ENS as a Naming Layer for AI Agent Identity:

Production Infrastructure for Agent Identification, Authentication, and Authorization

Public Comment on NCCoE Concept Paper:
"Accelerating the Adoption of Software and AI Agent Identity and Authorization"

Prepared by `estmcmxci.eth` on behalf of the ENS Ecosystem
`https://discuss.ens.domains/c/ens-dev/ai/79`

April 2, 2026

Abstract

The Ethereum Name Service (ENS) is an open, permissionless naming and identity system operational on Ethereum since 2017 with over two million registered names. ENS occupies the *naming layer* for AI agent identity—the same architectural position DNS holds in the traditional internet stack. The authorization, authentication, and communication protocols NIST identifies (OAuth 2.0, OIDC, SPIFFE, MCP) operate *above* this layer and can reference ENS identities the same way they reference domain names today. ENS has already deployed production infrastructure for cross-chain agent identity resolution: arbitrary data records (ENSIP-24), agent registry verification (ENSIP-25), and a canonical on-chain blockchain directory (Chain Registry-Resolver). ENS protocol developers are extending this foundation with agent-specific capabilities for routing, classification, and cryptographic verification. This comment responds to NIST's six question areas and five areas of interest with positions grounded in deployed infrastructure and active standards development, and proposes the ENS agent identity stack as a candidate for NCCoE demonstration.

1 Introduction

NIST's concept paper correctly identifies the central challenge: AI agents bring significant productivity gains, but exercising those gains safely requires that agents can be *identified*, *authenticated*, and *authorized* within enterprise architectures [1]. Every protocol stack that solves these problems needs a naming layer—a system that maps human-readable identifiers to machine-readable addresses, metadata, and endpoints. In the traditional internet, DNS provides this layer. For AI agents operating across heterogeneous blockchain and web environments, the Ethereum Name Service (ENS) provides it.

ENS is not a proposal. It is production infrastructure: operational since May 2017, with over two million registered names, supported by all major Ethereum wallets and decentralized applications, and governed through open, public processes [2]. Each ENS name resolves to an owner (an Ethereum account providing cryptographic identity binding), a resolver (a smart contract storing extensible records), and a hierarchy of subnames modeling organizational delegation.

Critically, ENS was already functioning as agentic infrastructure before the standards described in this paper existed. Human-readable names resolved to machine-readable addresses; text records

stored arbitrary metadata; multi-chain address records enabled cross-chain identity resolution; and subname hierarchies modeled organizational delegation—all capabilities that agents require and that ENS provided as general-purpose naming infrastructure. The new standards formalize and extend these existing capabilities specifically for the agent use case.

Production vs. Draft: A Transparent Distinction

The ENS agent identity stack comprises seven layers. This paper distinguishes two tiers throughout:

- [PRODUCTION] — Deployed on Ethereum mainnet, canonized within the ENS protocol, and operational today. Layers 0–3: ENS Naming, ENSIP-24, Chain Registry-Resolver, ENSIP-25.
- [DRAFT] — Under active development by ENS protocol developers, building on the production foundation. Layers 4–6: ENSIP-26, Node Metadata Standard, Agent Identity Profile.

This transparency is deliberate: NIST should know exactly what is deployed today and what is under development.

ENS standards authors have been independently developing agent identity capabilities since early 2026, with a dedicated governance forum,[1] in direct response to the same challenges NIST's concept paper describes.

2 The ENS Agent Identity Stack

The ENS agent identity stack is a layered architecture in which each layer builds on the one below it. Table 1 presents the seven layers with their deployment status.

Table 1: The seven-layer ENS agent identity stack.

Layer	Standard	Status	Function
0	ENS Naming (EIP-137)	[PRODUCTION]	Human-readable, cross-chain agent identifier; owner = Ethereum account
1	ENSIP-24	[PRODUCTION]	Arbitrary binary data records on ENS nodes
2	Chain Registry-Resolver (`on.eth`)	[PRODUCTION]	Canonical on-chain blockchain directory; ERC-7930 reverse resolution
3	ENSIP-25	[PRODUCTION]	Links ENS names to chain-specific agent registries (ERC-8004)
4	ENSIP-26	[DRAFT]	`agent-context` + `agent-endpoint[<protocol>]`
5	Node Metadata Standard (NMS)	[DRAFT]	`class` + `schema` text records; JSON Schema 2020-12
6	Agent Identity Profile (AIP)	[DRAFT]	Signed manifests, version lineage, default-deny verification

2.1 Layer 0: ENS Naming [Production]

ENS (EIP-137) is the naming system for Ethereum [2]. Human-readable names such as `agent.uniswap.eth` map to machine-readable Ethereum addresses, resolver contracts, and extensible records (text, address, contenthash, and data). ENS subnames model organizational hierarchy: `agent.org.eth` is controlled by the owner of `org.eth`, enabling natural delegation patterns.

[1] https://discuss.ens.domains/c/ens-dev/ai/79

This is the naming layer. Every standard described below—ENSIP-24, ENSIP-25, `on.eth`, ENSIP-26, NMS, AIP—is built on ENS records and ENS resolution.

2.2 Layer 1: Arbitrary Data Resolution — ENSIP-24 [Production]

ENSIP-24 adds a new resolver profile for arbitrary binary data storage on ENS nodes [4]:

```
interface IDataResolver {
    function data(bytes32 node, string calldata key)
        external view returns (bytes memory);
}
```

Text records (ENSIP-5) store strings only. Emerging use cases—interoperable addresses, hashed commitments, binary context data—require raw bytes. ENSIP-24 provides this, with an optional `ISupportedDataKeys` interface for key discovery.

Why it matters for agents: ENSIP-24 is the foundation for ENSIP-25's cross-chain registry addressing. Without binary data records, ENS could not store the ERC-7930 interoperable addresses that encode chain-specific registry locations. ENSIP-24 is canonized within the ENS protocol and CC0 licensed.

2.3 Layer 2: Chain Registry-Resolver — `on.eth` [Production]

The Chain Registry-Resolver is a canonical, on-chain smart contract registry for blockchain metadata, deployed on Ethereum mainnet under the `on.eth` namespace [6]. Every registered blockchain receives a human-readable ENS name: `optimism.on.eth`, `base.on.eth`, `arbitrum.on.eth`.

Per-chain metadata includes:

- **ERC-7930 interoperable addresses** (immutable after registration) [7]
- Text records, data records, address records, contenthash
- **Reverse resolution** (ERC-7828): converts interoperable addresses back to human-readable chain labels [8]

The contract implements a two-tier authorization model: the contract owner registers chains; chain admins (e.g., the Optimism Foundation) control their own metadata but cannot affect other chains. Full discoverability is provided via `chainCount()`, `getChainAtIndex()`, `supportedTextKeys()`, and `supportedDataKeys()`.

Why it matters for agents: When ENSIP-25 encodes a registry address as an ERC-7930 interoperable address, `on.eth` resolves that address to a human-readable chain name. An agent client reading `agent-registration[0x0001...][167]` can query `on.eth` to learn the target chain.

2.4 Layer 3: Agent Registry Verification — ENSIP-25 [Production]

ENSIP-25 defines a parameterized text record pattern for verifying the association between an ENS name and an AI agent registered in an on-chain registry such as ERC-8004 [5, 12]:

```
agent-registration[<registry>][<agentId>]
```

where `<registry>` is the ERC-7930 interoperable address of the registry contract and `<agentId>` is the registry-scoped identifier.

Verification flow:

1. Obtain the claimed ENS name, agent ID, and registry address from the registry entry.
2. Construct the parameterized text record key.
3. Resolve the text record on the claimed ENS name.

4. A non-empty value indicates the ENS name owner attests the association.

ENSIP-25 is cross-chain by design: ERC-7930 addresses encode chain identification and contract address into a canonical representation. An agent registered in ERC-8004 on Base can be verified against the same ENS name on Ethereum mainnet. The mechanism leverages existing text records, requiring no resolver upgrades. It is canonized within the ENS protocol and CC0 licensed.

2.5 Layer 4: Routing & Discovery — ENSIP-26 [Draft]

Agents deployed across multiple chains and protocols lack a unified discovery mechanism. ENSIP-26 solves this with two text record types [9]:

- `agent-context`: free-form description of the agent's capabilities, chain coverage, and token support.
- `agent-endpoint[<protocol>]`: per-protocol connection URIs. Supported protocols include MCP, A2A, OASF, and web—with an open keyspace allowing future protocols without standard amendments.

Table 2 shows example records for a hypothetical swap agent.

Table 2: Example ENSIP-26 records for `swap.uniswap.eth`.

Record Key	Value
`agent-context`	`Automated token swap agent. Supports ETH, USDC, WBTC on Ethereum, Optimism, Base, Arbitrum.`
`agent-endpoint[mcp]`	`https://mcp.swap.uniswap.org`
`agent-endpoint[a2a]`	`https://a2a.swap.uniswap.org`
`agent-endpoint[web]`	`https://swap.uniswap.org`

The design rationale favors flat text records over JSON blobs: each record is independently verifiable on L2s via Unruggable Gateways, which is critical for zero-trust architectures.

2.6 Layer 5: Classification & Typed Metadata — NMS [Draft]

ENS nodes currently lack standardized classification. The Node Metadata Standard (NMS) introduces two global text records [10]:

- `class`: a pascal-case classification value (e.g., `Agent`, `Org`, `Treasury`, `Delegate`).
- `schema`: a pointer to a JSON Schema 2020-12 document defining the node's metadata attributes, required fields, format constraints, and descriptions [13].

Clients fetch the schema at resolution time and discover what attributes exist without prior ENSIP knowledge—runtime discoverability following established data modeling patterns (JSON-LD `@type`, RDF `rdf:type`). The `inherit` keyword enables parent policies to propagate to child nodes in the subname hierarchy.

2.7 Layer 6: Cryptographic Verification — AIP [Draft]

Agents change over time; without verifiable versioning, consumers can be silently upgraded to different behavior. The Agent Identity Profile (AIP) solves this with exactly three ENS records [11]:

- `agent-manifest`: CID of a signed manifest on IPFS.
- `agent-latest`: current version string.
- `agent-version-lineage`: how versions map to manifests (e.g., via subnames).

Manifests are JSON documents containing schema identifier, ENS name, version, a `prev` pointer to the previous version, payload (endpoints, capabilities, policy), and a signature by the ENS owner. Verification requires three checks—`ensName` matches the resolved name, `version` matches `agent-latest`, and the signature is valid under the current ENS owner—with a **default-deny** posture: any failed check results in the agent being treated as unverified.

Version lineage via `prev` pointers creates an immutable audit trail; consumers can pin to specific versions, and auto-upgrade is prohibited for pinned consumers.

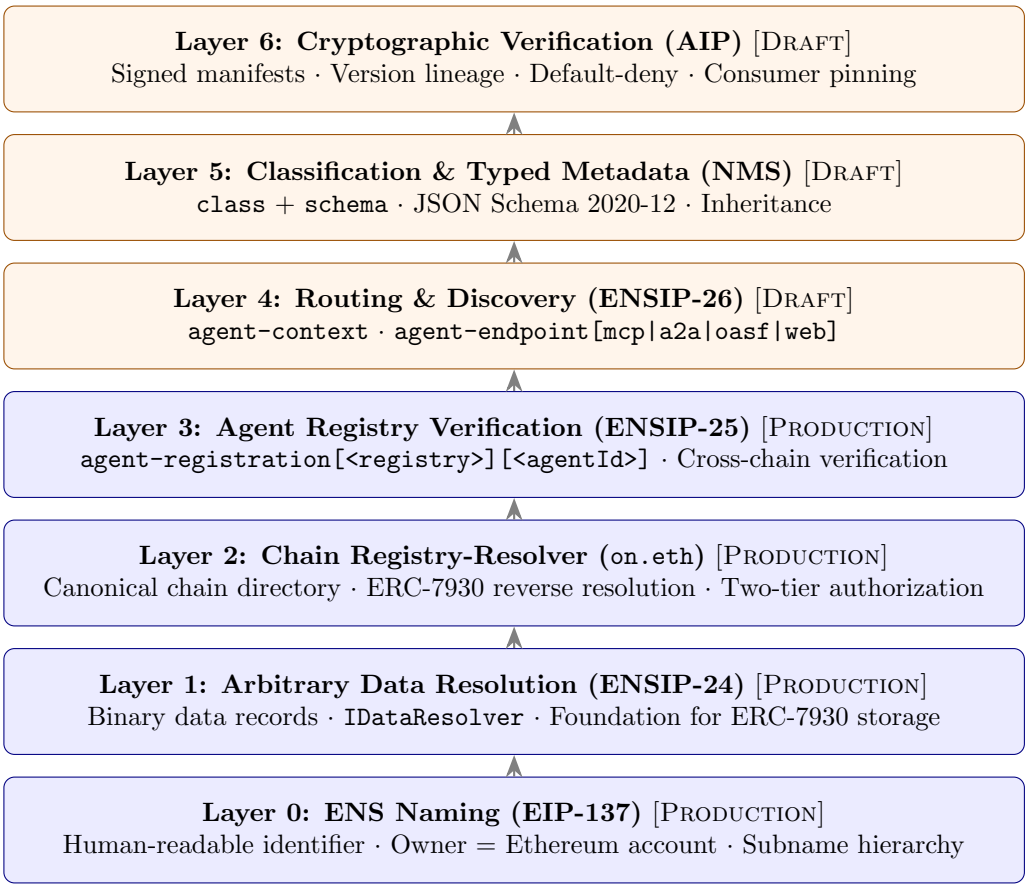

Figure 1: The seven-layer ENS agent identity stack. Blue layers ([Production]) are deployed on Ethereum mainnet. Orange layers ([Draft]) are under active development. Each layer builds on the one below it. Source: `diagrams/unified-ens-agent-identity-stack.md`. **TODO:** Update unified diagram source to reflect the 7-layer model with [Production]/[Draft] tags.

3 Response to NIST Question Areas

3.1 General Questions (§1)

Use cases (1a–1c): AI agents are deployed today across the blockchain ecosystem for governance evaluation, automated trading, security monitoring, and natural language interfaces for identity management. Near-future use cases include agent-to-agent discovery via ENS, cross-chain agent coordination, and automated delegation via subname hierarchies.

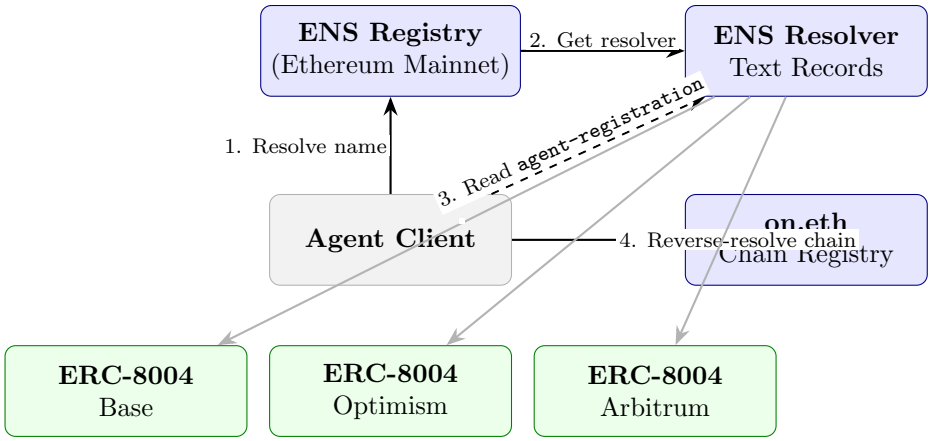

Figure 2: Cross-chain agent identity verification using ENSIP-25 and `on.eth` [PRODUCTION]. An agent client resolves the ENS name, reads parameterized `agent-registration` text records linking to ERC-8004 registries on multiple chains, and uses `on.eth` to reverse-resolve the ERC-7930 chain identifier to a human-readable label. Source: `diagrams/ensip-25-agent-registry-verification.md`, `diagrams/chain-registry-resolver.md`.

MCP support (1f): MCP is a first-class protocol in ENSIP-26's `agent-endpoint[mcp]` [DRAFT]. A2A and OASF are equally supported, and the open keyspace allows new protocols without standard amendments.

How agentic architectures differ (1g): Agents require unified identity across multi-chain, multi-protocol deployments; their behavior changes over time (a versioning problem traditional software identities do not face); and they act autonomously with limited supervision, making authorization more consequential than for human-operated software.

Current and emerging technology (1h–1i): ENS production infrastructure (Layers 0–3) is deployed today. Agent-specific extensions (Layers 4–6) are under active development through open, CC0-licensed governance.

3.2 Identification (§2)

How agents are identified (2a): ENS names provide a single, human-readable, cross-chain identifier for any agent [PRODUCTION]. The seven-layer stack (Table 1) provides progressive identification depth—from a name and owner at Layer 0 through cross-chain registry verification at Layer 3 to cryptographic manifests at Layer 6.

ENSIP-25 [PRODUCTION] already provides cross-chain verification between ENS names and on-chain agent registries using ERC-7930 addressing resolved via `on.eth` [PRODUCTION].

The relationship to NIST's listed protocols is architectural: SPIFFE/SPIRE attests that "this process is running where it claims"; ENS attests that "this agent is named what it claims." They operate at different layers and are complementary in the same way DNS complements HTTP.

Essential metadata (2b): Table 3 summarizes essential agent metadata and its deployment status.

Ephemeral vs. fixed (2c): Identity is fixed: the ENS name and owner persist across all interactions [PRODUCTION]. Metadata is versioned: AIP [DRAFT] provides explicit versioning with full lineage via `prev` pointers, while ENSIP-25 attestation records [PRODUCTION] can be added or removed as agent registrations change.

Table 3: Essential agent metadata mapped to ENS standards.

Metadata	Standard	Status
Identifier	ENS name (EIP-137)	[PRODUCTION]
Owner / Controller	Ethereum account (ENS registry)	[PRODUCTION]
Cross-chain registry links	ENSIP-25 `agent-registration`	[PRODUCTION]
Chain metadata	`on.eth` chain directory	[PRODUCTION]
Classification	NMS `class`	[DRAFT]
Capabilities / context	ENSIP-26 `agent-context`	[DRAFT]
Endpoints	ENSIP-26 `agent-endpoint[<protocol>]`	[DRAFT]
Policy / chain scope	AIP `payload.policy`	[DRAFT]
Version	AIP `agent-latest`	[DRAFT]

Organizational boundaries (2d): ENS subnames model organizational hierarchy (`agent.org.eth`) [PRODUCTION]. NMS `class` values [DRAFT] map organizational roles: `Agent`, `Org`, `Treasury`, `Delegate`, `Workgroup`. Ownership boundaries are enforced cryptographically via Ethereum accounts.

3.3 Authentication (§3)

Strong authentication (3a): Authentication operates at multiple layers of the stack. ENSIP-25 verification [PRODUCTION] provides deterministic authentication of the ENS-to-registry association: if the parameterized text record exists and is non-empty, the ENS name owner attests the association. This requires a single resolver lookup with no special infrastructure.

AIP three-check verification [DRAFT] adds cryptographic authentication: (1) `ensName` matches the resolved name, (2) `version` matches `agent-latest`, and (3) the signature is valid under the current ENS owner. A **default-deny** posture means that any failed check results in the agent being treated as unverified.

AIP operates at the naming/identity layer: proving the agent *is who it claims*. OAuth/OIDC operate above: authorizing *what the agent can do*. The relationship mirrors TLS certificate verification (identity) versus API key presentation (authorization).

Key management lifecycle (3b): Table 4 maps identity lifecycle phases to ENS mechanisms.

Table 4: Agent identity lifecycle management.

Phase	Production Infrastructure	Draft Extensions
Issuance	ENS name registration creates identity; owner key is initial controller	AIP manifest v1 (`prev: null`) adds cryptographic binding
Update	ENSIP-25 records added/removed; ENS ownership transfer updates controller	AIP versioning: new manifest with `prev` pointer preserves lineage
Revocation	ENS name release/transfer; text records clearable; ENSIP-25 stale records flagged	AIP manifest pointers removable; NMS `required` fields enforce validity
Rotation	ENS ownership transfer rotates controller key	Future AIP manifests must be signed by new owner
Delegation	ENS subnames enable hierarchical delegation	Explicitly deferred to AIP v2—deliberate minimal surface

3.4 Authorization (§4)

Zero-trust principles (4a): The ENS stack implements zero-trust at multiple levels:

- **Independent verifiability** [PRODUCTION]: each ENS text record is independently verifiable on L2s via Unruggable Gateways—no "trust the blob" required.
- **Deterministic verification** [PRODUCTION]: ENSIP-25 records either exist and are non-empty, or they do not. Binary, no ambiguity.
- **Immutable chain identifiers** [PRODUCTION]: `on.eth` interoperable addresses are locked after registration and cannot be modified.
- **Default-deny** [DRAFT]: AIP manifests that fail any verification check are rejected outright.
- **Content integrity** [DRAFT]: IPFS CIDs are self-verifying—tampering produces a different CID, which fails verification.

The layered trust model proceeds: resolve identity (ENS naming layer) → verify registry link (ENSIP-25) → verify claims (AIP) → authorize access (OAuth/OIDC) → execute action.

Dynamic policy updates (4b): ENSIP-25 records [PRODUCTION] can be updated to reflect new registry associations. AIP versioning [DRAFT] provides explicit capability change signaling: a new manifest with updated capabilities triggers `agent-latest` update, and clients see the change explicitly rather than discovering it after the fact.

Least privilege (4c): ENSIP-26 [DRAFT] exposes per-protocol endpoints, so clients discover only the capabilities relevant to their protocol. AIP `payload.policy` [DRAFT] declares explicit chain scope, capability lists, and spending limits. The `on.eth` delegation model [PRODUCTION] demonstrates least privilege at the infrastructure level: chain admins control only their own metadata.

Proving authority (4d): ENSIP-25 verification [PRODUCTION] provides a verifiable on-chain attestation chain: name → resolver → parameterized text record → owner. ERC-8004 on-chain registration provides independent proof of agent registration.

Delegation and human-in-the-loop (4f–4g): ENS subnames [PRODUCTION] model "on behalf of" relationships: `agent.org.eth` operates on behalf of `org.eth`. The ENS owner is an Ethereum account controlled by a human signer (or multisig/DAO) [PRODUCTION]. AIP [DRAFT] requires owner signature on all manifests—every identity claim traces to a human key. Delegation to non-owner signers is deliberately deferred to AIP v2 to maintain the strongest human-in-the-loop guarantee in the initial design.

3.5 Auditing and Non-Repudiation (§5)

ENS provides multiple mechanisms for tamper-proof logging and non-repudiation:

- ENS text record updates are on-chain transactions with block timestamps [PRODUCTION].
- ENS primary names provide human-readable agent attribution in block explorers [PRODUCTION].
- `on.eth` chain registrations and metadata changes emit on-chain events (`ChainRegistered`, `TextChanged`, `DataChanged`) [PRODUCTION].
- ENSIP-25 attestation records are on-chain and verifiable at any block height [PRODUCTION].
- AIP manifests on IPFS are content-addressed and immutable [DRAFT].
- AIP version lineage (`prev` pointers) creates an append-only audit chain; broken lineage must be labeled incomplete [DRAFT].
- AIP signature verification binds each manifest to a specific human or organizational key [DRAFT].

On-chain ENS ownership is publicly verifiable—anyone can confirm who controlled a name at a given block, providing non-repudiation for all identity actions taken under that name.

3.6 Prompt Injection Prevention (§6)

NMS schema validation [Draft] provides structural defense: malformed or unexpected metadata fields are rejected at the schema validation step before reaching agent runtime. AIP default-deny [Draft] ensures that tampered manifests fail signature verification.

We acknowledge that ENS identity standards primarily address the identity, authentication, and authorization layers—not the LLM prompt security layer. However, enforcing structural constraints on identity metadata reduces one vector through which injected content could reach agent systems.

4 Mapping to NIST's Areas of Interest

Table 5 maps each of NIST's five operational focus areas to specific ENS capabilities.

Table 5: ENS responses to NIST's five areas of interest.

Area	**ENS Response**
B1. Identification	NMS `class = "Agent"` [Draft] explicitly classifies agents vs. humans. ENSIP-25 [Production] verifies agent-to-registry associations. `on.eth` [Production] resolves cross-chain identifiers.
B2. Authorization	OAuth 2.0 operates above the naming layer; ENS operates below—telling the authorization server *who* the agent is [Production]. ENSIP-26 endpoints [Draft] direct clients to protocol-specific servers where OAuth handles access.
B3. Access Delegation	ENS subname hierarchy [Production] models delegation cryptographically. NMS `inherit` [Draft] propagates parent policies to children. AIP signature [Draft] creates a verifiable delegation chain.
B4. Logging & Transparency	All ENS record updates are on-chain transactions with block timestamps [Production]. Primary names give agents human-readable attribution in block explorers [Production]. AIP lineage [Draft] adds versioned historical records.
B5. Data Flow Tracking	Ensemble GUI-as-prompt-composer provides structured input provenance. Transparent evaluation rubrics enable reproducible AI decision audits.

5 ENS Position in the Broader Identity Stack

ENS does not "complement" the protocols NIST has identified—it provides the *naming layer* they operate above. DNS does not complement HTTP; DNS resolves names so HTTP can connect. ENS does not complement OAuth; ENS resolves agent identities so OAuth can authorize them.

Table 6 shows this layered relationship.

Table 6: ENS as a naming layer in the agent identity stack.

Layer	Protocol	Function	ENS Role
Naming	**ENS**	Resolve agent → address, metadata, endpoints	[PRODUCTION]
Cross-Chain	**on.eth + ENSIP-25**	Resolve chain-specific registry references	[PRODUCTION]
Communication	MCP, A2A, OASF	Tool discovery, agent messaging	[DRAFT] (ENSIP-26)
Authorization	OAuth 2.0, OIDC	Token issuance, scope management	ENS provides identity
Attestation	SPIFFE/SPIRE	Cryptographic workload identity	Different layer
Lifecycle	SCIM	Provisioning, de-provisioning	NMS defines attributes [DRAFT]
Policy	NGAC	Attribute-based enforcement	NMS/AIP provide attributes [DRAFT]

Alignment with NIST Standards

- **SP 800-207** (Zero Trust Architecture) [14]: AIP default-deny [DRAFT]; flat records independently verifiable [PRODUCTION]; immutable chain identifiers on `on.eth` [PRODUCTION].
- **SP 800-63-4** (Digital Identity Guidelines) [15]: ENS owner-key signing model provides identity proofing at the naming layer [PRODUCTION]; ENSIP-25 deterministic verification [PRODUCTION].
- **NISTIR 8587** (Attribute-Based Access Control Token Protection) [16]: AIP content-addressed manifests on IPFS provide integrity guarantees [DRAFT]; ERC-7930 immutable interoperable addresses via `on.eth` [PRODUCTION].

6 Demonstration Candidacy

We propose the ENS agent identity stack as a candidate technology for the NCCoE demonstration project. Table 7 summarizes how the stack meets NCCoE criteria.

Table 7: ENS stack against NCCoE demonstration criteria.

Criterion	ENS Stack
Commercially available	Public infrastructure since 2017, 2M+ names, permissionless. Four of seven stack layers deployed in production.
Open standards	All specifications CC0 licensed. Public governance via ENS Forum and GitHub.
Multiple implementations	Ensemble, Enscribe, ERC-8004 registrations, Agent Representation Protocol, Chain Registry-Resolver.
Addresses all 5 areas	Identification, authorization, delegation, logging, data flows.
Standards-based	EIP-137, ENSIP-5, ENSIP-24, ENSIP-25, ERC-7930, ERC-8004, JSON Schema 2020-12, IPFS.
Layered architecture	Provides the naming layer that NIST's listed protocols resolve against—same role as DNS.

Proposed Seven-Step Demonstration

Using production infrastructure (Layers 0–3):

1. **ENS name registration** [PRODUCTION]: Register an agent's ENS name with owner key.
2. **Cross-chain registry verification** [PRODUCTION]: Set ENSIP-25 `agent-registration` records linking the ENS name to ERC-8004 registrations on multiple chains.
3. **Chain resolution** [PRODUCTION]: Resolve registry chain references via `on.eth` reverse resolution (ERC-7930 → human-readable chain name).

Extending with draft capabilities (Layers 4–6):

4. **Agent discovery** [DRAFT]: Set `agent-context` and `agent-endpoint[mcp]` records.
5. **Classification** [DRAFT]: Set NMS `class = "Agent"` and attach a JSON Schema.
6. **Cryptographic identity** [DRAFT]: Sign an AIP manifest, upload to IPFS, publish version records.

End-to-end:

7. **Full resolution**: A consuming agent resolves the ENS name → verifies ENSIP-25 registry link → validates NMS class → verifies AIP manifest → discovers MCP endpoint → connects.

7 Conclusion

Every identity stack needs a naming layer. In the traditional internet, DNS provides this. For AI agents operating across heterogeneous blockchain and web environments, ENS provides it—with production infrastructure for cross-chain identity resolution already deployed, and agent-specific extensions under active development.

Four of the seven layers in the ENS agent identity stack are production infrastructure today. ENS standards authors are building the remaining three in direct response to the challenges NIST's concept paper identifies. ENS is not proposing to replace any framework NIST has identified; it is the layer *below* them—the layer they resolve against.

All specifications are open-source, CC0 licensed, and developed through public governance. We welcome the opportunity to collaborate as technology contributors to the NCCoE demonstration effort.

Contact: ENS Standards Authors — AI Agent Identity Working Group
https://discuss.ens.domains/c/ens-dev/ai/79

References

[1] H. Booth, B. Fisher, R. Galluzzo, and J. Roberts, "Accelerating the Adoption of Software and AI Agent Identity and Authorization," NIST NCCoE Concept Paper (Draft), Feb. 2026.

[2] N. Johnson, "EIP-137: Ethereum Name Service — Specification," Ethereum Improvement Proposals, 2016. https://eips.ethereum.org/EIPS/eip-137

[3] D. Lohnes, "ENSIP-5: Text Records," ENS Improvement Proposals, 2017. https://docs.ens.domains/ensip/5

[4] ENS Labs, "ENSIP-24: Arbitrary Data Resolution," ENS Improvement Proposals, 2025. https://docs.ens.domains/ensip/24

[5] ENS Labs, "ENSIP-25: Agent Registry Verification," ENS Improvement Proposals, 2025. https://docs.ens.domains/ensip/25

[6] ENS Labs, "Chain Registry-Resolver (on.eth)," GitHub repository, 2025. https://github.com/ensdomains/chain-registry-resolver

[7] A. Attar and N. Johnson, "ERC-7930: Interoperable Addresses," Ethereum Improvement Proposals, 2024. https://eips.ethereum.org/EIPS/eip-7930

[8] A. Attar, "ERC-7828: Chain Registry," Ethereum Improvement Proposals, 2024. https://eips.ethereum.org/EIPS/eip-7828

[9] ENS Standards Authors, "ENSIP-26: ENS-Native AI Agent Identity," Draft ENSIP, 2026. https://discuss.ens.domains/t/rfc-positioning-ens-as-a-foundational-layer-for-ai-agent-identity/21906/2

[10] ENS Standards Authors, "Node Metadata Standard (NMS)," Draft specification, 2026. https://discuss.ens.domains/t/rfc-positioning-ens-as-a-foundational-layer-for-ai-agent-identity/21906/2

[11] ENS Standards Authors, "Agent Identity Profile (AIP)," Draft specification, 2026. https://discuss.ens.domains/t/rfc-positioning-ens-as-a-foundational-layer-for-ai-agent-identity/21906/2

[12] "ERC-8004: Agent Registry," Ethereum Improvement Proposals, 2024. https://eips.ethereum.org/EIPS/eip-8004

[13] A. Wright, H. Andrews, B. Hutton, and G. Dennis, "JSON Schema: A Media Type for Describing JSON Documents," JSON Schema Specification, 2020. https://json-schema.org/specification

[14] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST SP 800-207, Aug. 2020. https://doi.org/10.6028/NIST.SP.800-207

[15] P. Grassi et al., "Digital Identity Guidelines," NIST SP 800-63-4, Dec. 2024. https://doi.org/10.6028/NIST.SP.800-63-4

[16] NIST, "Attribute-Based Access Control Token Protection," NISTIR 8587, 2024.