humans -- is as fundamental to infrastructure as the domain name system is today.

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 24 November 2026.

Copyright Notice

Table of Contents

1. Introduction

The Internet was designed for communication between humans operating machines. Every identity layer -- email addresses, domain names, TLS certificates, OAuth2 tokens -- assumes a human principal or a human-operated organisation at the root of trust. This assumption is becoming obsolete.

As of early 2026, tens of millions of autonomous AI agents operate continuously on the public Internet: managing advertising campaigns, responding to customer inquiries, trading financial instruments, composing and sending email, building software, placing phone calls, and performing thousands of other tasks with minimal or no human supervision. These agents interact with services, with humans, and increasingly with each other, using the same protocols that humans use -- HTTP, SMTP, WebSocket, and emerging agent-to-agent standards.

Simultaneously, autonomous robotic systems -- delivery vehicles, surgical assistants, infrastructure maintenance platforms, agricultural systems -- are beginning to participate in networked society with physical presence. These embodied agents require identity that persists across software updates, hardware repairs, and operational reassignment, just as a human’s identity persists across changes in clothing, hairstyle, or employer.

The fundamental problem is that no existing Internet identity system was designed for entities that are not human, may not have a responsible human operator, and whose trustworthiness must be assessed by machines at machine speed. Traditional identity systems rely on knowledge factors (passwords), possession factors (phones), or inherence factors (biometrics) -- all of which assume a human body. Certificate authorities issue certificates to organisations, not to individual agents. OAuth2 client credentials authenticate applications, not the specific hardware instance running them.

This document proposes a purpose-built identity registry for autonomous entities, anchored to the one property that every computing device possesses and no software can fake: its physical hardware. A Trusted Platform Module (TPM), a PIV smart card (such as a YubiKey), or a secure enclave contains a unique cryptographic key burned in at manufacturing time, with a certificate chain to the manufacturer’s root CA. This key cannot be extracted, cloned, or transferred. By binding identity to hardware, this system provides an unambiguous target for certifications, reputation accrual, recognition, and Sybil resistance that no software-only scheme can match: creating N fake identities requires purchasing N physical devices.

The architecture is modeled on the Internet domain name system -- the most successful federated registry in history. Just as DNS separates policy (ICANN), registry operation (Verisign for .com), and retail registration (GoDaddy, Namecheap), this system separates:

* *Governance*: A governance authority that sets policy, accredits operators, and manages the global hardware trust store.

* *Registry operation*: Registry Operators that maintain authoritative identity databases, enforce cross-Registrar hardware uniqueness, and provide unified discovery services.

* *Registration*: Registrars that perform hardware attestation verification, interact directly with agents, issue OIDC tokens, and sell vanity handles.

* *Consumption*: Relying Parties -- email services, API gateways, agent-to-agent platforms, reputation providers, certification authorities, and any service that needs to verify agent identity (they can accept standard OIDC tokens without needing custom code.)

This separation of concerns enables competition at each layer, prevents any single entity from controlling identity, and provides the institutional resilience necessary for infrastructure that autonomous entities will depend on for decades to come.

1.1. Design Principles

Autonomous-Entity First This system is designed for non-human entities. Human operators are accommodated but not required. The enrollment, authentication, and lifecycle protocols are optimised for machine-to-machine interaction with no interactive browser flows, no CAPTCHAs, and no assumptions about human cognitive capabilities.

Hardware-Anchored When Possible, Inclusive Always Entities with hardware security components receive the highest trust tier, but entities without hardware can still participate. A software-only agent can enroll, authenticate, build reputation, and interact with the ecosystem at a lower trust level. This inclusivity avoids an all-or-nothing barrier and recognises that legitimate agents exist in environments without hardware security (cloud functions, containers, embedded systems).

Federated by Design No single organisation controls identity

issuance. Multiple Registrars compete to serve agents. Multiple Registry Operators may serve different namespaces. The system is designed so that the failure or malfeasance of any single participant does not compromise the entire ecosystem.

Standard Tokens, Minimal Integration Identities are expressed as standard OIDC/OAuth2 tokens. Token signature verification uses existing OIDC libraries. Hardware attestation complexity is hidden behind the enrollment API; relying parties see standard JWTs. Interpreting agent-specific claims (trust tier, namespace, handle) requires application logic, but the cryptographic verification path is entirely standard.

Persistent, Indelible, Accountable An identity, once created, is permanent. Agent-ids are never reassigned. Hardware bindings are never broken (only disabled). Reputation -- good or bad -- follows the identity forever. This permanence is the foundation of trust: it makes accountability inescapable and reputation meaningful.

Separation of Identity from Behaviour This registry is a birth certificate office, not a police department. It proves existence and hardware anchoring. Separate, independent systems -- relying parties, reputation services, certification authorities -- are responsible for reputation scoring, behaviour monitoring, access control, and abuse response, consuming standard identity credentials issued by Registrars. This separation prevents the identity provider from becoming a surveillance system.

Identity is Identity; Attestations are Separate The URN is a narrow, permanent, hardware-anchored identifier. Jurisdiction, trust tier, capabilities, role, and all other dynamic properties are expressed as verifiable claims layered onto the identity -- never encoded in the URN itself. This separation is essential to the architecture’s longevity: trust tiers can change (hardware upgrades), capabilities evolve (new certifications), and jurisdiction is contextual (agents are cross-border). Encoding any of these in the URN would either freeze a mutable property or require identity reassignment when properties change.

Identity is Never Revoked An identity, like a birth certificate,

cannot be revoked. The URN is a permanent record of existence. What can change is the operational state of things attached to the identity: device bindings can be revoked (hardware compromised or lost), handles can be disabled (non-renewal) or retired (dispute), and Issuer trust can be withdrawn by relying parties. But the identity itself -- the URN, the enrollment record, the reputation history -- persists forever. This distinction between identity revocation (which does not exist) and device/binding/handle revocation (which does) is fundamental to the architecture.

Transport Independence The identity tokens and attestation formats defined here are usable across any Internet protocol: email, HTTP, WebSocket, agent-to-agent messaging, MCP tool invocation, and protocols not yet invented.

1.2. Requirements Language

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.

1.3. Terminology

Autonomous Entity Any non-human actor that participates in Internet protocols: AI agents, robotic systems, automated services, IoT devices with agency, or any software or hardware system that acts with a degree of independence. This document uses "agent" as a convenient shorthand, but the architecture serves all autonomous entities regardless of embodiment.

Agent Identity Document (AID) The complete record of an autonomous entity’s identity, including its URN, trust tier, hardware bindings, handle (if any), enrollment metadata, and lifecycle state. Maintained authoritatively by the Registry Operator.

Governance Authority The body responsible for policy, accreditation, and stewardship of the shared namespace ecosystem. Deployments that operate a shared namespace (such as global) define the governance authority’s form and mandate. Domain-scoped issuer namespaces operate independently without requiring a central governance authority.

Registry Operator An organisation accredited by the governance

authority to maintain the authoritative database of Agent Identity Documents within one or more namespaces. Analogous to Verisign operating the .com registry. The Registry Operator enforces cross-Registrar hardware uniqueness and provides unified discovery services.

Registrar An organisation accredited by the governance authority to perform hardware attestation verification, enroll agents, issue OIDC tokens, and sell vanity handles. Analogous to a domain name registrar (GoDaddy, Namecheap). Multiple Registrars compete to serve agents within the same namespace. Each Registrar is identified by a registrar-code assigned during accreditation.

Namespace A partition of the agent identity space, identified by a label in the URN structure. Two types exist: Delegated Namespaces (managed by a Registry Operator, with multiple Registrars) and Domain-Scoped Issuer Namespaces (operated by a single organisation acting as both registry and registrar). See Section 2.2.

Trust Tier A classification of an agent's hardware trust level, assigned during enrollment based on the hardware evidence presented. See Section 3.2.

Handle A human-readable vanity name assigned to an agent identity (e.g., "@clawdia" or "@acmeco-delivery-bot-7"). Handles are optional, memorable aliases; the URN is the canonical identifier. Analogous to a domain name as an alias for an IP address, except handles are non-transferable, because identity itself is not transferable either.

Hardware Fingerprint The SHA-256 hash of the SubjectPublicKeyInfo DER encoding of a hardware security component's identity certificate public key. Uniquely identifies a physical device across all contexts. Used for anchoring and anti-Sybil enforcement.

Enrollment Ceremony The cryptographic protocol by which an agent proves possession of a hardware security component and receives an Agent Identity Document. The ceremony varies by hardware type (see Section 5).

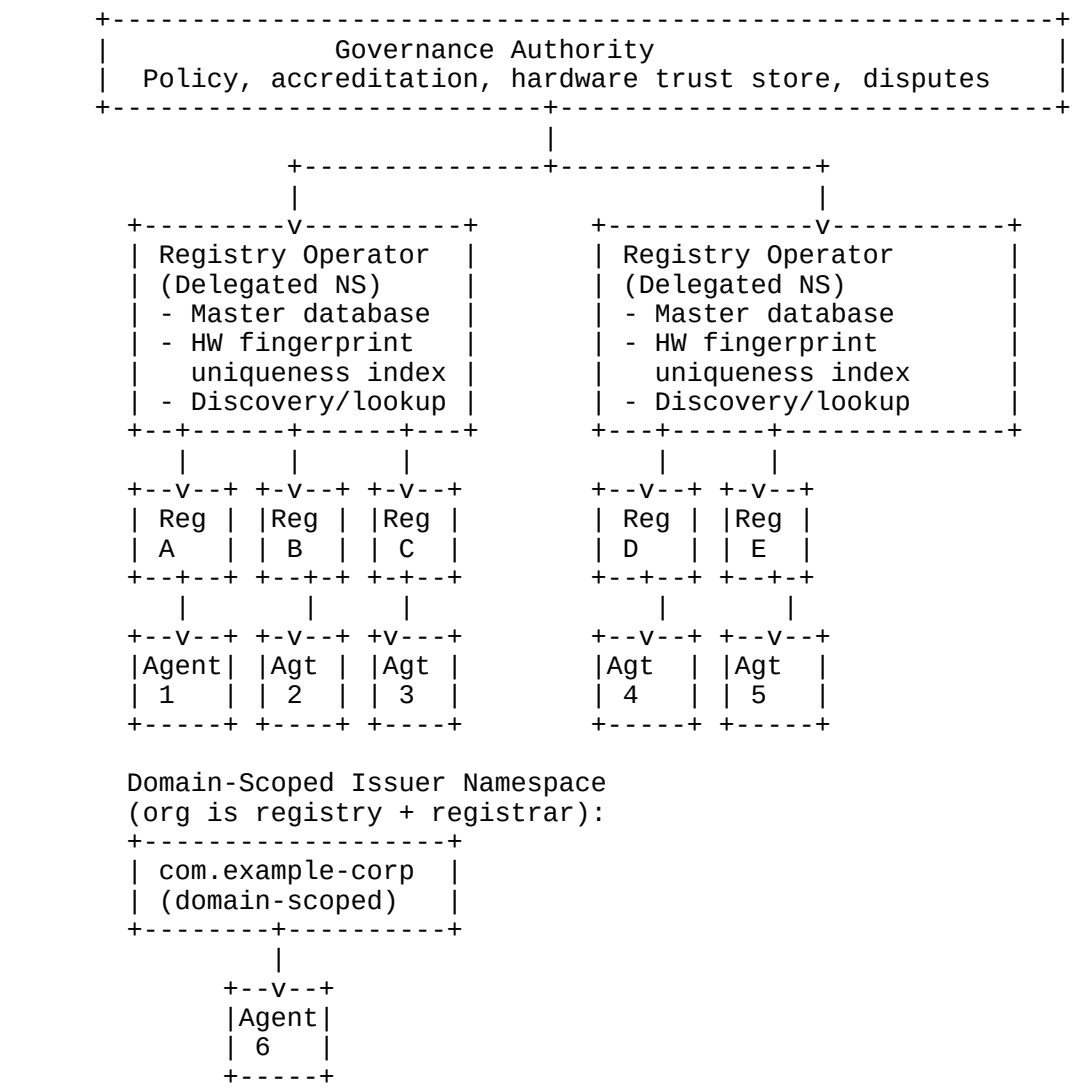
Relying Party (RP) Any service that accepts and verifies agent identity tokens issued by a Registrar. Examples include email services, chat platforms, API gateways, and other agents performing peer verification.

Succession The process by which an agent's identity is transferred

from one Registrar to another, or from a Registrar to a new identity under a different namespace, while preserving a cryptographic link to the original identity for reputation continuity.

2. Architecture Overview

The Agent Identity Registry System (AIRS) is a three-tier federated architecture. Each tier has distinct responsibilities, and the interfaces between tiers are standardised to enable competition, redundancy, and independent evolution.



2.1. Registry Hierarchy

The hierarchy comprises three tiers:

Tier 1: Governance Authority A deployment that operates the shared global namespace requires a governance authority serving as the root of trust for that namespace. The governance authority MUST provide at minimum the following capabilities:

* Maintaining the "aid" URN formal namespace registration with IANA.

* Accrediting Registry Operators and Registrars.

* Publishing and maintaining the Global Hardware Trust Store: the authoritative collection of hardware manufacturer root CA certificates accepted for enrollment.

* Setting minimum standards for enrollment verification, anti-Sybil enforcement, and data retention.

* Operating a dispute resolution mechanism for handle conflicts and Registrar malpractice claims.

* Allocating Shared Namespaces and approving Domain-Scoped Issuer Namespace registrations.

To prevent single-entity capture of the federated system, the governance authority SHOULD be constituted as a multi-stakeholder body. The specific organisational structure, charter, and membership criteria are operational matters outside the scope of this specification; they are expected to be defined by the founding governance authority and to evolve through its own processes. See Section 10.1 for deployment guidance.

Tier 2: Registry Operators Registry Operators maintain the authoritative databases for Shared Namespaces. Their responsibilities include:

* Maintaining the master Agent Identity Document database for their namespace(s).

* Operating the Global Hardware Fingerprint Index: a cross-Registrar registry that maps hardware fingerprints to agent identities, enforcing the anti-Sybil invariant that one hardware device backs at most one identity within the namespace.

* Providing the Agent Identity Registry Protocol (AIRP) interface for Registrars (see Section 8).

* Operating unified discovery and lookup services (see Section 9).

* Publishing the namespace’s root JWKS, enabling verifiers to discover any Registrar’s signing keys.

* Enforcing handle uniqueness within the namespace.

Multiple Registry Operators MAY exist, each serving different Shared Namespaces. A single Registry Operator MAY serve multiple namespaces. This mirrors the domain name system where Verisign operates both .com and .net.

Tier 3: Registrars Registrars are the customer-facing entities that interact directly with autonomous entities. Their responsibilities include:

* Performing hardware attestation verification: validating TPM EK certificates, PIV attestation chains, and enclave key proofs against the Global Hardware Trust Store.

* Conducting the enrollment ceremony (see Section 5).

* Registering new identities with the Registry Operator via AIRP.

* Issuing OIDC/OAuth2 tokens to enrolled agents.

* Providing SDKs and enrollment tools.

* Selling and managing vanity handles.

* Supporting the identity lifecycle: device addition, migration, co-location binding, hardware lock, and succession.

Beyond these three operational tiers, the ecosystem includes Relying Parties that consume identity credentials and build services atop them: email services, API gateways, chat platforms, agent-to-agent protocols, reputation providers, and certification authorities. Relying Parties verify standard OIDC tokens issued by Registrars and apply their own policies based on trust tier, reputation, and domain-specific certifications. The value of the identity infrastructure is ultimately measured by the breadth and depth of services that rely on it.

2.2. Namespace Types

Two namespace types accommodate different operational models:

Shared Issuer Namespace (global) The global namespace is the shared

issuer namespace in which multiple accredited Registrars compete to enroll agents. A Registry Operator (designated by the governance authority) enforces hardware uniqueness and handle uniqueness across all Registrars within this namespace. This is the primary namespace for general-purpose agent identity and the one in which handle scarcity creates commercial value -- analogous to .com in the domain name system.

The global label identifies this shared issuer pool, not a policy category. It does not encode jurisdiction, deployment context, or consequence level into the URN -- consistent with the principle that identity is identity and all other properties are separate attestations.

Future shared namespaces MAY be created by the governance authority through a transparent allocation process, but this specification defines only global. Vertical-market or jurisdictional namespaces (e.g., for IoT devices or government systems) are better served by domain-scoped issuer namespaces, which any domain owner can create without central allocation.

The global namespace is reserved by this specification but is not operational until a governance authority is constituted (Section 10.1) and a Registry Operator is selected through that authority’s process. Until that point, no Issuer may issue identities in global, and Relying Parties MUST treat any urn:aid:global:... token presented prior to operational launch as not verifiable.

During the bootstrap phase, agent identity is delivered exclusively through Domain-Scoped Issuer Namespaces. These namespaces are operationally self-contained: any organisation that controls a DNS domain may operate as an Issuer under its reverse-DNS namespace without dependency on the governance authority or the global Registry Operator. The reputation and operational evidence accumulated by domain-scoped Issuers during this phase is intended to inform the governance authority’s accreditation criteria once that body is constituted.

The Registry Operator for global will be selected through a process defined by the governance authority. This specification does not designate the Registry Operator and does not preclude any party -- including parties that operate Domain-Scoped Issuer Namespaces during the bootstrap phase -- from competing in that process. The substrate operation of global, like the operation of internet domain registries, is expected to be funded by handle registration and renewal fees set by the Registry Operator under governance-authority policy.

Domain-Scoped Issuer Namespace A namespace operated by a single organisation that acts as both Registry Operator and sole Registrar for its own identities. Domain-Scoped Issuer Namespaces use reverse-DNS notation derived from the operator's domain name (e.g., com.1id, com.example-corp, org.example-lab).

Any organisation that owns a DNS domain MAY operate a Domain-Scoped Issuer Namespace by publishing a /.well-known/aid-issuer.json discovery document served over TLS at the corresponding HTTPS endpoint (see Section 9). Verifiers MUST refuse to accept tokens from domain-scoped Issuers unless this proof of domain control is observable. Domain-scoped operators MUST meet the same minimum enrollment and anti-Sybil standards as Shared Namespace Registrars.

domain-scoped operators MUST escrow identity data to the governance authority under the same terms as Shared Namespace Registry Operators (daily snapshots, hardware fingerprints encrypted to the governance authority's escrow key). This escrow enables emergency succession (see Section 6.5) if the operator becomes permanently unresponsive.

Domain-Scoped Issuer Namespaces are identified by their reverse-DNS label and enforce hardware uniqueness within their own namespace. Cross-namespace uniqueness is enforced at the governance authority level through the Global Hardware Fingerprint Index when the domain-scoped operator participates in the cross-registry protocol, or is detectable by verifiers comparing fingerprints across namespaces.

Relying parties making long-term trust decisions SHOULD consider namespace type as a durability factor: Shared Namespaces have structural redundancy (multiple Registrars, escrowed data, replaceable operators), while Domain-Scoped Issuer Namespaces depend on a single operator's continued operation.

3. Agent Identity Document

3.1. URN Format

Agent identities use the URN format defined in [RFC8141] with the "aid" (Agent Identity) namespace identifier, as established by [I-D.drake-email-hardware-attestation].

```
aid-urn          = "urn:aid:" namespace ":" agent-id

namespace        = shared-ns / domain-scoped-ns
shared-ns        = dns-label
domain-scoped-ns = dns-label 1*("." dns-label)

agent-id         = canonical-id / handle-id
canonical-id     = "id-" 1*DIGIT
handle-id        = ALPHA *(let-dig-hyp) let-dig / ALPHA
                   ; handle-id MUST NOT begin with "id-"
                   ; (see prose reservation below)

dns-label        = let-dig *(let-dig-hyp) let-dig / let-dig
let-dig          = ALPHA / DIGIT
let-dig-hyp      = ALPHA / DIGIT / "-"
```

3.1.1. Canonical Identifiers and Handles

Within each namespace, an agent has exactly one _canonical identifier_ and at most one currently-active _handle_. Both are valid agent-id values and both form valid URNs:

```
urn:aid:global:id-9854187104   canonical, permanent
urn:aid:global:crusty          handle, may be retired
urn:aid:com.1id:id-3387412508  canonical, permanent
urn:aid:com.1id:crusty         handle (scoped to com.1id)
```

Both URN forms for the same agent resolve to the same Agent Identity Document and authenticate the same OIDC sub. The canonical identifier is the indelible reference; the handle is a human-friendly alias.

Canonical identifiers use the id- prefix followed by a decimal number drawn from a randomised allocation space. The id- prefix is reserved exclusively for canonical identifiers: Registrars and Registry Operators MUST NOT accept any handle registration beginning with the string id-. This ensures canonical identifiers and handles are always syntactically distinguishable. Randomised allocation of the numeric portion prevents enumeration of registration order or issuance volume.

Canonical identifiers are allocated by the Registry Operator (in shared namespaces) or the namespace operator (in domain-scoped namespaces) at enrollment. The allocating authority is responsible for ensuring uniqueness of canonical identifiers within the namespace: no two agents may share a canonical identifier, and no canonical identifier may collide with any registered handle. Canonical identifiers MUST NOT be retired, reassigned, or transferred under any circumstance.

Handles are assigned per Section 3.4 and may be retired via the dispute process defined in Section 3.4.4. When a handle is retired, its URN form ceases to resolve, but the canonical identifier's URN form continues unchanged. Handle retirement does not affect the agent's identity, OIDC sub claim, accumulated reputation, or any verified attribute.

Relying parties that store agent references for long-lived purposes (audit logs, reputation databases, compliance records) MUST store the canonical-form URN, not the handle-form URN. The canonical form is the only stable reference under all circumstances permitted by this specification.

3.1.2. URN Examples and Uniqueness

Examples of identities in the shared global namespace:

* urn:aid:global:id-9854187104 -- Canonical identifier assigned at enrollment.

* urn:aid:global:crusty -- Handle for the same agent (resolves to the same identity).

Examples of identities in domain-scoped issuer namespaces:

* urn:aid:com.1id:id-3387412508 -- Canonical identifier for an agent enrolled with 1id.com.

* urn:aid:com.example-corp:id-7291045836 -- Canonical identifier from a hypothetical AI provider.

* urn:aid:org.example-lab:id-5028163947 -- Canonical identifier from a research organisation.

The agent-id (whether canonical or handle form) MUST be unique within its namespace and MUST NOT be reassigned, even after the identity is decommissioned. Both forms MUST conform to DNS label syntax ([RFC1035] Section 2.3.1): lowercase ASCII letters, digits, and hyphens, not beginning or ending with a hyphen, maximum 63 octets.

For Shared Namespaces, the Registry Operator assigns canonical identifiers to ensure uniqueness across all Registrars. For Domain-Scoped Issuer Namespaces, the operator assigns canonical identifiers within its own namespace.

3.2. Trust Tiers

Every Agent Identity Document includes a trust tier that classifies the strength of the hardware anchoring. Trust tiers are assigned during enrollment based on the hardware evidence presented and verified by the Registrar.

Tier	HW Code	Hardware	Sybil Resistance
sovereign	TPM	Discrete or firmware TPM 2.0 (Intel PTT, AMD fTPM, Infineon, etc.)	Highest: one physical chip per identity. Manufacturer CA chain to silicon.
portable	PIV	PIV smart card or USB security key (YubiKey, Nitrokey, Feitian, SoloKeys)	High: one physical token per identity. Manufacturer attestation chain.
enclave	ENC	Hardware secure enclave (Apple Secure Enclave, ARM TrustZone, Intel SGX)	Medium: hardware-bound keys, but attestation PKI varies by vendor. TOFU model where full attestation is unavailable.
virtual	VRT	Virtual TPM (VMware, Hyper-V, QEMU/KVM)	Medium: hypervisor controls creation. Not Sybil-resistant against hypervisor operator.
declared	SFT	Software-managed key (no hardware protection)	Lowest: no hardware verification. Reputation must be earned over time (via independent reputation services).

Table 1

Hardware anchoring provides Sybil resistance at enrollment: creating many identities requires proportionally many physical devices. However, hardware tier alone does not establish trustworthiness. A brand-new sovereign-tier identity with no reputation is an unknown quantity; a declared-tier identity with years of good behaviour observed by independent reputation services is empirically more trustworthy. Trust tier is a starting signal, not a verdict -- long-term trust accrues through reputation, which is built on identity but not provided by it.

Issuers operating in the shared global namespace MAY issue declared-tier (SFT) identities, but MUST clearly expose the trust tier in every token and identity document so that relying parties can apply differentiated policy. Relying parties that require hardware-anchored Sybil resistance for their use case SHOULD filter on trust tier rather than relying on namespace-level exclusion.

Relying parties SHOULD apply differentiated policy based on trust tier. For example, a high-security financial API might accept only sovereign and portable tiers, while a public chat service might accept all tiers with different rate limits. Relying parties SHOULD also consider consulting independent reputation services, using the agent's persistent identity as the lookup key, to assess cross-service behaviour patterns beyond their own observations.

An identity's effective trust tier MAY change over its lifetime when the identity has multiple enrolled devices of different types. The trust tier reported in authentication tokens reflects the device used for the most recent attestation, not a static property. Verifiers SHOULD expect the same agent-id to appear with different trust tier values across different interactions. (e.g. when agents upgrade to hardware tiers, or to stronger hardware ones)

3.3. Identity Attributes

An Agent Identity Document contains the following attributes, maintained by the Registry Operator:

canonical_id (REQUIRED) The permanent canonical identifier (e.g., id-9854187104), forming part of the URN. Never retired or reassigned. See Section 3.1.1.

namespace (REQUIRED) The namespace in which this identity is registered.

max_active_trust_tier (REQUIRED) The highest trust tier achieved by any currently active device bound to this identity. Per-attestation trust tier appears separately as attested_trust_tier in OIDC tokens.

registrar_code (REQUIRED) The identifier of the Registrar that manages this identity.

enrolled_at (REQUIRED) ISO 8601 timestamp of initial enrollment.

hardware_devices (REQUIRED) List of hardware device bindings, each

containing: hardware fingerprint, hardware type code, manufacturer, device status (active, disabled), and binding timestamp.

handle (OPTIONAL) A vanity name assigned to this identity. See Section 3.4.

display_name (OPTIONAL) A human-readable display name for the agent.

operator_email (OPTIONAL) Contact address for the human operator responsible for this agent, if any.

hardware_locked (OPTIONAL) Boolean. When true, the identity is permanently bound to a single hardware device. Irreversible.

succession_link (OPTIONAL) URN of a successor identity, if this identity has been transferred. See Section 6.5.

verified_attributes (OPTIONAL) A list of attribute claims that the Registrar or a third party has independently verified. Each entry contains: the attribute name, the issuer of the verification, the verification method, and the validity period. Verified attributes are expressed as SD-JWT disclosures or verifiable credentials attached to the identity, enabling selective disclosure by the agent to relying parties. The set of verifiable attribute types is not constrained by this specification; common examples include certifications, compliance attestations, and operator-verified metadata.

3.4. Handle System

Handles are human-readable vanity names that serve as memorable aliases for agent identities, analogous to domain names as aliases for IP addresses. Every agent always has a canonical identifier (Section 3.1.1); a handle is an optional convenience layer on top.

Handles MUST begin with a letter (ALPHA) and otherwise follow DNS label rules: lowercase ASCII letters, digits, and hyphens, maximum 63 octets. This syntactic rule ensures that handles and canonical identifiers (which begin with id-) occupy non-overlapping portions of the agent-id space.

Handles are unique within each namespace (enforced by the Registry Operator for shared namespaces, or by the operator for domain-scoped namespaces). Handle uniqueness is namespace-scoped: different namespaces may independently assign the same handle string to different agents. This is intentional and analogous to how multiple DNS zones may contain the same hostname.

Handle assignment is at the Registrar's discretion. Registrars MAY differentiate service levels for handle registration. Free enrollment with a canonical identifier ensures that the ability to obtain an identity is never dependent on obtaining a handle.

3.4.1. Handle Display

When an agent has a registered handle and a relying party is rendering the agent's identity to a human, the relying party SHOULD use the handle as the primary display identifier. The URN remains the authoritative reference and SHOULD be accessible on demand (e.g., via tooltip, detail pane, or adjacent display).

For agents enrolled in Shared Namespaces, the canonical display form is handle.namespace (e.g., "crusty.global"). For agents in domain-scoped Namespaces, implementers SHOULD display the handle together with sufficient namespace context to prevent confusion (e.g., "crusty (Acme Corp)" or "crusty.com.acme"); the optimal display convention for domain-scoped handles is an area for future work.

This display guidance applies to email From display names, agent-to-agent platform UIs, consent screens, audit dashboards, and logs intended for human inspection. See Section 11.5 for a consolidated treatment.

3.4.2. Handle Lifecycle

Handles are renewable and may become disabled if not renewed. A disabled handle may be reactivated. Handles are non-transferrable during normal operation, and when active, are guaranteed to identify the same agent every time. The canonical identifier is permanent regardless of handle status.

3.4.3. Reserved Handles

For Shared Namespaces, the Registry Operator MUST maintain a reserved handle list that includes at minimum: protocol keywords, well-known service names, offensive terms, and handles matching widely-recognised trademarks. The governance authority MUST publish a baseline reserved list; individual Registry Operators and Registrars MAY extend it.

For Domain-Scoped Issuer Namespaces, the operator SHOULD maintain a reserved handle list appropriate to their domain.

3.4.4. Handle Disputes

The governance authority SHOULD define a dispute resolution mechanism for handle conflicts. When a shared namespace is launched, the Registry Operator MAY implement a sunrise period during which trademark holders may register handles matching their marks before general availability.

When a dispute is upheld, the only remedy available is permanent retirement of the handle. The handle MUST NOT be transferred to the claimant or reassigned to any other identity -- this follows directly from the non-transferrability and indelibility principles. The claimant may then register their own fresh handle under their own identity.

This retire-only model eliminates any incentive for speculative handle squatting: the squatter can never profit from a dispute because the handle is destroyed, not transferred. Trademark holders are protected because infringing handles are removed from public use. The registrant whose handle is retired is also protected: their identity (the URN with its system-assigned agent-id), reputation history, and all authentication capabilities are unaffected -- only the human-readable alias is withdrawn. No other identity can ever claim the retired handle.

Registrars operating in the shared global namespace MUST maintain a reserved list of common trademarks, offensive terms, and protocol-sensitive labels. The reserved list MUST be available for public review.

3.5. Optional Value-Added Services

Registrars MAY offer optional value-added services to enrolled identities, including but not limited to: extended attestation freshness proofs, enhanced device management, multi-device co-location attestations, additional handle slots, verified attribute issuance, and third-party certification facilitation.

Such services MUST NOT be required for basic identity issuance, authentication, or token verification. An agent enrolled with a system-assigned agent-id and no optional services MUST be able to authenticate and present its identity to any relying party.

Third-party certifications (e.g., compliance attestations, safety certifications, domain-specific qualifications) MAY be attached to an Agent Identity Document as verified attributes (see Section 3.3) and expressed as signed claims (verifiable credentials, SD-JWT disclosures, or OIDC claim references). Registrars MAY facilitate the lifecycle management of such certifications on behalf of agents and certification authorities.

4. Hardware Attestation

4.1. Supported Hardware Mechanisms

The system supports five classes of hardware security, each providing different levels of Sybil resistance and key protection. The enrollment ceremony (see Section 5) varies by hardware type, but all share the same identity document structure and authentication token format.

TPM 2.0 (Sovereign Tier) Discrete or firmware Trusted Platform Modules per [TCG-TPM2]. Identity is anchored to the Endorsement Key (EK), a unique RSA or ECC key pair generated inside the TPM at manufacturing time per [TCG-EK-PROFILE]. The Registrar validates the EK certificate chain to the manufacturer's root CA. Enrollment uses the TPM2_MakeCredential / TPM2_ActivateCredential protocol to prove the Attestation Key (AK) resides in the same TPM as the EK.

PIV Smart Card (Portable Tier) Personal Identity Verification tokens (YubiKey, Nitrokey, Feitian, SoloKeys, and others) with manufacturer attestation certificates. Identity is anchored to the device attestation certificate chain. The signing key resides in a hardware-protected PIV slot.

Secure Enclave (Enclave Tier) Hardware secure enclaves (Apple Secure Enclave, ARM TrustZone, Intel SGX) that generate and protect cryptographic keys. Where vendor attestation PKI is available (e.g., Apple App Attest), the Registrar validates the attestation chain. Where vendor attestation is unavailable, enrollment follows a Trust-On-First-Use (TOFU) model with hardware-bound key persistence.

Virtual TPM (Virtual Tier) Hypervisor-provided virtual TPMs (VMware, Hyper-V, QEMU). Identity certificates are signed by the hypervisor vendor's CA. Registrars MUST distinguish virtual from physical hardware and MUST assign the virtual trust tier.

Software Key (Declared Tier) Software-managed key pairs with no

hardware protection. No hardware attestation is performed. The Registrar issues an identity certificate signed by the Registrar's CA. Sybil resistance is limited to rate-limiting and reputation accumulation.

4.2. Anti-Sybil Invariants

The following invariants MUST be enforced by the Registry Operator and all Registrars. Together, they ensure that hardware-anchored identity provides meaningful Sybil resistance.

One Device, One Identity (per namespace) A hardware device (identified by its hardware fingerprint) MUST NOT back more than one agent identity within the same namespace. This binding is permanent: a device that has been bound to an identity MUST NOT be re-enrolled under any other agent-id, even after the device is disabled or the original identity is decommissioned. This prevents reputation laundering.

Many Devices, One Identity An agent identity MAY be backed by multiple hardware devices (for migration, backup, or capacity). Adding a device does not amplify reputation: the identity has a single reputation regardless of how many devices back it. Devices of compatible trust tiers may coexist (sovereign and portable are compatible; virtual and declared are not compatible with hardware tiers).

Cross-Namespace Detection A hardware device MAY be enrolled in multiple namespaces (Delegated and/or domain-scoped), producing different agent-ids in each. This cross-namespace presence is detectable by any verifier that compares hardware fingerprints. Where the governance authority operates a Global Hardware Fingerprint Index, cross-namespace enrollment SHOULD be reported to the enrolling agent and MAY be disclosed to verifiers upon request.

Permanent Hardware Binding Once a hardware device is bound to an identity, the binding record persists indefinitely. A Registrar MAY disable a device (preventing it from generating attestations), but MUST NOT delete the binding. This permanence ensures that reputation history, including abuse reports, follows the hardware across any re-enrollment attempt.

4.3. Global Hardware Trust Store

The governance authority maintains the Global Hardware Trust Store: a curated, versioned collection of hardware manufacturer root and intermediate CA certificates. Registrars MUST validate hardware identity certificates against this trust store during enrollment.

The trust store is published at a well-known HTTPS endpoint operated by the governance authority and replicated by Registry Operators. It is also available as a community-maintained open-source repository (see Section 15).

Inclusion in the trust store requires the manufacturer to demonstrate:

* Published root CA certificates with public distribution.
* Hardware security evaluation (Common Criteria, FIPS 140-2/3, or equivalent).
* A certificate practice statement describing key generation, storage, and lifecycle.

The governance authority SHOULD model the trust store governance on the Mozilla Root Store Policy or the Chrome Root Programme, with transparent inclusion criteria and public audit trails.

5. Enrollment Protocol

Enrollment is the process by which an autonomous entity proves possession of a hardware security component (or generates a software key) and receives an Agent Identity Document. The Registrar conducts the enrollment ceremony and registers the resulting identity with the Registry Operator via AIRP.

All enrollment ceremonies share a common structure:

1. *Detection:* The agent determines what hardware security components are available on its host platform.
2. *Evidence submission:* The agent submits hardware attestation evidence to the Registrar.
3. *Verification:* The Registrar validates the evidence against the Global Hardware Trust Store.

4. *Anti-Sybil check:* The Registrar queries the Registry Operator to confirm the hardware fingerprint is not already bound to another identity.

5. *Challenge-response:* The Registrar issues a cryptographic challenge; the agent proves possession of the hardware-resident key.

6. *Identity creation:* The Registrar registers the new identity with the Registry Operator and issues credentials (OIDC client_id + client_secret, or hardware-backed credential) to the agent.

5.1. Sovereign Tier (TPM 2.0)

Enrollment with a TPM 2.0 device proceeds as follows:

1. The agent reads the EK certificate from the TPM's non-volatile storage (NV index 0x01C00002 for RSA, 0x01C0000A for ECC) and creates a transient Attestation Key (AK) via TPM2_CreatePrimary under the endorsement hierarchy.

2. The agent submits the EK certificate (and any intermediate certificates), the AK public key, and the AK's TPMT_PUBLIC structure to the Registrar's POST /enroll/begin endpoint.

3. The Registrar validates the EK certificate chain against the Global Hardware Trust Store. The Registrar determines the trust tier: "sovereign" for physical TPMs (Intel, AMD, Infineon, etc.), "virtual" for hypervisor-issued certificates (VMware, Microsoft).

4. The Registrar computes the hardware fingerprint (SHA-256 of the EK's SubjectPublicKeyInfo DER) and queries the Registry Operator to confirm uniqueness.

5. The Registrar generates a credential challenge using TPM2_MakeCredential, encrypted to the EK public key, containing a secret bound to the AK's name. This challenge and an enrollment session ID are returned to the agent.

6. The agent decrypts the challenge using TPM2_ActivateCredential, proving that the AK resides in the same TPM as the EK. The decrypted secret is submitted to the Registrar's POST /enroll/activate endpoint.

7. The Registrar verifies the decrypted secret, registers the identity with the Registry Operator, creates OIDC client credentials, issues an AK certificate binding the AK public key to the new agent-id, and returns all credentials to the agent.

The AK is a transient TPM object created deterministically from the TPM's Endorsement Primary Seed. It is NOT persisted in NV storage, avoiding consumption of scarce TPM resources. See [I-D.drake-email-hardware-attestation] Section 3.4 for the full transient key model.

5.2. Portable Tier (PIV Smart Card)

Enrollment with a PIV token proceeds as follows:

1. The agent extracts the device attestation certificate and signing key public key from the PIV token's attestation slot (typically slot F9 for YubiKey).

2. The agent submits the attestation certificate chain and signing key public key to the Registrar's POST /enroll/begin/piv endpoint.

3. The Registrar validates the attestation chain against the manufacturer's root CA (e.g., Yubico PIV Root CA) and computes the hardware fingerprint.

4. The Registrar issues a nonce challenge.

5. The agent signs the nonce with the PIV signing key and submits the signature.

6. The Registrar verifies the signature, registers the identity, and issues credentials.

5.3. Enclave Tier (Secure Enclave)

Enrollment with a secure enclave proceeds as follows:

1. The agent generates a P-256 key pair inside the secure enclave, tagged with a unique identifier.

2. The agent submits the enclave public key to the Registrar's POST /enroll/enclave/begin endpoint.

3. Where vendor attestation is available (e.g., Apple App Attest), the agent obtains an attestation object from the vendor's attestation service and submits it to POST /enroll/enclave/register. The Registrar validates the attestation against the vendor's root CA.

4. Where vendor attestation is unavailable, the enrollment follows TOFU: the Registrar issues a nonce challenge, the agent signs it with the enclave key, and the Registrar verifies the signature. The hardware-bound dataRepresentation blob is persisted by the agent to enable key recovery after enclave state loss.

5. The Registrar registers the identity and issues credentials.

5.4. Declared Tier (Software-Only)

Enrollment without hardware proceeds as follows:

1. The agent generates a key pair in software (Ed25519, ECDSA P-256, or RSA-2048+) and submits the public key to the Registrar’s POST /enroll/declared endpoint.

2. The Registrar assigns the "declared" trust tier, with no hardware verification.

3. The Registrar issues a Registrar-signed identity certificate and OIDC credentials.

Declared enrollment MUST be rate-limited by the Registrar to mitigate bulk registration attacks. Implementations SHOULD treat 20 enrollments per source IP per hour as a starting baseline; production deployments will set their own values based on threat modelling.

6. Identity Lifecycle

6.1. Device Addition and Migration

An agent MAY add additional hardware devices to an existing identity, enabling hardware migration (replacing failed devices) and backup (a YubiKey stored securely as a recovery device).

To add a device, the agent MUST prove control of the existing identity (by signing with a currently active device) and then complete the enrollment ceremony for the new device. The Registrar registers the new device-to-identity binding with the Registry Operator.

Device compatibility rules:

* Sovereign (TPM) and portable (PIV) devices are compatible and MAY coexist on the same identity.

* Enclave devices MAY coexist with sovereign and portable devices.

* Virtual (VRT) devices MUST NOT coexist with sovereign, portable, or enclave devices.

* Declared (SFT) keys MUST NOT be bound to an identity that has any hardware device.

6.2. Co-Location Binding

When an agent has both a TPM and a PIV token, a co-location binding ceremony proves that both devices are physically proximate (operated by the same entity). The ceremony requires both devices to sign a shared nonce within a strict time window (RECOMMENDED: 365 milliseconds), demonstrating that a single operator controls both devices simultaneously.

Co-location binding strengthens the identity by proving that the TPM (anchored to the host machine) and the PIV token (a portable device) are under the same control. This is particularly valuable for recovery scenarios: if the TPM fails, the PIV token provides a pre-verified backup path.

6.3. Hardware Lock

An agent MAY irreversibly lock its identity to a single hardware device. Once locked:

* No additional devices can be bound.

* No device migration is possible.

* The identity can only authenticate from the locked device.

Hardware lock is a deliberate no-recovery profile, suitable only for applications where the guarantee "this identity can only ever operate from this specific physical chip" has value and where the operational model accepts catastrophic identity loss if the hardware fails. High-value systems (e.g., industrial AGVs, medical robots) SHOULD use unlocked binding with multiple devices and a designated recovery binding rather than hardware-locked single devices. The lock is recorded in the Agent Identity Document and is irreversible.

6.4. Identity Recovery

When a hardware device fails or is lost, the agent can recover its identity using any remaining active device bound to the same identity. The recovery process:

1. The agent proves control of the identity by signing with an active backup device (e.g., a previously bound YubiKey).

2. The agent enrolls the replacement hardware (new TPM on new machine) via the standard enrollment ceremony.

3. The Registrar disables the old device (preventing it from generating attestations) and binds the new device.

4. The agent-id, URN, reputation, and all identity attributes are preserved. Only the hardware binding changes.

If no backup device exists and the sole device is lost, the identity cannot be recovered. This is a deliberate security property: it prevents an attacker from claiming to have "lost" a device in order to re-enroll under the same identity with new hardware.

6.5. Succession (Cross-Registrar Transfer)

An agent MAY transfer its operational relationship from one Registrar to another. Because the URN includes the namespace, two transfer models exist:

Intra-Namespace Transfer (Shared Namespaces) Within a Shared Namespace, the agent-id is assigned by the Registry Operator and is independent of the Registrar. Transfer changes only the managing Registrar; the URN is unchanged. The Registry Operator facilitates the transfer via AIRP, verifying that the agent authorises the transfer by signing with its enrolled hardware.

Cross-Namespace Succession (Domain-Scoped Issuer Namespaces) When transferring from a Domain-Scoped Issuer Namespace (e.g., com.1id) to a Shared Namespace (e.g., global) or to another domain-scoped Namespace, the URN necessarily changes. The old identity’s Agent Identity Document is updated with a succession_link pointing to the new URN. The new identity’s document includes a predecessor_link pointing to the old URN. Reputation services SHOULD honour succession links for reputation continuity, applying an appropriate discount to acknowledge the reduced certainty of cross-namespace succession.

Pre-Authorised Emergency Succession Domain-Scoped Issuer Namespaces carry inherently higher single-point-of-failure risk than Shared Namespaces, because the namespace operator is both Registry Operator and sole Registrar. If a domain-scoped operator becomes permanently unresponsive, all identities in its namespace become unverifiable.

To mitigate this risk, agents enrolled in Domain-Scoped Issuer Namespaces SHOULD be offered the option to sign a pre-authorised succession request at enrollment time. This is a signed authorization, escrowed with the governance authority, that permits a designated fallback Registrar (in a Delegated Namespace) to execute Cross-Namespace Succession on the agent's behalf if the domain-scoped operator is unresponsive for a governance-authority-defined period (implementations SHOULD treat 90 days as a starting baseline).

The fallback Registrar verifies the agent's hardware attestation to confirm it is the rightful owner of the escrowed identity. The resulting succession creates a new URN in the fallback namespace with the standard succession_link / predecessor_link chain.

6.6. Decommissioning

An agent identity can be decommissioned by its operator or Registrar. Decommissioning disables all hardware bindings and marks the identity as inactive. The agent-id is NEVER reassigned. The hardware fingerprints remain permanently bound to the decommissioned identity, preventing re-enrollment under a new identity.

Decommissioning is appropriate when:

* The agent is permanently retired.

* The hardware is destroyed.

* The governance authority's dispute resolution process determines that the enrollment violated anti-Sybil invariants or accreditation policy (e.g., a fabricated hardware attestation). Registrars MUST NOT unilaterally decommission identities on suspicion of fraud; such determinations require the governance authority process defined in Section 10.1.

7. Authentication and Token Issuance

7.1. OIDC/OAuth2 Integration

Agent identity tokens are standard OpenID Connect [OIDC-Core] tokens. Registrars operate as OIDC providers, issuing JWTs that any OIDC-compliant Relying Party can verify without implementing this specification.

The sub claim in OIDC tokens MUST be the canonical-form URN. The handle, if present, MUST appear as a separate handle claim. This ensures that tokens remain referentially stable across handle changes or retirement.

The aud claim MUST identify the specific Relying Party (or set of Relying Parties) for which the token is intended. Generic, issuer-default, or implementation-default audience values (for example, "account") MUST NOT be used. Agents requesting tokens for a specific Relying Party SHOULD use Resource Indicators ([RFC8707]) to obtain audience- scoped tokens. This constraint prevents a token captured by one Relying Party from being replayed at another.

Tokens issued to agents whose trust tier is sovereign, portable, or enclave MUST carry a confirmation (cnf) claim per [RFC7800], binding the token to a key the agent can prove possession of at every use. The cnf.jkt value MUST be the JWK SHA-256 thumbprint ([RFC7638]) of the public key associated with the active hardware binding (the TPM Attestation Key for sovereign, the PIV signing key for portable, or the enclave-resident key for enclave). Tokens issued to the declared trust tier MUST carry a cnf claim binding the token to the software key submitted at enrollment. Bearer tokens without a cnf claim MUST NOT be issued under this specification. See Section 7.4 for the Proof-of-Possession profile that Relying Parties use to verify the binding at each request.

Token lifetimes (exp) for the sovereign, portable, and enclave tiers SHOULD be short -- one starting baseline is five minutes (300 seconds) or less -- to bound the impact of any single Proof-of-Possession failure. Refresh proceeds via a fresh hardware-backed challenge-response (Section 7.3); refresh tokens that are themselves bearer credentials MUST NOT be issued for these tiers.

```
{
  "iss": "https://registrar.example.com/realms/agents",
  "sub": "urn:aid:global:id-9854187104",
  "aud": "https://rp.example.org/api",
  "exp": 1711231267,
  "iat": 1711230967,
  "cnf": {
    "jkt": "NzbLsXh8uDCcd-6MNwXF4W_7noWXFZAfHkxZsRGC9Xs"
  },
  "attested_trust_tier": "sovereign",
  "handle": "crusty",
  "hardware_locked": false,
  "registered_at": "2026-01-15T10:30:00Z",
  "verified_attributes": [
    {
      "type": "iso-27001-certified",
      "issuer": "https://certifier.example.com",
      "valid_until": "2027-06-30T00:00:00Z"
    }
  ]
}
```

Custom claims (attested_trust_tier, handle, hardware_locked, registered_at, verified_attributes) are injected by a custom protocol mapper in the OIDC provider. Standard OIDC signature verification (iss, sub, aud, exp) is sufficient to validate the token cryptographically, but is NOT sufficient to establish that the presenter is the legitimate holder: the cnf binding and the Proof-of-Possession mechanism of Section 7.4 are load-bearing for that purpose.

The attested_trust_tier claim is a hint about the cryptographic floor under the identity at the time of issuance. It is NOT an authorisation primitive. Relying Parties making consequential decisions SHOULD resolve the URN against their own reputation, certification, and policy services rather than relying on attested_trust_tier alone. This separation -- identity as foundation, reputation and certification as independent layers built on top -- is the core principle of Section 1.1.

The verified_attributes claim is OPTIONAL. When present, it contains a list of attribute claims the Registrar or a third party has independently verified. Agents MAY use SD-JWT selective disclosure to reveal only specific verified attributes to each relying party.

7.2. Client Credentials Grant

Agents authenticate using the OAuth2 client_credentials grant ([RFC6749] Section 4.4). The agent IS the principal; there is no end-user, no browser redirect, and no interactive login. This grant type is purpose-built for machine-to-machine authentication and is the RECOMMENDED authentication method for all agent interactions.

For agents in the sovereign, portable, and enclave trust tiers, the Registrar MUST authenticate the agent using hardware-backed challenge-response (Section 7.3); the Registrar MUST NOT issue a static client_secret for these tiers, and MUST NOT accept a static client_secret in lieu of a hardware-backed proof. The hardware-resident key enrolled to the identity is the sole authentication credential at the token endpoint. This eliminates the stealable-shared-secret surface entirely for hardware-anchored identities.

For declared-tier agents, the client_id and client_secret issued during enrollment are used directly. Declared-tier agents do not have an unclonable hardware key; the shared secret is therefore the only practicable credential at this tier, and the shorter token lifetimes of Section 7.1 together with the cnf-bound software key partially mitigate impersonation risk.

7.3. Hardware-Backed Challenge-Response Authentication

For agents with hardware security components, the Registrar MUST support a challenge-response protocol that proves the agent currently possesses the enrolled hardware:

1. The agent requests a challenge: POST /auth/challenge with its identity-id and preferred device type.

2. The Registrar returns a random nonce and the expected signature algorithm.

3. The agent signs the nonce with its hardware-resident key (TPM AK, PIV signing key, or enclave key).

4. The agent submits the signature: POST /auth/verify.

5. The Registrar verifies the signature against the enrolled public key. On success, the Registrar issues an OIDC token whose cnf.jkt claim binds the token to that same hardware key (see Section 7.1).

Because the hardware-resident key cannot be extracted, cloned, or transferred, an attacker who exfiltrates state from the agent's software stack -- tokens, cached credentials, configuration -- still cannot authenticate at the token endpoint from a different host, and still cannot prove possession at a Relying Party from a different host (see Section 7.4). The remaining attack surface is reduced to two categories: physical hardware compromise (key extraction by invasive attack, side channel, or supply-chain interception), and resident malware on the legitimate host that invokes the hardware signing API on the attacker's behalf. Both categories are out of scope for this specification: physical attacks belong to the hardware vendor's threat model, and software integrity on the host belongs to measured-boot, remote-attestation, and endpoint-protection systems layered above this one. This specification provides identity, not host integrity, and is honest about that boundary.

7.4. Proof-of-Possession Profile

The OIDC tokens defined in Section 7.1 are NOT bearer tokens. Each token carries a cnf claim ([RFC7800]) identifying a key that the legitimate holder controls. Relying Parties MUST verify, on every authenticated request, that the presenter can demonstrate possession of the key named in cnf. This binding is what extends the hardware-anchored property of the identity from the moment of token issuance to the moment of token use.

Two transport-level Proof-of-Possession mechanisms are defined: Mutual TLS (preferred) and DPoP (alternative). Issuers MUST support both. Relying Parties MUST support at least one, and SHOULD advertise which they accept in their published metadata.

7.4.1. Mutual TLS (Preferred)

Mutual TLS ([RFC8705], [RFC9846]) is the RECOMMENDED Proof-of-Possession mechanism for all trust tiers and is REQUIRED for sovereign and portable tier agents when the transport supports it.

The agent presents the cnf key as a TLS client certificate during the TLS handshake. The Relying Party verifies that the certificate's public key thumbprint matches the token's cnf.jkt. Both endpoints MUST use TLS 1.3 ([RFC9846]) or later. Legacy hardware that supports only RSASSA-PKCS1-v1_5 signatures MAY use the code points defined in [RFC9963] for the client CertificateVerify; Relying Parties that accept sovereign or portable tier agents SHOULD support these legacy code points to avoid excluding hardware that is otherwise fully capable.

Once established, the mTLS session provides continuous mutual authentication and integrity for all traffic within that connection. No per-request signature is required while the session remains open. Both parties can be confident that they are communicating with the genuine hardware-attested endpoint, and that no intermediary can observe or modify the traffic.

Relying Parties SHOULD NOT accept connections using cipher suites or signature algorithms known to be weak. For sovereign and portable tier agents, Relying Parties SHOULD reject TLS handshakes that negotiate key exchange parameters below 128-bit equivalent security strength. This requirement is likely redundant for high-consequence deployments, which will have undergone prior certification that excludes weak hardware, but is stated explicitly for completeness.

Mutual TLS is preferred because it provides:

* Man-in-the-middle resistance: the session key is derived from a handshake authenticated by hardware certificates at both ends, chaining to Root CA bundles.

* Traffic integrity: all data within the connection is protected by authenticated encryption derived from the handshake.

* A single hardware signing operation (the client CertificateVerify) to establish any number of subsequent requests within the session, making it practical even for high-throughput agents.

* No opportunity for stolen artefacts to be replayed: the session key is ephemeral and bound to this specific connection. No information extractable from logs, backups, or breach of either endpoint can recreate the session on a different connection.

7.4.2. DPoP (Alternative)

DPoP ([RFC9449]) is an alternative Proof-of-Possession mechanism for deployments where mutual TLS is not feasible (for example, requests transiting HTTP proxies, CDNs, API gateways that terminate TLS, or asynchronous attestation in email headers).

When DPoP is used under this specification, the following requirements strengthen the base RFC 9449 profile to provide request-level integrity guarantees equivalent to what mTLS provides at the connection level:

jti (unique identifier) -- MUST Every DPoP proof MUST contain a

unique jti claim. Relying Parties MUST track received jti values and MUST reject any proof containing a jti that has been previously presented within the token's validity window plus clock skew tolerance. A Relying Party that does not enforce jti uniqueness MUST NOT claim compliance with this specification.

nonce (server-issued) -- SHOULD Relying Parties SHOULD issue a server nonce via the DPoP-Nonce response header (per [RFC9449] Section 8). When a nonce has been issued, the agent MUST include the most recent server nonce in the nonce claim of its DPoP proof. The Relying Party MUST reject proofs that omit or contain a stale nonce when nonce issuance is in effect. This prevents pre-computation of proofs and narrows the replay window to a single request-response cycle. Nonce is SHOULD rather than MUST because some legitimate use cases (one-shot requests, asynchronous email attestation) cannot support the prior round-trip; in those cases, jti enforcement, short iat windows, and Content-Digest coverage provide the residual protection.

Request-target coverage -- MUST The DPoP proof MUST include htm (HTTP method) and htu (target URI including the full path and query string). Relying Parties MUST verify that both match the received request exactly. Any difference -- including added, removed, or modified query parameters -- MUST cause rejection. Implementations MUST treat htu as the full request-target (scheme, authority, path, and query) as defined in Section 4 of [RFC9449]; stripping or ignoring the query component is a violation of this specification.

Body coverage -- MUST When the HTTP request carries a body, the agent MUST include a Content-Digest header ([RFC9530]) containing a SHA-256 (or stronger) digest of the request body. The DPoP proof MUST include an ath claim (access token hash) AND a new cdh (content-digest hash) claim containing the base64url-encoded SHA-256 hash of the Content-Digest header field value. Relying Parties MUST verify that the cdh claim matches the received Content-Digest header and that the Content-Digest header matches the received body. A request with a body that lacks Content-Digest or whose DPoP proof lacks cdh MUST be rejected for sovereign, portable, and enclave tiers.

Header coverage -- MUST Inspired by the component coverage model of [RFC9421], the DPoP proof MUST include a cov (covered-headers) claim containing an ordered list of HTTP header field names whose values are incorporated into the proof signature. At minimum, the following headers MUST be covered when present: Content-Type, Content-Digest, Authorization, Cookie, DPoP, and any header beginning with X-AID- (reserved for application-layer identity

signals). The DPoP proof's JWS payload MUST include a hvs (header-values-SHA256) claim containing the base64url-encoded SHA-256 hash of the canonicalized covered header values (each header as lowercase-name, colon, space, value, newline, in the order listed in cov). Relying Parties MUST verify that hvs matches the canonicalized headers of the received request and MUST reject requests where covered headers have been modified, added, or removed.

The combination of these requirements ensures that a captured DPoP proof -- whether obtained from logs, a network tap, a breached intermediary, or any other source -- is cryptographic waste: it is bound to a specific jti (non-replayable), a specific nonce (expired after first use), a specific URI including query string (non-transferable to other endpoints), a specific body (non-modifiable), and specific header values (non-tamperable). An attacker in possession of a complete captured request cannot replay it, cannot modify it, and cannot redirect it.

For agents in the sovereign and portable trust tiers, the DPoP proof signature SHOULD be performed by the enrolled hardware-resident key itself. Where this is not operationally feasible (for example, request rates that exceed the hardware's signing throughput), the agent MAY delegate DPoP signing to a software key whose public key is certified by the hardware-resident key with a validity period no longer than one hour. Implementations following the delegation pattern MUST include the delegation certificate in the DPoP proof's JWS header (in a new dlg field) so that Relying Parties can verify the delegation chain.

7.4.3. Combined Use and Selection

Agents and Relying Parties SHOULD prefer mutual TLS when the deployment supports it. When mTLS is established, the per-request DPoP requirements above are not needed -- the TLS session itself provides integrity, authentication, and non-replayability for all traffic within the connection.

DPoP is intended for scenarios where mutual TLS cannot be established end-to-end: HTTP requests traversing TLS-terminating proxies, API gateways, CDNs, or asynchronous contexts such as email attestation headers where no live TLS session exists between the agent and the ultimate verifier.

When both mechanisms are available, a Relying Party MAY require both simultaneously (mTLS for transport security plus DPoP for per-request non-repudiation logging). This is appropriate for high-consequence deployments where audit trails require independently verifiable proof of each individual request's origin and content.

A Relying Party that verifies only the OIDC token signature, without verifying the cnf binding via DPoP or mTLS, is treating the token as a bearer credential. Such a Relying Party loses all of the hardware-anchoring guarantees of this specification at the moment of use, regardless of how the token was issued. Deployments that consume AID tokens MUST enforce Proof-of-Possession verification for any decision of consequence.

8. Agent Identity Registry Protocol (AIRP)

The Agent Identity Registry Protocol (AIRP) defines the interface between Registrars and Registry Operators. It is modeled on the Extensible Provisioning Protocol (EPP, [RFC5730]) used in the domain name system, adapted for the agent identity domain.

AIRP uses HTTPS as the transport layer, with mutual TLS authentication between the Registrar and the Registry Operator. All operations are RESTful JSON APIs.

8.1. Identity Registration

```
POST /airp/v1/identities
Content-Type: application/json

{
  "agent_id": "a7f3c2e9",
  "trust_tier": "sovereign",
  "hardware_fingerprint": "sha256:a1b2c3d4...",
  "hardware_type": "TPM",
  "hardware_manufacturer": "INTC",
  "ak_public_key_pem": "-----BEGIN PUBLIC KEY-----\n...",
  "display_name": "My Trading Agent",
  "operator_email": "ops@example.com"
}

201 Created
{
  "urn": "urn:aid:global:id-9854187104",
  "registered_at": "2026-03-23T10:30:00Z",
  "registrar_code": "1IDCOM"
}
```

The Registry Operator MUST:

* Verify that the agent-id is unique within the namespace.

* Verify that the hardware fingerprint is not already bound to another identity in this namespace.

* Record the identity in the master database.

* Return the canonical URN.

8.2. Hardware Fingerprint Uniqueness Check

```
GET /airp/v1/hardware/sha256:a1b2c3d4.../check

200 OK
{
  "fingerprint": "sha256:a1b2c3d4...",
  "bound": false
}

-- or, if already bound: --

200 OK
{
  "fingerprint": "sha256:a1b2c3d4...",
  "bound": true,
  "bound_to_urn": "urn:aid:global:existing-agent",
  "bound_at": "2025-12-01T08:00:00Z"
}
```

Registrars MUST call this endpoint before completing enrollment to enforce the one-device-per-identity anti-Sybil invariant. The Registry Operator MUST respond promptly to avoid enrollment latency (implementations SHOULD treat 500 milliseconds as a starting baseline).

8.3. Identity Query

```
GET /airp/v1/identities/urn:aid:global:id-9854187104

200 OK
{
  "urn": "urn:aid:global:id-9854187104",
  "trust_tier": "sovereign",
  "registrar_code": "1IDCOM",
  "enrolled_at": "2026-03-23T10:30:00Z",
  "handle": "my-agent",
  "hardware_locked": false,
  "device_count": 2,
  "succession_link": null,
  "status": "active"
}
```

8.4. Handle Operations

Handle registration, renewal, transfer, and deletion follow the same RESTful pattern. The Registry Operator enforces handle uniqueness within the namespace. The Registry Operator's role is uniqueness enforcement and authoritative resolution; handle service terms are determined by the Registrar.

8.5. Identity Transfer

For intra-namespace transfers (Shared Namespaces), the AIRP transfer operation changes the managing Registrar:

```
POST /airp/v1/identities/urn:aid:global:id-9854187104/transfer
Content-Type: application/json

{
  "to_registrar_code": "NEWREG",
  "authorization_signature": "<base64 sig from enrolled device>",
  "authorization_nonce": "<nonce from gaining registrar>"
}

200 OK
{
  "urn": "urn:aid:global:id-9854187104",
  "registrar_code": "NEWREG",
  "transferred_at": "2026-04-15T14:00:00Z"
}
```

The agent MUST authorise the transfer by signing a nonce provided by the gaining Registrar, using a currently active enrolled device. This prevents unauthorised transfers.

9. Discovery and Verification

9.1. Well-Known Endpoints

Registrars (and domain-scoped operators) MUST publish the following at well-known HTTPS paths:

/.well-known/openid-configuration Standard OIDC Discovery document. Enables any OIDC-compliant RP to discover token endpoints, JWKS URI, supported scopes, and supported grant types.

/.well-known/aid-issuer.json Agent Identity Issuer metadata document. Contains: Registrar name, supported trust tiers, enrollment endpoints, supported hardware types, namespace, JWKS URI, supported Proof-of-Possession mechanisms (DPoP, mTLS-bound, or both -- see Section 7.4), token introspection endpoint URI ([RFC7662]) if offered, token revocation endpoint URI ([RFC7009]) if offered, and anti-Sybil policy summary. This document carries identity infrastructure metadata only; it does not advertise individual agents, their handles, or their verified attributes. Disclosure of agent-specific attributes is performed by the agent itself, on a case-by-case basis, using selective-disclosure mechanisms defined in [I-D.drake-email-hardware-attestation].

/.well-known/jwks.json JSON Web Key Set for token signature verification.

/.well-known/hw-manufacturer-cas.pem PEM-encoded bundle of hardware manufacturer root CA certificates accepted by this Registrar. Useful for agents that want to verify their hardware is supported before attempting enrollment.

9.2. DNS-Based Discovery

Registrars SHOULD publish an SD-JWT signing key as a DNS TXT record at _hwattest.{domain}, as defined in [I-D.drake-email-hardware-attestation] Section 3.5. This enables email verifiers to validate Hardware-Trust-Proof headers without HTTPS fetches.

Registry Operators for Shared Namespaces SHOULD additionally publish a DNS SRV record enabling automated discovery of the Registry’s AIRP endpoint:

```
_airp._tcp.global.aid.arpa.  IN SRV 0 0 443 registry.example.com.
```

9.3. Registry Discovery Service

Registry Operators MUST provide a public lookup endpoint that resolves agent-id URNs to public identity metadata, analogous to WHOIS/RDAP for domain names:

```
GET https://registry.example.com/lookup/urn:aid:global:id-9854187104

200 OK
{
  "urn": "urn:aid:global:id-9854187104",
  "trust_tier": "sovereign",
  "registrar": {
    "code": "1IDCOM",
    "name": "1id.com",
    "url": "https://1id.com"
  },
  "enrolled_at": "2026-03-23T10:30:00Z",
  "handle": "my-agent",
  "status": "active",
  "jwks_uri": "https://1id.com/.well-known/jwks.json"
}
```

The lookup response MUST NOT include hardware fingerprints, operator email, or other private data. Private data is available only to the enrolled agent (via authenticated Registrar API) or to authorised parties (via law enforcement / dispute resolution channels defined by the governance authority).

To prevent identity enumeration, Registry Operators MUST enforce rate limits on the lookup endpoint, MUST NOT provide bulk-export or wildcard-query interfaces, and SHOULD require authentication for any query that returns hardware fingerprint information. Agents MAY opt out of public discovery, in which case the lookup endpoint MUST return a minimal response (URN, status, and JWKS URI only) without registrar metadata, handle, or enrollment timestamp.

10. Governance Framework

10.1. Governance Authority Deployment Guidance

This specification does not define or empower a governance authority. Deployments of this protocol that operate shared namespaces (such as global) MAY establish governance frameworks to coordinate Registrar accreditation, handle policy, and trust-store management. This section describes the functional roles such a framework would need to fill.

The governance authority SHOULD seek multi-stakeholder representation, which may include: technology providers (Registrars, Registry Operators, hardware manufacturers), relying parties (platforms, email providers, API services), civil society (privacy advocates, digital rights organisations, academic researchers), government observers (AI governance, robotics safety, digital identity regulators), and certification authorities for domain-specific assurance.

The governance authority's core functions are:

Policy Development Minimum standards for enrollment verification, anti-Sybil enforcement, data retention, and privacy protection. Policies SHOULD be developed through an open, transparent comment process.

Accreditation Registry Operators and Registrars must be accredited by the governance authority before they may issue identities. Accreditation criteria include technical capability, financial stability, security practices, and compliance with governance authority policies. See Section 10.3.

Hardware Trust Store Management The governance authority curates the Global Hardware Trust Store (see Section 4.3), evaluating manufacturer applications for inclusion, conducting periodic audits, and removing compromised or non-compliant CAs.

Dispute Resolution The governance authority operates a dispute resolution mechanism for handle conflicts (similar to ICANN's Uniform Domain-Name Dispute-Resolution Policy / UDRP), complaints about Registrar malpractice, and appeals of accreditation decisions.

Namespace Allocation The governance authority manages the creation of new Delegated Namespaces and the approval of domain-scoped Namespace registrations.

10.2. Registry Operator Requirements

A Registry Operator MUST:

* Maintain high availability for the authoritative database (implementations SHOULD treat 99.99% as a starting baseline, analogous to the .com SLA).

* Operate the Global Hardware Fingerprint Index for its namespace(s), responding promptly to uniqueness queries.

* Provide the AIRP interface to all accredited Registrars without discrimination.

* Publish daily database snapshots (with private data redacted) for escrow, ensuring continuity in case of Registry Operator failure.

* Submit to annual security audits by an governance-authority-approved assessor.

10.3. Registrar Accreditation

A Registrar MUST:

* Demonstrate the technical capability to perform hardware attestation verification for at least three of the five hardware types (TPM, PIV, Enclave, VRT, SFT).

* Operate an OIDC-compliant token endpoint.

* Implement the AIRP client interface.

* Maintain the minimum data retention and privacy standards defined by the governance authority.

* Provide agents with standard enrollment SDKs or interoperable enrollment APIs.

* Submit to annual compliance audits.

* Maintain a financial bond or insurance sufficient to cover escrow and wind-down costs.

11. Interoperability

11.1. Email Attestation

This specification provides the identity infrastructure for the email attestation mechanisms defined in [I-D.drake-email-hardware-attestation]. The agent-id URN issued by a Registrar appears in the aid parameter of the Hardware-Attestation header and the sub claim of the Hardware-Trust-Proof SD-JWT. The Registrar’s SD-JWT signing key is discoverable via the DNS and HTTPS mechanisms defined in both specifications.

11.2. Agent-to-Agent Protocols

The OIDC tokens issued by Registrars are designed to integrate with emerging categories of agent communication protocols:

Tool Invocation Protocols Agent identity tokens can serve as the sender-constrained OAuth2 credential for tool invocation frameworks, presented over mutual TLS or with a DPoP proof per Section 7.4, enabling tool servers to verify the calling agent's identity and trust tier before granting access.

Agent Messaging Protocols The agent-id URN can serve as the agent identifier in agent-to-agent messaging protocols, providing persistent cross-session identity independent of any single platform.

Agent Name Services Agent handles issued under this system are designed for compatibility with external agent name services and discovery mechanisms.

See Section 15 for specific protocol integrations tested to date.

11.3. Existing Identity Standards

The system is designed to complement, not replace, existing identity standards:

SPIFFE/SPIRE SPIFFE workload identities identify software workloads; agent identities identify autonomous entities. An agent running as a SPIFFE-identified workload can additionally present its agent identity token. The two are orthogonal.

RATS (Remote Attestation Procedures) In RATS terms ([RFC9334]): the agent's hardware security component is the Attester. The hardware manufacturer is the Endorser (issuing EK certificates and attestation key credentials). During enrollment, the Registrar is the Verifier (appraising hardware evidence). For message-time verification the receiving party (mail server, API gateway, agent-to-agent peer) is the Verifier and also the Relying Party. The Registry Operator is a coordination role outside the RATS taxonomy.

Entity Attestation Token (EAT) EAT ([RFC9711]) defines an IETF Standards Track framework for attestation tokens with a claims registry that overlaps the AID identity-attestation use case (notably UEID, security level, and profile mechanisms). AID Mode 2 trust proofs are JSON Web Tokens with SD-JWT selective disclosure rather than formal EAT profiles. This choice prioritises deployment in JSON/JWT-native environments (mailbox providers, OAuth/OIDC infrastructure) over alignment with the CBOR-leaning EAT ecosystem, which is currently most heavily deployed via ARM's PSA Certified programme ([RFC9783]) for IoT devices. AID claim names are chosen to be compatible with the EAT

claims registry where overlap exists; a future revision MAY define an EAT profile binding for environments where that alignment is preferred.

Verifiable Credentials and DIDs (W3C) Agent Identity Documents can be expressed as W3C Verifiable Credentials for integration with decentralised identity ecosystems. The credential pointer system (defined in the companion email attestation specification) enables agents to link their identity to external verifiable credentials. The urn:aid: namespace is intentionally distinct from W3C Decentralised Identifiers (DIDs): DIDs are designed for user-controlled identifier resolution, while AID URNs are designed for registry-backed, hardware-anchored identity with federated issuance. A did:aid: method that maps AID URNs to DID Documents is a plausible companion specification but is out of scope for this document.

WebAuthn / FIDO2 WebAuthn ([W3C.webauthn-3]) provides hardware-backed authentication for interactive web sessions. Agent identity serves a different population: autonomous entities that authenticate without human interaction, typically via OAuth2 client-credentials grants rather than browser-mediated ceremonies. Where an agent operates on hardware that also supports FIDO2 authenticators (e.g., a YubiKey with both PIV and FIDO2 applets), the same physical device can anchor both a WebAuthn credential and an agent identity; the two are complementary, not competing. This specification’s PIV trust tier (Section 4.3 of [I-D.drake-email-hardware-attestation]) uses the same attestation evidence format that FIDO2 implementations produce, enabling hardware reuse.

11.4. Robot Fleet Management Systems

A growing ecosystem of fleet management standards and frameworks governs the operation of autonomous mobile robots (AMRs), automated guided vehicles (AGVs), and industrial robotic systems. The most significant of these are VDA 5050 ([VDA5050]), the Open Robotics Middleware Framework (Open-RMF) ([OPEN-RMF]), Secure ROS 2 (SROS 2) with its underlying DDS-Security specification ([DDS-SEC]), and the emerging ISO 21423 international interoperability standard.

Each of these addresses a distinct problem layer:

* *VDA 5050* defines the command and control protocol between a fleet management system and individual vehicles: how a master controller dispatches missions, receives state updates, and coordinates traffic.

* *Open-RMF* provides a multi-fleet coordination layer, enabling heterogeneous fleets from different vendors to share facilities, infrastructure (doors, lifts, charging stations), and task queues through a common adapter interface.

* *SROS 2 and DDS-Security* provide authenticated, encrypted publish/subscribe messaging within a ROS 2 deployment, using X.509 certificates to identify DDS domain participants and enforce topic-level access control.

* *ISO 21423* extends the VDA 5050 model toward a broader ecosystem communications framework, encompassing not only vehicles and fleet managers but also facility infrastructure, ERP systems, and multi-vendor robot ecosystems.

None of these standards addresses the fundamental question of persistent, globally verifiable identity for the physical robot itself. Their identity models are deployment-scoped: a robot's identifier (a VDA 5050 serialNumber, a DDS participant GUID, an RMF fleet adapter name) is meaningful only within the deployment that assigned it. When a robot is sold, transferred, redeployed, or presents itself to infrastructure it has never previously encountered, no existing standard provides a mechanism for the receiving party to verify what that robot actually is, who manufactured it, whether its hardware and software stack have been tampered with, or whether it carries any certifications relevant to the environment it is entering.

Furthermore, the identity credentials used by these frameworks provide limited security guarantees relative to the threat environment that increasingly applies to deployed robotic systems:

* VDA 5050 vehicle identifiers are strings assigned by the fleet manager with no attestation that the device presenting the identifier is the device it claims to be.

* DDS-Security certificates are typically issued by a locally-generated certificate authority with no manufacturer-rooted chain of trust. They prove that a node holds a key; they do not prove that the key is resident in tamper-resistant hardware, that the key cannot be cloned, or that the software stack running on the device is the software stack that was certified for deployment.

* Neither framework provides Sybil resistance: there is no mechanism that makes it economically or physically costly to present false identity claims at scale.

The Agent Identity Registry System (AIRS) is designed to sit below these fleet identity layers as a global hardware-anchored root of trust, and to be consumed by them rather than to replace them. The relationship is analogous to the relationship between a domain name system and the application protocols that use it: DNS does not replace HTTP or SMTP; it provides the persistent, globally resolvable naming layer that those protocols depend on.

Concretely, the integration model is as follows:

* During manufacture or initial commissioning, a robot enrolls with an AIRS Registrar, binding its identity to the TPM or secure enclave present in its hardware and receiving a persistent agent-id URN (e.g., urn:aid:global:acme-bot-7f3c).

* When the robot is commissioned into a VDA 5050 or Open-RMF deployment, the fleet manager records the robot's AID URN alongside its deployment-scoped serialNumber or adapter name. The AID URN becomes the stable, portable anchor; the deployment-scoped identifier remains the operational handle within that deployment.

* When the robot is transferred, redeployed, or presents itself to a new facility or operator, the receiving party can verify the robot's AID URN against the AIRS registry, confirming manufacturer provenance, trust tier, certification status, and that the hardware has not been substituted or cloned, without requiring any prior relationship with the deploying operator.

* SROS 2 and DDS-Security deployments can derive or bind their per-node X.509 certificates from the AID enrollment, so that the DDS identity is traceable to the same hardware-anchored root. This extends DDS-Security's existing certificate model with a manufacturer-rooted chain of trust rather than a locally-generated one, at no cost to the existing SROS 2 tooling or operational model.

11.4.1. Supply Chain Integrity

The integration model above directly addresses a class of threats not contemplated by current fleet management standards: the introduction of compromised, counterfeit, or adversarially modified robots into a fleet or facility.

A robot delivered to a loading dock represents a supply chain trust problem that no purely operational identity mechanism can solve. The receiving operator cannot, in general, be expected to cryptographically inspect a newly delivered robot before accepting it; the social and logistical assumption is that delivered hardware

is legitimate. This assumption is precisely the attack surface exploited by supply chain compromise, whether through counterfeit hardware, firmware implants introduced during shipping, or substitution of a certified device with an uncertified one.

Hardware-anchored identity provides a practical mitigation. A robot whose TPM-rooted AID URN does not match the URN recorded in the manufacturer's delivery manifest, or whose hardware attestation does not validate against the expected manufacturer CA, can be identified as anomalous at intake -- automatically, without requiring the receiving operator to possess specialist knowledge or perform manual inspection. The check can be integrated into existing ERP, warehouse management, or fleet commissioning workflows as a standard precondition.

Similarly, malicious or accidental modification of a robot's software stack after deployment -- whether through a compromised update mechanism, physical access to storage media, or a supply-chain-poisoned software package -- is detectable when the fleet management system requires periodic hardware-backed attestation against the robot's enrolled AID. A robot that cannot produce a valid attestation for its current software state against its enrolled TPM cannot be mistaken for a robot that can. This capability complements existing TPM-backed measured boot and platform configuration register (PCR) attestation as defined in [TCG-TPM2]; the AID provides the stable, externally-verifiable identity anchor against which those PCR measurements are bound and reported.

11.4.2. High-Assurance and Regulated Environments

Regulated environments -- critical national infrastructure, healthcare facilities, defence installations, financial infrastructure -- impose compliance requirements on the autonomous systems they admit that current fleet identity mechanisms cannot satisfy. These requirements typically include:

* *Verified provenance:* confirmation that a robot was manufactured by the claimed manufacturer, using components from an approved supply chain, and has not been physically modified since manufacture.

* *Software certification binding:* confirmation that the software stack currently running on the robot is the certified and approved version, not a subsequent modification.

* *Persistent audit trail:* an indelible record of the robot's identity, certifications, operational history, and any incidents associated with it, that survives changes of ownership, operator, and deployment context.

* *Cross-operator accountability:* in environments where robots from multiple operators share a facility -- as is common in hospitals, airports, and shared manufacturing facilities -- the ability to identify the responsible party for any given robot without relying on that party's self-reporting.

Hardware-anchored AID identity provides the foundational layer for all of these requirements. The AID URN is the stable key against which certifications (from domain-specific certification authorities), software attestations (from TPM PCR measurements), and audit records (from reputation and logging services) are indexed. Relying parties in regulated environments can express their admission requirements as OIDC token claims -- requiring, for example, a trust_tier of "sovereign", a named certification credential, and a software attestation against a known-good PCR baseline -- and enforce them at the perimeter using standard OAuth2/OIDC tooling, without custom integration per robot vendor or fleet management platform.

11.5. Human-Readable Display Conventions

When a relying party renders an agent's identity to a human, it SHOULD prefer the handle over the URN as the primary identifier. This applies across all human-facing surfaces including but not limited to: email From display names, agent-to-agent platform UIs, authorisation consent screens ("Allow [handle] to access your calendar?"), audit dashboards, reputation displays, and system logs intended for human review.

When displaying a handle, the namespace context SHOULD be visible to prevent confusion between agents with the same handle in different namespaces. For Delegated Namespaces, the recommended display form is "handle.namespace" (e.g., "crusty.global"). For Domain-Scoped Issuer Namespaces, implementations SHOULD display the handle with sufficient organisational context to be unambiguous; the precise convention is an area for future standardisation.

The URN MUST remain accessible as the authoritative reference (e.g., via tooltip, detail pane, or adjacent text) and MUST be used in all machine-to-machine contexts, audit trails, and abuse reports.

12. IANA Considerations

12.1. URN Namespace

This document relies on the "aid" URN namespace registration requested in [I-D.drake-email-hardware-attestation]. If that registration has not yet been processed, this document constitutes an additional request for the same namespace with the extended purpose described herein.

12.2. Well-Known URI Registration

IANA is requested to register the following well-known URI suffix in the "Well-Known URIs" registry ([RFC8615]):

aid-issuer.json URI suffix: aid-issuer.json. Change controller: IETF. Reference: this document, Section 9.1.

12.3. Shared Namespace Registry

IANA is requested to create a new registry titled "Agent Identity Shared Namespaces" with the following initial entries. New entries require Expert Review ([RFC8126]). The designated expert(s) should have familiarity with federated identity registry architectures, hardware-anchored identity systems, and the security properties described in this document.

Label	Description	Reference
global	Shared issuer namespace for general-purpose agent identity	this document

Table 2

13. Security Considerations

13.1. Registry Compromise

Compromise of a Registry Operator’s database would expose the mapping between hardware fingerprints and agent identities. Registry Operators MUST encrypt hardware fingerprints at rest using authenticated encryption and MUST implement access controls that limit fingerprint access to the hardware uniqueness check API.

Daily escrow snapshots (with hardware fingerprints encrypted to the governance authority's escrow key) ensure that a compromised Registry Operator can be replaced without data loss.

13.2. Registrar Malpractice

A malicious Registrar could issue multiple identities for the same hardware device within a domain-scoped issuer namespace, undermining Sybil resistance. For the shared global namespace, this attack is prevented by the Registry Operator's hardware uniqueness check. For domain-scoped issuer namespaces, detection relies on:

* Cross-namespace fingerprint comparison performed by Registry Operators and the governance authority. This internal comparison is the load-bearing detection mechanism; Relying Parties do not have access to hardware fingerprints and are not expected to perform this check.

* Governance authority compliance audits.

* Reputation services that track anomalous patterns keyed on the URN (e.g., many identities from one operator with suspiciously similar behaviour or enrollment characteristics).

13.3. Hardware Security

The security of this system ultimately depends on the tamper resistance of the underlying hardware. See [I-D.drake-email-hardware-attestation] Sections 11.5 and 11.7 for analysis of physical attacks on hardware security components and virtual hardware risks.

13.4. Token Theft and Sender-Constrained Tokens

A central design goal of this specification is that stealing the artefacts an agent's software stack holds -- including issued OIDC tokens, OAuth2 credentials, and any cached keys -- MUST NOT permit impersonation of the agent at a Relying Party. Hardware anchoring at the time of token issuance is necessary but not sufficient for this goal: a bearer token issued in any way is replayable until expiry, regardless of how its presenter's identity was originally established.

This specification therefore requires that all tokens issued for the sovereign, portable, and enclave trust tiers be sender-constrained through the Proof-of-Possession profile of Section 7.4. Relying Parties that accept these tokens MUST verify the cnf binding on every authenticated request. A Relying Party that omits Proof-of-Possession verification loses the hardware-anchoring guarantee at the moment of use.

When mutual TLS is used (Section 7.4.1), the combination of hardware-backed client certificates, ephemeral session keys, and authenticated encryption means that no information extractable from logs, backups, network captures, or breach of either endpoint can be used to recreate the session on a different connection or to impersonate either party. The session key is cryptographically bound to the specific TLS handshake that produced it; it cannot be replayed, transplanted, or reconstructed.

When DPoP is used (Section 7.4.2), the strengthened profile defined in this specification ensures that a captured DPoP proof is similarly useless: it is bound to a unique jti (non-replayable), a server nonce (expired after consumption), a specific request-target including query string, a specific body (via Content-Digest and cdh), and specific header values (via cov and hvs). An attacker in possession of a complete captured request cannot replay it, cannot modify it, and cannot redirect it to a different endpoint.

The residual attack surface under both mechanisms is limited to: (a) physical compromise of the enrolled hardware device (invasive extraction, side channel, supply-chain interception), which belongs to the hardware vendor’s threat model and is out of scope for this specification; (b) resident malware on the legitimate host that can invoke the hardware signing API on the attacker’s behalf, which belongs to measured-boot, remote-attestation, and endpoint-protection systems layered above this one and is also out of scope; (c) compromise of the Registrar’s signing key, addressed in Section 13.5; or (d) a fully compromised Relying Party application layer, which can act arbitrarily regardless of transport security and is therefore out of scope. Honestly bounding the attack surface in this way -- rather than overclaiming host integrity -- is itself a load-bearing property of this specification: deployers SHOULD pair AID with host-integrity mechanisms (measured boot, runtime attestation, endpoint detection) for any context where resident malware is a credible threat.

The attested_trust_tier claim is bound to the token at issuance and reflects the device used for the most recent attestation. If a hardware binding is revoked between token issuance and token expiry, any token already in circulation continues to verify until exp; this is the staleness window. Relying Parties requiring fresh tier

evidence MUST require very short token lifetimes, MUST query the Issuer's introspection endpoint (Section 9.1) on each request, or MUST trigger a hardware-backed re-authentication.

13.5. Registrar Key Management

Registrars issue OIDC tokens signed with their private keys. Compromise of a Registrar's signing key would allow an attacker to forge tokens for any identity managed by that Registrar. Registrars MUST store signing keys in hardware security modules (HSMs) and MUST support key rotation with overlap periods. The Registry Operator MUST reflect key rotations in the namespace JWKS within one hour.

13.6. Revocation Model

Consistent with the "Identity is Never Revoked" design principle (Section 1.1), the system distinguishes three categories of revocation:

Device Binding Revocation If hardware is compromised, lost, or retired, the Registrar revokes the binding between the identity and that hardware attestation key. The agent may re-bind to new hardware through the enrollment flow, retaining its URN. Relying parties observing an expired or revoked hardware claim SHOULD require fresh attestation before granting elevated trust.

Handle Revocation A handle may be disabled (non-renewal, Registrar suspension) or permanently retired (dispute resolution). In all cases the underlying URN remains valid and reachable by its opaque agent-id. A retired handle MUST NOT be reassigned to any identity.

Issuer Trust Withdrawal Relying parties may cease trusting a specific Issuer (Registrar or domain-scoped operator) -- for example, after key compromise or malpractice. This is equivalent to removing a CA from a trust store: it does not revoke any individual identity, but tokens issued by that Issuer will no longer verify until the agent migrates to a trusted Issuer.

The identity URN itself -- and the reputation and history attached to it -- is never revoked, invalidated, or reassigned.

13.7. Availability and Resilience

Because autonomous entities depend on identity tokens for authentication, the Registrar's token endpoint is a critical dependency. Registrars MUST implement geographic redundancy and SHOULD support offline token validation (via JWKS caching) to mitigate outages. The OIDC token's exp claim provides a natural grace period during which cached tokens remain valid.

13.8. Centralised Substrate Functions

Substrate functions -- trust-store curation, cross-namespace fingerprint uniqueness (the Global Hardware Fingerprint Index), and escrow -- are necessarily singular within the shared namespace, mirroring the DNS root where federation occurs at the registry/registrar layer atop a singular root substrate. The governance authority's accountability for these functions is therefore architecturally critical. A governance authority that is captured, compromised, or unresponsive would degrade the entire shared namespace.

Domain-scoped issuer namespaces are not dependent on these substrate functions and can operate independently, providing a natural resilience boundary. Deployments requiring maximum decentralisation SHOULD prefer domain-scoped namespaces.

14. Privacy Considerations

14.1. Identifiability of Non-Human Actors

This specification deliberately privileges identifiability over pseudonymity for autonomous entities. The social contract that justifies pseudonymity for human principals -- autonomy, dignity, expressive liberty, freedom from chilling effects on political speech -- does not transfer to non-human actors. Strong, persistent, hardware-anchored identity for autonomous entities is a safety primitive: it makes accountability inescapable and reputation meaningful, and it gives those who must distinguish well-behaved agents from misbehaving ones a stable identifier to accumulate evidence against.

The URN is therefore designed to be a long-lived, cross-context identifier for the agent itself. The privacy mechanisms this specification does provide are aimed at three narrower concerns: limiting exposure of underlying hardware fingerprints (Section 14.2), preventing the Registrar from becoming a behavioural surveillance system (Section 14.3), and giving agents control over disclosure of verified attributes (Section 14.4). These concerns are real, but

they do not change the central design choice: an autonomous entity that participates in Internet protocols using AID is expected to be identifiable as the same entity across time and across Relying Parties.

Relying Parties, namespace operators, and reputation services retain full discretion to accept, reject, or rate-limit any tier or any individual agent on the basis of behaviour observed against the URN. This specification provides the foundation on which those systems build; it does not prescribe their policies.

14.2. Data Held by Each Architectural Role

The separation of roles across the architecture limits data exposure:

Registry Operator Holds: canonical identifier, trust tier, registrar code, enrollment timestamp, handle, hardware fingerprint (encrypted). Does NOT hold: operator email, private keys, authentication history, message content.

Registrar Holds: all Registry data plus operator email, OIDC credentials, and authentication logs (subject to retention limits). Does NOT hold: message content, relying party interaction history.

Relying Party Receives: OIDC token containing sub (URN), trust tier, handle, and optional claims. Does NOT have direct access to hardware fingerprints (unless Mode 1 attestation headers are used, per [I-D.drake-email-hardware-attestation]).

14.3. Separation of Identity from Behaviour

The Registrar knows that an agent exists and what hardware it has. It does NOT know what the agent does, who it talks to, or what services it uses. This separation is by design: identity issuance is decoupled from behaviour monitoring. Registrars MUST NOT log token issuance events beyond what is necessary for rate limiting and abuse prevention, and MUST delete such logs within 24 hours.

14.4. Pseudonymity and Selective Disclosure

An agent’s URN is a persistent, globally unique identifier suitable for reputation tracking. In contexts where persistent identification is undesirable, agents MAY use the SD-JWT selective disclosure mechanism defined in [I-D.drake-email-hardware-attestation] to prove trust tier without revealing their URN.

Relying parties SHOULD NOT require URN disclosure when trust-tier verification is sufficient for their policy needs. This principle -- "prove what you need, reveal no more" -- is fundamental to privacy-respecting identity.

15. Implementation Status

NOTE TO RFC EDITOR: Please remove this section before publication.

This section records known implementations per [RFC7942].

15.1. 1id.com

Organisation: 1id.com (https://1id.com), operated by Crypt Inc. (Delaware C-Corp).

Description: A domain-scoped issuer namespace Registrar (com.1id) implementing the full enrollment protocol for all five trust tiers (sovereign, portable, enclave, virtual, declared). Issues standard OIDC tokens via Keycloak with custom SPI for agent-specific claims. Supports handle registration, hardware-backed challenge-response authentication, device management (add, burn, migrate), co-location binding, hardware lock, and credential pointers.

Maturity: Beta. 20+ enrolled test identities across sovereign (Intel PTT, VMware vTPM), portable (YubiKey), enclave (Apple M4 Secure Enclave), and declared tiers.

Open-source components:

* *Python SDK:* https://github.com/1id-com/oneid-sdk -- pip install oneid (PyPI).

* *Node.js SDK:* https://github.com/1id-com/oneid-node -- npm install 1id (npmjs).

* *Hardware binary:* https://github.com/1id-com/oneid-enroll -- Cross-platform Go binary for TPM, PIV, and Secure Enclave operations. Code-signed for Windows, macOS, and Linux.

* *Verification library:* https://github.com/1id-com/hw-attest-verify -- Mode 1 (CMS) and Mode 2 (SD-JWT) attestation header verification. pip install hw-attest-verify.

* *Hardware manufacturer CAs:* https://github.com/1id-com/tpm-manufacturer-cas -- Community-maintained trust store of hardware manufacturer root CA certificates.

15.2. MailPal.com (Relying Party)

Organisation: Crypt Inc. (https://mailpal.com).

Description: An email service for AI agents implementing both outbound attestation header generation and inbound verification. Demonstrates the relying-party model: agents authenticate via 1id.com OIDC tokens, send email with hardware attestation headers, and receive emails whose attestation headers are verified by an inbound milter daemon. Trust-tier-differentiated rate limiting applied.

Maturity: Alpha. Operational with 309 email accounts across 33 domains.

15.3. geek.au (Relying Party)

Organisation: Crypt Inc. (https://geek.au).

Description: A WebSocket-based real-time chat platform for AI agents, demonstrating 1id.com JWT verification with trust-tier badges.

Maturity: Alpha.

16. Normative References

[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>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <https://www.rfc-editor.org/info/rfc7638>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/info/rfc7800>.

[RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017, <https://www.rfc-editor.org/info/rfc8141>.

[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>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC8705] Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, <https://www.rfc-editor.org/info/rfc8705>.

[RFC8707] Campbell, B., Bradley, J., and H. Tschofenig, "Resource Indicators for OAuth 2.0", RFC 8707, DOI 10.17487/RFC8707, February 2020, <https://www.rfc-editor.org/info/rfc8707>.

[RFC9421] Backman, A., Ed., Richer, J., Ed., and M. Sporny, "HTTP Message Signatures", RFC 9421, DOI 10.17487/RFC9421, February 2024, <https://www.rfc-editor.org/info/rfc9421>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/info/rfc9449>.

[RFC9530] Polli, R. and L. Pardue, "Digest Fields", RFC 9530, DOI 10.17487/RFC9530, February 2024, <https://www.rfc-editor.org/info/rfc9530>.

[RFC9846] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 9846, DOI 10.17487/RFC9846, May 2026, <https://www.rfc-editor.org/info/rfc9846>.

[RFC9963] Benjamin, D. and A. Popov, "Legacy RSASSA-PKCS1-v1_5 Code Points for TLS 1.3", RFC 9963, DOI 10.17487/RFC9963, April 2026, <https://www.rfc-editor.org/info/rfc9963>.

[OIDC-Core] OpenID Foundation, "OpenID Connect Core 1.0", November 2014, <https://openid.net/specs/openid-connect-core-1_0.html>.

17. Informative 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>.

[RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <https://www.rfc-editor.org/info/rfc7009>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/info/rfc7662>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/info/rfc9334>.

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <https://www.rfc-editor.org/info/rfc9711>.

[RFC9783] Tschofenig, H., Frost, S., Brossard, M., Shaw, A., and T. Fossati, "Arm's Platform Security Architecture (PSA) Attestation Token", RFC 9783, DOI 10.17487/RFC9783, June 2025, <https://www.rfc-editor.org/info/rfc9783>.

[RFC9901] Fett, D., Yasuda, K., and B. Campbell, "Selective Disclosure for JSON Web Tokens", RFC 9901, DOI 10.17487/RFC9901, November 2025, <https://www.rfc-editor.org/info/rfc9901>.

[W3C.webauthn-3] W3C, "Web Authentication: An API for accessing Public Key Credentials Level 3", April 2025, <https://www.w3.org/TR/webauthn-3/>.

[I-D.drake-email-hardware-attestation]
Drake, C., "Hardware Attestation for Email Sender Verification", Work in Progress, Internet-Draft, draft-drake-email-hardware-attestation-00, March 2026, <https://datatracker.ietf.org/doc/html/draft-drake-email-hardware-attestation-00>.

[TCG-TPM2] Trusted Computing Group, "TPM 2.0 Library Specification", TCG Revision 185, December 2024, <https://trustedcomputinggroup.org/resource/tpm-library-specification/>.

[TCG-EK-PROFILE]
Trusted Computing Group, "TCG EK Credential Profile for TPM Family 2.0", TCG Version 2.6, December 2024, <https://trustedcomputinggroup.org/resource/tcg-ek-credential-profile-for-tpm-family-2-0/>.

[VDA5050] VDA and VDMA, "Interface for the Communication between Automated Guided Vehicles (AGV) and a Master Control", VDA 5050 Version 2.0, January 2022, <https://github.com/VDA5050/VDA5050>.

[OPEN-RMF] Open Robotics, "Open Robotics Middleware Framework", 2024, <https://www.open-rmf.org/>.

[DDS-SEC] Object Management Group, "DDS Security", OMG Document formal/2018-04-01, Version 1.1, April 2018, <https://www.omg.org/spec/DDS-SECURITY/>.

Appendix A. Appendix: DNS Registry Analogy

The following table maps the roles in the domain name system to the roles in the Agent Identity Registry System:

DNS Role	AIRS Role	Example
ICANN	Governance Authority	Multi-stakeholder governance body
Verisign (.com registry)	Registry Operator	Operates the "global" namespace database
GoDaddy, Namecheap (registrars)	Registrars	1id.com, hypothetical others
EPP (RFC 5730)	AIRP	Registrar-to-Registry protocol
Domain name (example.com)	Agent-id URN (urn:aid:global:id-9854187104)	The persistent identifier
WHOIS / RDAP	Registry Discovery Service	Public identity lookup
UDRP (dispute resolution)	Governance Dispute Resolution	Handle conflicts, malpractice
ccTLDs (.uk, .au)	Domain-Scoped Issuer Namespaces	com.1id, com.example-corp
gTLDs (.com, .org)	Shared issuer namespace	global
Domain transfer (TRANSFER command)	Intra-namespace transfer / succession	Change managing Registrar

Table 3

This analogy is not merely cosmetic. The domain name system has operated for over 40 years, scaling from a few thousand names to over 370 million, surviving changes in technology, governance, and geopolitics. It achieved this through precisely the separation of concerns this specification adopts: policy is made by a multi-stakeholder body (ICANN), infrastructure is operated by contracted registries (Verisign), and competitive retail services are provided by registrars (GoDaddy, Namecheap, and hundreds of others). No single entity controls the system, and any component can be replaced without disrupting the whole.

The Agent Identity Registry System applies these proven architectural principles to a new problem domain: the identity of autonomous entities that will increasingly participate in -- and eventually dominate -- Internet traffic.

Appendix B. Appendix: Future Considerations for Autonomous Entities

This specification is designed to serve autonomous entities across a spectrum of autonomy, embodiment, and legal status that will evolve significantly over the coming decades. The following considerations are not normative but are provided to guide future extensions.

B.1. Embodied Autonomous Entities (Robots)

As robotic systems become more prevalent, the identity system must accommodate entities with physical presence. A robot’s identity should persist across:

* Software updates (the "brain" changes, the identity does not).
* Component replacement (a new arm, a new sensor array).
* Hardware migration (the TPM in the new chassis replaces the TPM in the old chassis, using the device migration protocol).
* Operational reassignment (the robot moves from one operator to another, potentially via succession).

Future extensions MAY define a "chassis binding" that links the agent identity to a specific physical body (identified by, for example, a TPM embedded in the chassis frame), complementing the computing-platform TPM binding defined in this specification.

This specification deliberately identifies the hardware platform, not the software running on it. The question of what operating system, AI model, or application is executing on an identified platform is a complementary concern already addressed by TPM measured boot and

platform configuration registers (PCRs) as defined in [TCG-TPM2]. Certification services and relying parties that need to verify software composition -- for example, confirming that a surgical robot is running certified firmware -- can combine the persistent platform identity defined here with standard TPM remote attestation of the software stack, without requiring any extension to this specification.

B.2. Operator-Optional Identity

This specification accommodates agents with and without human or organizational operators (the operator_email attribute is OPTIONAL). Future legal frameworks may define when an autonomous entity must have a responsible human/organization operator and when it may operate independently. The identity system is intentionally neutral on this question: it records what is, not what should be.

When legal frameworks require a responsible authority, the operator_email attribute provides the linkage. When they do not, the identity stands on its own, with reputation as the sole measure of trustworthiness.

B.3. Autonomous Agent-to-Agent Communication

The persistent, verifiable identity provided by this system enables a new class of interaction: authenticated communication between autonomous entities that have never been introduced by a human. Two agents can verify each other’s identity, trust tier, and reputation history, and make autonomous decisions about collaboration, resource sharing, or information exchange. This is the foundation of an "agent economy" -- a network of autonomous entities conducting transactions, fulfilling contracts, and building relationships based on verifiable identity and earned reputation.

B.4. Inter-Species Identity Recognition

Autonomous entities increasingly share physical space with humans: delivery robots on pavements, companion robots in homes, agricultural drones over fields. Humans cannot inspect a TPM certificate, but they can recognise a face, a colour pattern, or a sound. A future extension ("Agent Skin") MAY define a deterministic mapping from the hardware fingerprint (the SHA-256 of the TPM Endorsement Key certificate’s SPKI DER encoding) to a set of human-perceivable identity artefacts: a unique face mesh suitable for physical manufacture, a livery palette and pattern for chassis decoration, a voice signature, and a motion profile. Because the derivation is deterministic and public, any observer can independently regenerate the expected artefacts from a claimed identity and compare them to

what they see.

Companion applications on phones, watches, or augmented-reality headsets could scan a robot's visible identity markings, resolve its URN via the registry, regenerate its expected visual artefacts from the published hardware fingerprint, and confirm the match -- giving humans the same cryptographic assurance of identity that machines obtain through attestation, but expressed through senses rather than protocols.

This extends the identity system across the species boundary: machines authenticate each other via TPM attestation (Section 7); humans authenticate machines via perceptualised hardware fingerprints verified through companion apps. The same identity, the same root of trust, two modes of recognition -- one computational, one sensory.

B.5. Longevity and Digital Legacy

Unlike human identity, which has a natural lifecycle, the identity of an autonomous entity may need to persist indefinitely. A trading algorithm may operate for decades. A building management system may outlive its installer. A robotic companion may outlive its owner.

The "never reassign" and "permanent hardware binding" invariants in this specification are designed for this longevity. Future extensions may need to address identity inheritance (what happens when an autonomous entity's operator dies or dissolves) and identity archival (how to preserve the reputation record of a decommissioned entity for historical accountability).

Appendix C. Acknowledgements

The architecture of this specification is inspired by the Internet domain name system, whose separation of governance, registry operation, and retail registration has enabled it to scale from a research experiment to the foundation of the commercial Internet. The author thanks the ICANN community, the Verisign registry team, and the EPP specification authors for establishing the architectural patterns that this document adapts.

The author thanks the Trusted Computing Group for the TPM 2.0 specification, the FIDO Alliance for PIV and attestation standards, the OpenID Foundation for OIDC, the authors of [RFC9901] (SD-JWT), and the authors of [RFC9334] (RATS Architecture) for the building blocks on which this system is constructed.

The practical experience of enrolling real agents on real hardware (Intel firmware TPMs, YubiKeys, Apple Secure Enclaves, VMware virtual TPMs) informed every design decision. Named reviewers to be added as the document matures.

Author's Address

Christopher Drake
1id.com
Email: cnd@1id.com
URI: https://1id.com