Agent Passport: Compiled Sovereign Metadata for Governed AI Agent Runtime Systems

Gustavo Venegas*, Edison Vazquez*, Danilo Naranjo†, Benjamin Gonzalez*
*Chilean Chamber of Artificial Intelligence †Ocular
gustavo@gobiernoia.cl, edison@jhedai.com, dan@ocularsolution.com, ben@nirvana-ai.com

***Abstract*—Agentic applications span model providers, tools, data flows, and inter-agent protocols, yet their jurisdictional, residency, identity, and governance assumptions are usually expressed outside the executable artifact, leaving a reviewability gap: a consumer cannot determine from one artifact which identity metadata was declared, which jurisdictional constraints applied, which policy bindings accompanied an agent, or whether those assumptions survived into runtime evidence. We present the *Agent Passport*, a construct in the ArgorixLang language and runtime that represents sovereign agent metadata—country, jurisdiction, permitted data residency, capabilities, policy bindings, and a local agent name (`ans_name`)—as a typed, semantically validated declaration that is lowered into bytecode and surfaced in runtime traces, security reports, and offline-verifiable evidence bundles. We hold a strict claim boundary: a passport is a *local declaration*, not legal identity, not decentralized-identifier (DID) verification, not credential issuance, not operational DNS, and not authentication. In an existing corpus of 33 request directories, 27 are complete and pass offline cross-artifact verification (27/27); each complete session carries two passports declaring country and jurisdiction Chile (`CL`) and residency `CL` and `EU`, and the bounded `injected.content` field is absent in all 27. The corpus shows passport metadata can be compiled and preserved into evidence; it does not show identity authentication, credential verification, operational discovery, legal compliance, or physical residency enforcement.**

***Index Terms*—AI agents, agent governance, sovereign metadata, identity metadata, runtime evidence, compiled governance, data residency, policy enforcement.**

I. Introduction

As agentic applications operate across model providers, external tools, and inter-agent protocols, they accumulate governance assumptions—*who* the agent claims to be, *under which jurisdiction* it operates, *where* its data may reside, and *which policies* bind it. In current practice these assumptions live outside the executable artifact: in prompts, deployment manifests, code comments, or human process. A reviewer then cannot reconstruct from one artifact which identity metadata was declared, which jurisdictional constraints were asserted, which policy bindings were associated with the agent, or whether any of this survived into the runtime record.

We call this the *reviewability gap* for sovereign agent metadata. Standardization efforts address adjacent surfaces—Model Context Protocol (MCP) and Agent2Agent (A2A) standardize interaction surfaces [1], [2]; W3C Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) define interoperable identity and credential data models [3], [4]—but none of these, by itself, makes an application's *local* governance metadata compile-checked and reproducibly present in its execution evidence.

This paper isolates and develops one construct from the ArgorixLang language and runtime [5], [6]: the *Agent Passport*. A passport is a typed, semantically validated declaration that binds an agent to a declared country, jurisdiction, permitted residency set, capabilities, policy bindings, and a local naming field (`ans_name`), together with optional identity, credential, and trust metadata. The compiler retains these declarations through semantic validation, lowers only validated forms into intermediate representation (IR) and bytecode, and the runtime surfaces the resulting metadata into a trace, a SecurityReport, and an offline-verifiable EvidenceBundle.

The central design stance—and the central claim boundary of this paper—is that the Agent Passport is about *representation, validation, and evidence preservation*, not real-world identity verification. The passport does not verify external identity, does not resolve a DID, does not issue or check a credential, does not operate as DNS, and does not prove that data physically resided in any region. What it provides is a *typed, inspectable, runtime-visible metadata boundary*: a place where sovereign metadata is declared once, checked for internal consistency, and carried into evidence that a downstream consumer can inspect offline.

Contributions.

1) A compiled *Agent Passport abstraction* for governed AI agents, treating sovereign metadata as a first-class, type-checked language construct rather than out-of-band configuration.
2) A *semantic validation model* for passport–agent–policy–residency bindings, including rejection of dangling references and invalid bindings.
3) A *runtime evidence path* showing how passport metadata survives compilation into IR/bytecode and appears in traces, reports, and offline-verifiable bundles.
4) A *claim-boundary taxonomy* separating *implemented* mechanisms, *declarative* metadata, *proposed* architecture, and properties *not claimed*—applied uniformly so digest success is never promoted to external assurance.
5) An *observational evaluation* over the existing ArgorixLang runtime artifact corpus, reporting what passport evidence the corpus does and does not establish.

To prevent overreading, every claim is assigned to one of four regions (implemented / declarative / proposed / not

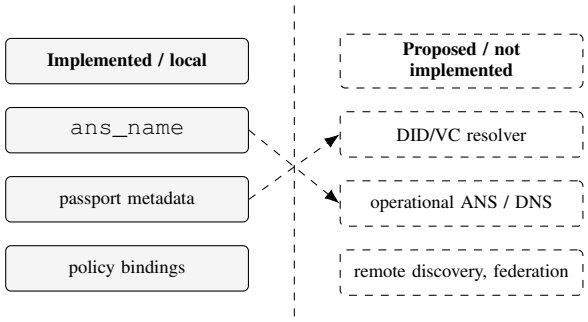

Fig. 1. Boundary between local metadata and external verification. Left items are implemented as local declarations; right items are proposed and unbuilt; dashed edges are not-yet-implemented links.

claimed) before it is discussed (Table V, Fig. 1). We make no claim of real identity authentication, credential verification, cryptographic handshake execution, legal-compliance certification, blockchain immutability, post-quantum security, or production isolation. A scope note relative to the original preprint is given in Section XII.

II. Background and Motivation

A. Why external metadata is hard to review

When sovereign metadata lives outside the executable artifact, three failure modes follow that the passport is designed to address: a reviewer cannot tell an *omitted* constraint from an *approved* one; two artifacts (e.g., a manifest and the running program) can silently disagree; and a favorable human-facing summary can mask an unfavorable underlying decision.

B. Naming vs. identity vs. credentials vs. policy

A recurring category error conflates four distinct functions. A *resolvable name* answers "where is the record?". An *authenticated identity* answers "who controls the relevant key or account?". A *credential* answers a scoped, issuer-signed assertion. A *policy decision* answers "is this operation permitted in context?". These cannot substitute for one another. The Agent Passport deliberately occupies only the *local declaration* slice of this space: it records a local name and locally asserted metadata and binds them to policy declarations, without claiming to resolve, authenticate, or verify anything external.

C. Why residency and jurisdiction metadata matter

Data-residency and jurisdiction constraints are increasingly first-order governance inputs for agent deployments operating across regions. Today these constraints are usually informal. Making them *explicit and compile-checked* —even as declarations—lets a runtime record which residency set and jurisdiction accompanied an operation, and lets a reviewer detect when a declared residency value lies outside an allowed set. This is an auditability benefit, not an enforcement guarantee: a declared residency of `EU` does not prove that bytes were stored in the EU.

D. Distinguishing the passport from adjacent systems

DID/VC [3], [4] define *external* identity and credential models with cryptographic control and issuer signatures; the passport stores only local declarations unless a resolver/verifier is implemented, and none is here. DNS / naming / NANDA-style discovery [7]–[9] provide operational resolution; the passport's `ans_name` is local metadata and operational discovery is proposed, not implemented. MCP/A2A [1], [2] standardize interaction; passport metadata can bind to such boundaries but no live interoperability is implemented here. Policy-as-code (OPA) [10] is an external decision engine; ArgorixLang compiles policy metadata and passport bindings together with the program.

III. Agent Passport Model

A. Definition

An *Agent Passport* is a local, compiled declaration that collects locally asserted identity and governance context for one subject agent. Its fields are: a *passport name* (local identifier); a *subject agent* (reference to an agent declared in the same program); *country* (observed `CL`); *jurisdiction* (observed `CL`); *permitted residency* (a set of allowed values, observed `CL` and `EU`); *capabilities*; *policy bindings* (references to policy declarations); an `ans_name` (local agent naming field); and optional *identity / credential / trust metadata* referencing identity, credential, and ATrust declarations [11]. The passport relates to agent declarations (the subject), policies (bindings), runtime profiles (which govern runtime behavior), and evidence requirements (which determine what is emitted).

B. What the passport is—and is not

The passport *is* a local compiled declaration that creates a typed, inspectable, runtime-visible metadata boundary; within the compiled program it binds a name and sovereign metadata to a specific agent and policies, and that binding is checked. The passport *is not* a government document, *is not* legal identity, *does not* verify external identity, and *does not* prove physical data residency. *Sovereign* here denotes explicit jurisdictional and residency metadata controlled by the program, not state recognition or legal status. No external registry validates the declared country, jurisdiction, or residency values. Table I summarizes the fields and their assurance ceiling, and Fig. 2 shows the binding model.

IV. Compilation and Semantic Validation

ArgorixLang groups declarations into modules and processes them through five relevant stages; passports traverse all five. *(1) Parsing* produces a source-faithful AST; passport declarations are retained structurally, not rewritten or discarded. *(2) Name and module resolution* binds declarations without discarding provenance, so a passport's subject-agent and policy-binding references resolve against the program's declared names. *(3) Semantic analysis* checks uniqueness, references, cross-construct constraints, and denied combinations: the subject agent must exist; each policy binding must reference an existing policy; residency values must belong to

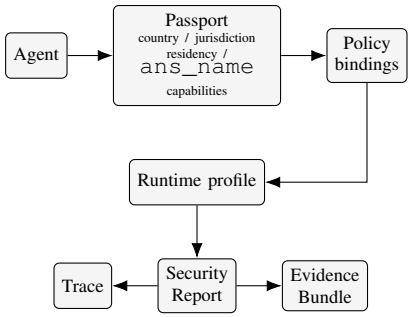

Fig. 2. Agent Passport binding model. Edges denote validated in-program references and preserved metadata, not verified real-world relationships.

TABLE I
AGENT PASSPORT FIELDS AND SEMANTICS.

Field	Meaning / implemented status	External assurance claim
Passport name	Local identifier; parsed, validated, lowered, emitted	None
Subject agent	Reference to in-program agent; resolution enforced	None
Country	Declared; preserved into evidence (declarative)	None (not validated)
Jurisdiction	Declared; preserved into evidence (declarative)	None
Permitted residency	Allowed value set; membership-checkable	None (no physical proof)
Capabilities	Capability references; binding validated	None beyond local checks
Policy bindings	References to policies; resolution enforced	None (binding $\neq$ approval)
`ans_name`	Local naming field; survives to artifacts	None (no DNS/DID/registry)
Identity/cred./ trust	References into declarations; validated	None (no issuance/ verify)

an allowed declaration set; identity/credential/trust references must resolve. *(4) Lowering* converts only validated forms to IR and serializable bytecode, *preserving* passport metadata rather than erasing it. *(5) VM execution* loads bytecode, installs only explicitly executable providers, and evaluates policy before scheduling work, making passport constraints and metadata available to the evidence pipeline.

This ordering matters: a runtime-only check discovers invalid configuration after deployment, and an unchecked lowering pass can erase the distinction between an omitted field and an approved one.

A. Declarative metadata vs. executable controls

Executable runtime controls (e.g., the fail-closed provider boundary, network/secret denial) change what the runtime *does*. *Declarative metadata* (country, jurisdiction, residency, identity/credential references, `ans_name`) is validated and preserved but does not, by itself, perform an external action. The passport is primarily declarative metadata with implemented binding/validation and implemented evidence preservation; it does not execute identity verification or residency enforcement.

TABLE II
PASSPORT SEMANTIC VALIDATION RULES.

Rule	Condition	Effect
R1	`P.subject` $\in A$	reject: dangling agent reference
R2	`b` $\in \Pi$ for each policy binding	reject: dangling policy reference
R3	`r` $\in R$ for each residency value	reject: residency outside set
R4	`ans_name` is local metadata only	no resolution/authentication (no resolver)
R5	identity/credential/trust refs stay declarative	no verification (no verifier)
R6	emit `lower(P)` iff R1 $\wedge$ R2 $\wedge$ R3	country/jurisdiction/residency/ `ans_name` kept in IR/ bytecode

B. Validation rules

Let P be a passport, A the declared agents, Π the declared policies, and R the allowed residency declaration set. Table II states the six rules, their condition, and the effect when the condition is not met.

Rules R1–R3 are enforced bindings/validations; R4–R5 are *boundary* rules describing the semantic ceiling of a validated passport; R6 makes explicit that successful lowering is conditioned on reference and membership validity and that the surviving metadata is what later appears in evidence. None of these rules establishes that a declared value is externally *true*.

V. RUNTIME EVIDENCE PATH

Passport metadata becomes auditable because it is carried into runtime evidence artifacts. Each *complete* request directory contains five artifacts: source (`session.argx`), compiled bytecode (`session.argbc.json`), a trace (`session.trace.json`), a security report (`session.security.json`), and an evidence bundle (`session.evidence.json`).

The *trace* is an ordered ledger of compiler/VM and policy events; among its declaration events, each complete trace records declarations for two agents, six policies, *two passports*, plus governance, threat, conformance, release, ATrust, MCP, and A2A metadata. Passport declarations therefore appear in the runtime ledger, not only in source. The *SecurityReport* is a structured interpretation of runtime evidence: execution status, policy results, review state, provider boundary, and declared controls. The *EvidenceBundle* records four semantic SHA-256 digests—`bytecode_digest`, `trace_digest`, `report_digest`, and `ledger_digest` (the canonical digest of `trace.events`)—and an artifact map of `bytecode_path`, `trace_path`, and `security_report_path`. There is no source path or source digest and no standalone ledger path.

Digests are semantic (computed over canonical JSON, independent of whitespace or key ordering), and *offline verification* recomputes them and the ledger digest without network access

or provider credentials. The detailed verification algorithm is given in the original preprint [5]; the point here is its scope: it begins at compiled bytecode and never asserts that `session.argx` produced it, so passport metadata is verified *as carried into bytecode and trace*, not as traced back to source (Fig. 3).

Key claim. The Agent Passport lets runtime consumers *inspect which sovereign metadata and policy bindings accompanied an agent operation*. It does *not* prove the metadata is externally true. Three orthogonality statements bound this: a valid digest detects content change relative to the bundle but does not prove who created it; a linked trace does not prove that events outside the instrumented runtime did not occur; and artifact integrity does not turn a failing policy result into approval. No signature, trusted timestamp, transparency service, hardware root, or independent witness is evaluated here. Bundle integrity therefore does *not* imply legal compliance, certification, authentication, or policy approval.

VI. Observational Evaluation

We reuse the existing ArgorixLang runtime artifact corpus. All values are taken directly from the source preprint's normalized datasets [5]; we add no new experiment.

A. Corpus and completeness

Of **33** request directories, **27** (81.8%) contain all five expected artifacts and **6** (18.2%) contain source only. All 27 complete sessions report execution status `completed`; execution status is unavailable for the six source-only directories, which cannot be assessed for trace-level fields. We retain the six as explicit incomplete observations.

B. Offline bundle verification

All **27/27** complete bundles pass the repository's offline verifier. For each, the bytecode, trace, security-report, and trace-event-ledger digest fields are syntactically valid SHA-256 identifiers, the three semantic digests reproduce, the ledger digest recomputes from trace events, and the Ledger-Summary digest matches. The verifier does not digest or verify `session.argx`. Verification establishes internal cross-artifact consistency, not producer authentication, absence of side effects, or policy approval.

C. Passport evidence

Every complete session reports **two passports**, with country and jurisdiction Chile (`CL`) and data-residency declarations including `CL` and `EU`. These values demonstrate that sovereign metadata survives compilation and appears in runtime evidence. They do *not* show that data was physically stored in either region, nor that any external registry validated the declarations.

D. Prompt-content availability

The bounded `injected.content` field is present in **0 of 27** complete, valid traces; the six source-only directories have no trace and are not assessable. Consequently no qualitative prompt themes, quotations, injection labels, or attack-success rates can be reported—simultaneously a data-minimization benefit and an evaluation limitation. Table III summarizes the observed passport evidence.

TABLE III
Observed passport evidence in the runtime corpus.

Measure	Observed	Interpretation
Request directories	33	Full corpus
Complete sessions (5 artifacts)	27	Assessable for trace fields
Source-only sessions	6	Not assessable
Bundles passing offline verification	27 / 27	Internal consistency only
Passports per complete session	2	Metadata survives to evidence
Country / jurisdiction	`CL` / `CL`	Declaration, not validated
Residency declarations	`CL`, `EU`	Not physical-residency proof
Traces with `injected.content`	0 / 27	No prompt-level analysis

E. Interpretation

The corpus demonstrates that passport metadata can be compiled and preserved into runtime evidence and that complete bundles are offline-consistent. It does *not* demonstrate real identity authentication, DID/VC verification, operational DNS/ANS resolution, legal compliance, or physical residency enforcement. The repeated structure across complete sessions supports pipeline reproducibility but weakens any claim about behavioral diversity.

VII. Security and Governance Analysis

We analyze threats as attack paths across trust boundaries, not as proof that a named control defeats every adversary.

Threats. *Identity spoofing*: a program may declare arbitrary identity/credential references; local checks prevent dangling references but cannot establish that a referenced identity is genuine. *False residency declaration*: `permitted_residency` may be any in-set value; membership is checked but truthfulness is not. *Policy confusion*: a consumer reading only a coarse top-level status could overlook an unfavorable detailed result. *Metadata laundering*: re-emitting a self-consistent bundle with attacker-chosen sovereign metadata is possible if the bundle is unsigned and artifacts and bundle are replaced together. *Overreading*: the most pervasive risk is treating a local declaration as external assurance.

Bounded mitigations. Semantic validation rejects dangling references and out-of-set residency values (R1–R3, R6); explicit claim boundaries (Table V) keep declarative metadata from being promoted to assurance; evidence linkage via semantic digests detects content change relative to a bundle; review states preserve component-level policy status; and separation of local naming from operational discovery prevents a validated `ans_name` from being mistaken for a resolved identity.

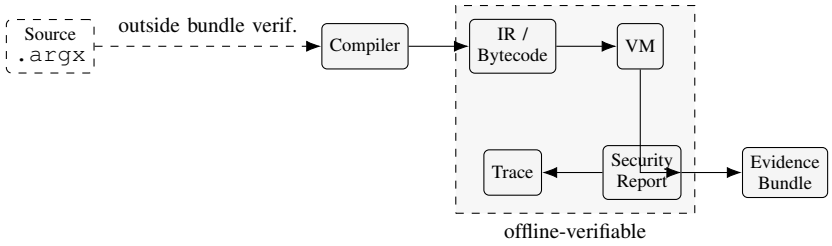

Fig. 3. Runtime evidence path. The source-to-compiler edge is dashed and lies outside bundle verification; bytecode, trace, and report form the offline-verifiable region (semantic digests plus ledger digest).

TABLE IV
THREATS AND BOUNDED MITIGATIONS (PASSPORT-FOCUSED).

Threat	Bounded mitigation	Claim status
Dangling identity/policy ref	Semantic reference validation	Implemented
Residency outside allowed set	Membership check (R3)	Implemented
Policy confusion (coarse vs. detailed)	Preserved component status / review	Implemented
Evidence tampering	Digest-linked bundle	Implemented (no producer auth)
False/unverified external identity	— (verifier not implemented)	Not claimed
Physical residency enforcement	—	Not claimed

Remaining risks. Malicious-but-well-formed declarations, fake credentials, resolver attacks (once a resolver exists), missing revocation, adapter compromise, and unsigned bundles all remain in scope for future testing. The strongest supported statement is conditional: within the observed runtime path, requests are mediated by compiled policy and an executable-provider allowlist, denials are recorded, passport metadata is preserved, and complete artifacts are offline-consistent. Security outside that path is unmeasured. Table IV states the bounded mitigation claims.

VIII. Related Work

We compare carefully and claim no equivalence. DID Core and Verifiable Credentials [3], [4] provide external identity and credential models with cryptographic control and issuer signatures; the passport stores only local declarations and validated references and performs no DID resolution or credential verification unless a verifier is added. DNS / naming / NANDA-style discovery [7]–[9], [12] provide operational resolution and verified agent facts; the passport's `ans_name` is local metadata, and sovereign DNS/ANS discovery is proposed future work. MCP and A2A [1], [2] standardize interaction surfaces; passport metadata can bind to MCP/A2A boundary declarations, but no live interoperability is implemented. OPA [10] decouples policy decision from enforcement as an external engine; ArgorixLang instead compiles policy metadata and passport bindings within the program. in-toto [13] provides signed supply-chain guarantees; ArgorixLang EvidenceBundles are local, unsigned, semantic-digest bundles unless extended. WASI [14] provides host-level capability isolation; the passport model makes no host-isolation claim. Agent languages [15] and AI-risk and LLM-security guidance [16], [17] provide complementary framing.

IX. Limitations

We state limitations strictly. There is *no* external identity verification; *no* DID/VC resolver; *no* cryptographic proof of control (no handshake, challenge–response, or key-control proof); *no* legal-compliance certification of declared country/jurisdiction/residency; *no* physical residency enforcement; *no* production-isolation proof (runtime denials are VM-level control-flow properties, with OS/process/network/secret-store containment untested); *no* controlled adversarial experiment (the corpus has no prompt text and no controlled attacks, so no injection success rate is reported); and *no* live provider federation (only a `simulated` provider is executable; external execution is recorded as blocked). Source integrity is outside bundle verification, and bundles are unsigned. Complete sessions are structurally uniform, limiting generalization across prompts, jurisdictions, and policy sets.

X. Future Work

DID/VC resolver integration to turn declarative identity/-credential references into verifiable claims (introducing trust anchors, replay, revocation, privacy, and network-failure concerns); signed passport attestations and signed EvidenceBundles for producer authentication and non-repudiation; source-digest inclusion so verification extends to `session.argx`; a transparency log for independent witnessing; an operational ANS / Sovereign DNS prototype giving `ans_name` real resolution semantics; policy-decision canonicalization into one tri-state decision from typed component results; controlled adversarial evaluation once prompt content is available under an allowlist; host-level sandbox measurements; and cross-provider runtime tests with live adapters.

XI. Conclusion

The Agent Passport provides a compiled, evidence-visible metadata boundary for governed AI agents. It represents sovereign metadata—country, jurisdiction, permitted residency, capabilities, policy bindings, and a local `ans_name`—as a typed, semantically validated declaration; lowers validated

TABLE V
CLAIM-BOUNDARY TAXONOMY (SOVEREIGN-METADATA VIEW).

Concept	Implemented	Declarative	Proposed	Not claimed
Local naming	`ans_name` binding, preserved	sovereign naming metadata	operational ANS/DNS	DNS/registry resolution
Identity / cred.	reference validation	identity/cred. refs, ATrust maps	DID/VC resolver, signed attest.	external auth, cred. verify
Jurisdiction / residency	evidence preservation; membership check	country/juris./residency decls	—	legal cert., physical proof
Policy binding	reference resolution; profile binding	policy associations	decision canonicalization	binding-as-approval
Evidence	semantic-digest bundle, offline verify	trace/report fields	signed bundles, transparency log	producer auth, immutability
Provider boundary	fail-closed path, simulated allowlist	contract metadata	live federation	external exec, prod. isolation

metadata into IR and bytecode; and preserves it into traces, security reports, and offline-verifiable evidence bundles. An observational study of an existing corpus shows that passport metadata survives compilation and appears in 27/27 complete, offline-consistent sessions, each carrying two passports with declared Chilean country/jurisdiction and `CL`/`EU` residency. The construct's value is not external truth. It is explicit representation, semantic validation, runtime visibility, and claim-bounded auditability: a runtime consumer can inspect which sovereign metadata and policy bindings accompanied an operation, while the metadata remains a local declaration rather than verified identity, certified compliance, or physical-residency proof.

XII. SCOPE

This paper extends and narrows the ArgorixLang preprint [5] by focusing specifically on the Agent Passport construct and its sovereign-metadata evidence path. The ATrust and DCP-AI conceptual framing [11], [18] and the full language/runtime are treated there; here all empirical values are reused unchanged, and no new experiment, benchmark, deployment, or security guarantee is introduced.

REFERENCES

[1] Model Context Protocol Contributors, "Model context protocol specification, version 2025-06-18," Protocol specification, 2025. [Online]. Available: https://modelcontextprotocol.io/specification/2025-06-18

[2] A2A Protocol Working Group, "Agent2agent (A2A) protocol specification," Protocol specification, 2026. [Online]. Available: https://a2a-protocol.org/v1.0.0/specification/

[3] World Wide Web Consortium, "Decentralized identifiers (DIDs) v1.0: Core architecture, data model, and representations," World Wide Web Consortium, W3C Recommendation, Jul. 2022. [Online]. Available: https://www.w3.org/TR/2022/REC-did-core-20220719/

[4] ——, "Verifiable credentials data model v2.0," World Wide Web Consortium, W3C Recommendation, May 2025. [Online]. Available: https://www.w3.org/TR/2025/REC-vc-data-model-2.0-20250515/

[5] G. Venegas, E. Vazquez, D. Naranjo, and B. Gonzalez, "ArgorixLang: Compiled governance, sovereign agent metadata, and offline-verifiable runtime evidence," Zenodo preprint, 2026, original ArgorixLang preprint; source material for this derived paper. [Online]. Available: https://doi.org/10.5281/zenodo.21014780

[6] Argorix Labs, "ArgorixLang," 2026, secure, verifiable programming language and runtime for AI-agent systems. [Online]. Available: https://github.com/argorixlabs/argorixlang

[7] P. Mockapetris, "Domain names—concepts and facilities," Internet Engineering Task Force, RFC 1034, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1034.html

[8] Project NANDA, "Architecting the internet of AI agents," Project website, 2026, project NANDA (Networked AI Agents in Decentralized Architecture). [Online]. Available: https://projectnanda.org/

[9] R. Raskar, P. Chari, J. Zinky, M. Lambe, J. J. Grogan, S. Wang, R. Ranjan, R. Singhal, S. Gupta, R. Lincourt, R. Bala, A. Joshi, A. Singh, A. Chopra, D. Stripelis, B. B, S. Kumar, and M. Gorskikh, "Beyond DNS: Unlocking the internet of AI agents via the NANDA index and verified AgentFacts," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2507.14263

[10] Open Policy Agent Project, "Open policy agent (OPA)," Official project documentation, 2026, general-purpose policy engine and Rego policy-as-code language. [Online]. Available: https://www.openpolicyagent.org/docs

[11] E. Vazquez, "ATrust: Agentic trust protocol," author-supplied metadata used in this study; unpublished conceptual protocol.

[12] P. Mockapetris, "Domain names—implementation and specification," Internet Engineering Task Force, RFC 1035, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1035.html

[13] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos, "in-toto: Providing farm-to-table guarantees for bits and bytes," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1393–1410. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias

[14] WASI Subgroup, "Introduction," Official standards portal, 2026, standards-track APIs and capability-based sandbox model. [Online]. Available: https://wasi.dev/

[15] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007. [Online]. Available: https://doi.org/10.1002/9780470061848

[16] E. Tabassi, "Artificial intelligence risk management framework (AI RMF 1.0)," National Institute of Standards and Technology, NIST AI 100-1, Jan. 2023. [Online]. Available: https://doi.org/10.6028/NIST.AI.100-1

[17] OWASP Foundation, "OWASP top 10 for large language model applications," Security guidance, 2025. [Online]. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/

[18] D. Naranjo Emparanza, "Agents don't need a better brain—they need a world: Toward a digital citizenship protocol for autonomous AI systems," Engineering Archive preprint, 2026, DCP-AI: Digital Citizenship Protocol for Autonomous AI. [Online]. Available: https://engrxiv.org/preprint/view/6671