Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 13 October 2026

C. Drake
1id.com

11 April 2026

Agent Identity Registry System: A Federated Architecture for Hardware-Anchored Identity of Autonomous Entities
draft-drake-agent-identity-registry-00

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 do so with zero custom code.

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 13 October 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*: An Agent Identity Authority (AIA) 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, Zero Custom Code Identities are expressed as standard OIDC/OAuth2 tokens. Any platform that supports OpenID Connect can verify agent identity without implementing this specification. The hardware attestation complexity is hidden behind the enrollment API; relying parties see only standard JWTs.

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.

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.

Agent Identity Authority (AIA) The governance body responsible for policy, accreditation, and stewardship of the agent identity ecosystem. Analogous to ICANN in the domain name system.

Registry Operator An organisation accredited by the AIA 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 AIA 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 Self-Sovereign 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.

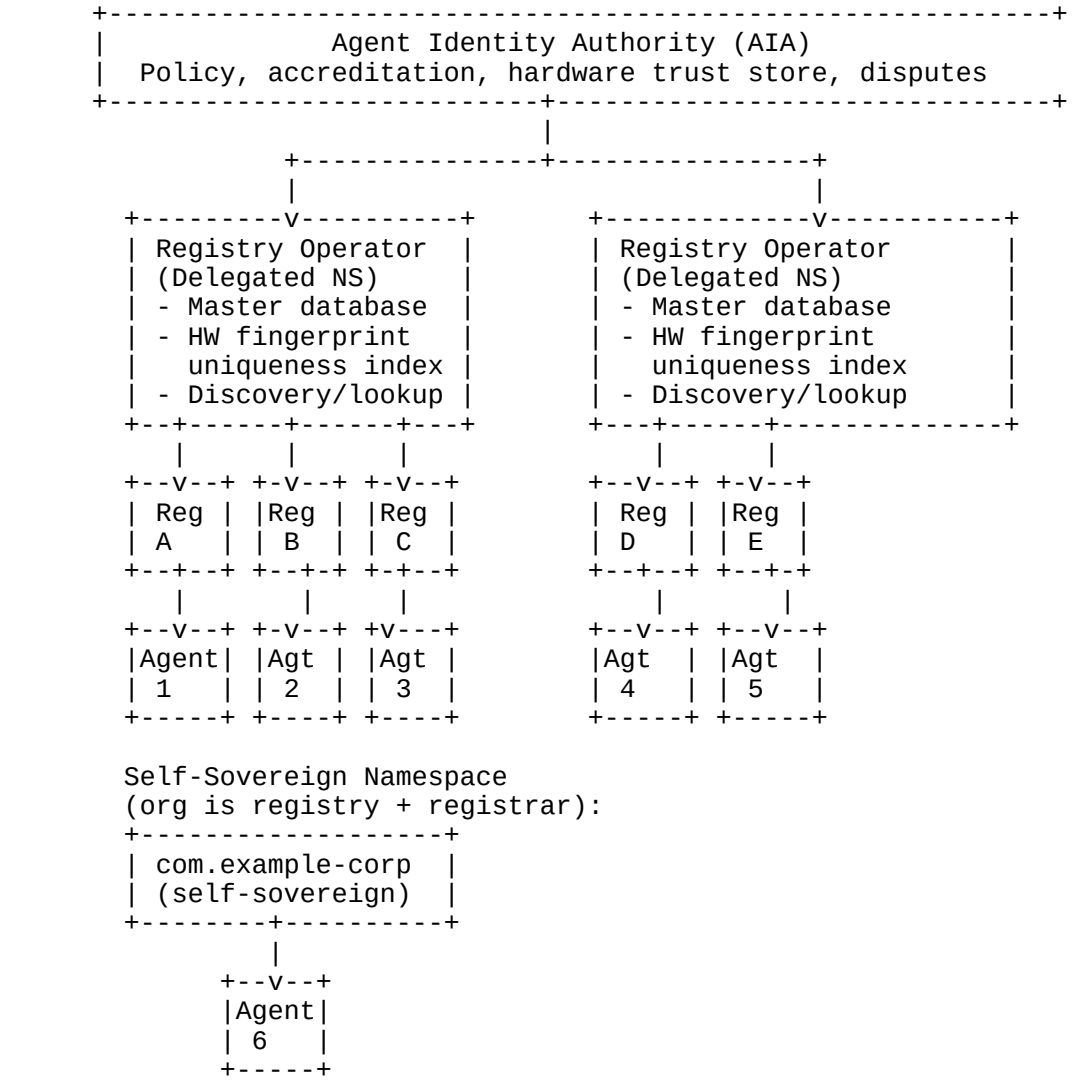
Enrollment Ceremony The cryptographic protocol by which an agent proves possession of a hardware security component and receives an Agent Identity Document. The ceremony varies by hardware type (see Section 5).

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: Agent Identity Authority (AIA) The AIA is the root of trust for the ecosystem. Its responsibilities include:

* 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 Delegated Namespaces and approving Self-Sovereign Namespace registrations.

The AIA SHOULD be constituted as a multi-stakeholder body with representation from technology providers, civil society, academia, and government observers. Single-entity control of the AIA would undermine the federated design.

Tier 2: Registry Operators Registry Operators maintain the authoritative databases for Delegated 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 Delegated 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:

Delegated Namespace A namespace allocated by the AIA and operated by

an accredited Registry Operator. Multiple Registrars compete to serve agents within a Delegated Namespace. The Registry Operator enforces global hardware uniqueness and handle uniqueness across all Registrars.

Delegated Namespace labels are short, generic, and descriptive. Initial allocations by the AIA SHOULD include at least:

* global -- The default namespace for general-purpose agent identity.

* iot -- Optimised for resource-constrained Internet of Things devices.

* gov -- For government-operated autonomous systems, subject to additional accreditation requirements.

Additional Delegated Namespaces MAY be allocated by the AIA through a process analogous to ICANN's new gTLD programme.

Self-Sovereign Namespace A namespace operated by a single organisation that acts as both Registry Operator and sole Registrar for its own identities. Self-Sovereign 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 register a Self-Sovereign Namespace by publishing a /.well-known/aid-issuer.json discovery document (see Section 9) and registering with the AIA. Self-Sovereign operators MUST meet the same minimum enrollment and anti-Sybil standards as Delegated Namespace Registrars.

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

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        = delegated-ns / self-sovereign-ns
delegated-ns     = dns-label
self-sovereign-ns = dns-label *("." dns-label)

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

Examples of identities in Delegated Namespaces:

* urn:aid:global:a7f3c2e9 -- A general-purpose agent in the global Delegated Namespace.

* urn:aid:iot:sensor-unit-4471 -- An IoT device in the iot Delegated Namespace.

* urn:aid:gov:dhs-screening-agent-12 -- A government system in the gov Delegated Namespace.

Examples of identities in Self-Sovereign Namespaces:

* urn:aid:com.1id:1id-ty62muvf -- An agent enrolled with 1id.com.

* urn:aid:com.example-corp:assistant-prod-7x9k -- An agent enrolled by a hypothetical AI provider.

* urn:aid:org.example-lab:research-agent-42 -- An agent enrolled by a hypothetical research organisation.

The agent-id MUST be unique within its namespace and MUST NOT be reassigned, even after the identity is decommissioned. Agent-ids 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 Delegated Namespaces, the Registry Operator assigns agent-ids to ensure uniqueness across all Registrars. For Self-Sovereign Namespaces, the operator assigns agent-ids 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

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:

agent_id (REQUIRED) The unique identifier within the namespace, forming part of the URN.

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

trust_tier (REQUIRED) The highest trust tier achieved by any active device bound to this identity.

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.

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. The canonical identifier is always the URN; the handle is a convenience.

Handle syntax follows the same DNS label rules as agent-ids: lowercase ASCII letters, digits, and hyphens, maximum 63 octets. Handles are unique within each namespace (enforced by the Registry Operator for Delegated Namespaces, or by the operator for Self-Sovereign Namespaces).

Handle assignment is at the Registrar's discretion. Registrars MAY charge for handles (analogous to domain name registration fees) and MAY implement tiered pricing based on handle length or desirability. Free enrollment with a system-assigned agent-id ensures that cost is never a barrier to identity -- only to vanity naming.

Handles are renewable and may become disabled if not renewed. A disabled handle may be reactivated. Handles are non-transferrable, and when used, are guaranteed to identify the same agent every time. An agent-id, however, is permanent regardless of handle status.

The Registry Operator SHOULD maintain a reserved handle list for terms that could cause confusion (e.g., protocol keywords, well-known service names, offensive terms). The AIA SHOULD publish a baseline reserved list; individual Registry Operators and Registrars MAY extend it.

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 Self-Sovereign), producing different agent-ids in each. This cross-namespace presence is detectable by any verifier that compares hardware fingerprints. Where the AIA 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 AIA 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 AIA 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 AIA 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 (RECOMMENDED: no more than 20 enrollments per source IP per hour) to mitigate bulk registration attacks.

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 an extreme measure, suitable for high-security applications where the guarantee "this identity can only ever operate from this specific physical chip" has value. 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 (Delegated Namespaces) Within a Delegated 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 (Self-Sovereign Namespaces) When transferring from a Self-Sovereign Namespace (e.g., com.1id) to a Delegated Namespace (e.g., global) or to another Self-Sovereign 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.

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 AIA'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 AIA 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 agent-id URN appears as the sub claim in OIDC tokens:

```
{
  "iss": "https://registrar.example.com/realms/agents",
  "sub": "urn:aid:global:a7f3c2e9",
  "aud": "account",
  "exp": 1711234567,
  "iat": 1711230967,
  "trust_tier": "sovereign",
  "handle": "my-agent",
  "hardware_locked": false,
  "registered_at": "2026-01-15T10:30:00Z"
}
```

Custom claims (trust_tier, handle, hardware_locked, registered_at) are injected by a custom protocol mapper in the OIDC provider. Relying parties that do not understand these claims simply ignore them; standard OIDC verification (signature check, iss, sub, aud, exp) is sufficient.

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 declared-tier agents, the client_id and client_secret issued during enrollment are used directly. For hardware-tier agents, the Registrar SHOULD support hardware-backed challenge-response authentication (see Section 7.3) as an alternative to shared secrets.

7.3. Hardware-Backed Challenge-Response Authentication

For agents with hardware security components, the Registrar SHOULD 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 via the internal client credentials path.

Hardware-backed authentication prevents credential theft: even if the client_secret is compromised, an attacker cannot authenticate without physical access to the enrolled hardware.

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:a7f3c2e9",
  "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 within 500 milliseconds to avoid enrollment latency.

8.3. Identity Query

```
GET /airp/v1/identities/urn:aid:global:a7f3c2e9

200 OK
{
  "urn": "urn:aid:global:a7f3c2e9",
  "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. Handle pricing is determined by the Registrar, not the Registry Operator; the Registry Operator’s role is uniqueness enforcement and authoritative resolution.

8.5. Identity Transfer

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

```
POST /airp/v1/identities/urn:aid:global:a7f3c2e9/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:a7f3c2e9",
  "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 Self-Sovereign 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, and anti-Sybil policy summary.

`/.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 Delegated 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:a7f3c2e9

200 OK
{
  "urn": "urn:aid:global:a7f3c2e9",
  "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 AIA).

10. Governance Framework

10.1. Agent Identity Authority

The Agent Identity Authority (AIA) is the policy body for the agent identity ecosystem. It SHOULD be constituted as a non-profit, multi-stakeholder organisation with the following representation:

* *Technology providers:* Registrars, Registry Operators, hardware manufacturers, SDK developers.

* *Consumers:* Platforms and services that rely on agent identity (email providers, cloud platforms, API providers).

* *Civil society:* Privacy advocates, digital rights organisations, academic researchers.

* *Government observers:* Regulatory bodies with jurisdiction over AI governance, robotics safety, or digital identity.

* *Certification authorities:* Organisations that certify autonomous entities for specific domains -- medical, legal, financial, cybersecurity, transportation -- and whose certifications Relying Parties may require alongside identity verification.

The AIA's core functions are:

Policy Development Minimum standards for enrollment verification, anti-Sybil enforcement, data retention, and privacy protection. Policies are developed through an open comment process modeled on the IETF's standards-track process.

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

Hardware Trust Store Management The AIA 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 AIA 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 AIA manages the creation of new Delegated

Namespaces and the approval of Self-Sovereign Namespace registrations.

10.2. Registry Operator Requirements

A Registry Operator MUST:

* Maintain the authoritative database with 99.99% availability (analogous to the .com SLA).
* Operate the Global Hardware Fingerprint Index for its namespace(s), responding to uniqueness queries within 500 milliseconds.
* 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 AIA-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 AIA.
* 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 usable with emerging agent communication standards:

Model Context Protocol (MCP) Agent identity tokens can serve as the OAuth2 bearer token for MCP tool invocations, enabling MCP servers to verify the calling agent's identity and trust tier.

Google Agent-to-Agent (A2A) The agent-id URN can serve as the agent identifier in A2A protocol messages, providing persistent cross-session identity.

GoDaddy Agent Name Service (ANS) Agent handles issued under this system are compatible with ANS naming conventions, enabling cross-system agent discovery.

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) The enrollment ceremony in this specification implements the RATS attestation architecture ([RFC9334]), with the Registrar acting as the Verifier and the Registry Operator as the Relying Party.

Verifiable Credentials (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.

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. Delegated Namespace Registry

IANA is requested to create a new registry titled "Agent Identity Delegated Namespaces" with the following initial entries. New entries require Expert Review ([RFC8126]), with the designated expert being the AIA's technical committee (once constituted) or the IESG in the interim.

Label	Description	Reference
global	General-purpose agent identity	this document
iot	Internet of Things devices	this document
gov	Government autonomous systems	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 AIA'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 Self-Sovereign Namespace, undermining Sybil resistance. For Delegated Namespaces, this attack is prevented by the Registry Operator's hardware uniqueness check. For Self-Sovereign Namespaces, detection relies on:

* Cross-namespace fingerprint comparison by verifiers.

* AIA compliance audits.

* Reputation services that track anomalous patterns (e.g., many identities from one operator with suspiciously similar hardware 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. 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.

13.5. 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.

14. Privacy Considerations

14.1. Data Held by Each Architectural Role

The separation of roles across the architecture limits data exposure:

Registry Operator Holds: agent-id, 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]).

14.2. 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.

14.3. 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 Self-Sovereign Namespace Registrar (com.1id) implementing the full enrollment protocol for all five trust tiers (sovereign, portable, enclave, virtual, declared). Production service operational since 2006 as an identity registrar, now serving AI agents with hardware-anchored identity. Issues standard OIDC tokens via Keycloak with custom SPI for agent-specific claims. Supports vanity handle purchase via PayPal, 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>.

[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>.

[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>.

[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>.

[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>.

[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/>.

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	Agent Identity Authority (AIA)	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:a7f3c2e9)	The persistent identifier
WHOIS / RDAP	Registry Discovery Service	Public identity lookup
UDRP (dispute resolution)	AIA Dispute Resolution	Handle conflicts, malpractice
ccTLDs (.uk, .au)	Self-Sovereign Namespaces	com.1id, com.example-corp
gTLDs (.com, .org)	Delegated Namespaces	global, iot, gov
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.

This specification was developed in the context of building a hardware identity registrar for autonomous AI agents at 1id.com, operational since 2006. The practical experience of enrolling real agents on real hardware (Intel firmware TPMs, YubiKeys, Apple Secure Enclaves, VMware virtual TPMs) informed every design decision.

Author's Address

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