A Survey of AI Agent Registry Solutions

Aditi Singh, Abul Ehtesham, Ramesh Raskar, Mahesh Lambe,
Pradyumna Chari, Jared James Grogan, Abhishek Singh, Saket Kumar

Project NANDA

Abstract

As autonomous AI agents scale across cloud, enterprise, and decentralized environments, the need for standardized registry systems to support discovery, identity, and capability sharing has become essential. This paper surveys three prominent registry approaches—each defined by a unique metadata model: MCP's mcp.json, A2A's Agent Card, and NANDA's AgentFacts. MCP uses a centralized metaregistry with GitHub-authenticated publishing and structured metadata for server discovery. A2A enables decentralized interaction via JSON-based Agent Cards, discoverable through well-known URIs, curated catalogs, or direct configuration. NANDA Quilt introduces AgentFacts, a cryptographically verifiable and privacy-preserving metadata model designed for dynamic discovery, credentialed capabilities, and cross-domain interoperability. We compare these approaches across four key dimensions: security, scalability, authentication, and maintainability. The paper concludes with suggestions and recommendations to guide future design and adoption of registry systems for the Internet of AI Agents.

1 Introduction

Autonomous AI agents are rapidly becoming foundational across domains—from cloud-native assistants and robotics to decentralized systems and edge-based IoT controllers. These agents act independently, make decisions, and collaborate at scale. As agent populations grow into the billions across heterogeneous platforms and administrative boundaries, the ability to *identify, discover, and trust agents in real time* has emerged as a critical infrastructure challenge.

Traditional mechanisms like DNS and static service catalogs are poorly suited to agent ecosystems, which demand *dynamic discovery*, *verifiable metadata*, and *privacy-preserving interactions*. Legacy systems assume fixed endpoints and ownership-based trust models, lacking the flexibility and cryptographic assurances needed for agents that rotate capabilities, change locations, and form ephemeral collaborations.

To address these limitations, several agent frameworks have introduced discovery metadata models. This paper focuses on three emerging approaches:

In this paper, we survey and compare four representative registry architectures that address these challenges in distinct ways:

- **MCP Registry** (Anthropic, 2025) [1] which provides a centralized metadata layer using structured `mcp.json` files for agent discovery and installation .
- **A2A Agent Cards** [2] which describe agent capabilities and endpoints in a standardized JSON format, enabling discovery via well-known URLs, curated catalogs, or private distribution.
- **Microsoft Entra Agent ID** (Microsoft, 2025) [3]: a SaaS-delivered, enterprise-grade directory built into Azure AD, offering built-in lifecycle, governance, and zero-trust controls.
- **NANDA Quilt: AgentFacts**, (MIT AIDE, 2025) [4] a decentralized, cryptographically verifiable format supporting dynamic resolution, credentialed assertions, and privacy-preserving queries.

Rather than surveying general agent communication protocols [5], this work is a focused comparison of these *AI registry solutions*. It explores how each approach supports real-time discovery, identity validation, and cross-domain interoperability.

The paper is organized as follows: Section II outlines background and motivation for agent registries. Section III introduces the evaluation framework and functional criteria. Sections IV–VI examine MCP, A2A, Microsoft Entra Agent ID and NANDA in depth. Section VII provides a comparative analysis across security, scalability, authentication, and maintainability. Section VIII concludes with design suggestions and practical recommendations for registry adoption in multi-agent systems.

By comparing these registries in the context of emerging agent infrastructure needs, this survey highlights the current gaps and emerging solutions driving the future of the Internet of AI Agents.

2 Background

The modern web operates on a reactive, client-driven model in which services wait for external requests before responding. Despite significant advances in cloud automation and event-driven design, this architecture remains largely inadequate for the emerging Internet of AI Agents—a paradigm shift where autonomous, goal-directed software agents negotiate, coordinate, and act proactively on behalf of users. Unlike traditional web resources, which are typically stateless and short-lived, autonomous AI agents are persistent computational entities capable of initiating control flow, retaining long-term memory, dynamically adapting to context, and spawning subordinate agents. These agents require infrastructure that supports high-frequency updates, real-time identity resolution, and trustable metadata exchange across heterogeneous systems and organizational boundaries.

This shift introduces significant challenges for discovery and coordination. The current Internet stack—built on DNS, IP addressing, and certificate authorities—was not designed to handle trillions of fast-moving, self-directed agents. Limitations in revocation latency, state propagation, identity verification, and routing scale all become critical bottlenecks.

To address these gaps, new registry models are emerging that shift away from static name-resolution systems to dynamic, metadata-rich discovery layers tailored to autonomous agents. This paper focuses on three such models, each coupled with a distinct metadata schema:

- **MCP Registry:** A centralized metaregistry that enables structured agent metadata publishing via `mcp.json`, supporting installability and versioning for MCP-compatible agents.
- **A2A Agent Cards:** A flexible, decentralized format for agent self-description, enabling discovery via well-known URLs, curated registries, or configuration files.
- **NANDA AgentFacts:** A cryptographically verifiable, privacy-preserving metadata schema designed for dynamic resolution, credentialed capability assertions, and federated environments.

These registry systems are positioned to address the foundational needs of the Internet of AI Agents: sub-second identity resolution, schema-validated capability representation, verifiable trust models, and privacy-aware discovery. Each model offers a different architectural stance—centralized, federated, or decentralized—on how to meet these requirements. This survey situates these three approaches within the broader transformation of web infrastructure, drawing historical parallels to transitions such as dial-up to broadband and IPv4 to IPv6. By understanding the limitations of existing systems and the unique demands of AI agents, we highlight why purpose-built registries are essential for scalable, secure, and interoperable agent ecosystems.

2.1 Design Evaluation Dimensions

To compare candidate registry architectures against the above requirements, we evaluate along four core dimensions.

- **Security:** Integrity of registry records and metadata via cryptographic signing. Resistance to spoofing, registry poisoning, and man-in-the-middle attacks.
- **Authentication:** Mechanisms for publisher identity verification (e.g., GitHub OAuth + DNS-TXT, DID-VC issuance, X.509 PKI). How registry updates are gated and how namespace ownership is enforced.
- **Scalability:** Ability to handle high lookup volumes and large agent populations via TTL-based caching, federated deployments, or CDN offload. Support for low-latency, geo-distributed resolution.
- **Maintenance:** Operational simplicity: schema-first designs, minimal core code, decoupled metadata hosting. Ease of upgrades, migration paths, and reduced patch surface by avoiding executable code hosting.

These dimensions provide a structured, source-grounded rubric for the comparative analysis in Sections 4–9.

3 MCP Registry

The MCP registry is a centralized "metaregistry" for discovering and installing MCP servers. Publishers push a versioned `mcp.json` via a CLI tool that performs a GitHub OAuth flow and, for reverse-DNS namespaces, a DNS TXT challenge. The Go-based REST API exposes read endpoints (no authentication) and write endpoints (GitHub OAuth + DNS verification), stores raw JSON in object storage, indexes metadata in MongoDB (with an in-memory fallback), and generates asynchronous jobs or webhooks. Downstream MCP client apps poll the registry (or private mirrors), cache the data locally, and serve end-users without direct live calls to the central service.

3.1 Security

The registry only accepts metadata from authenticated GitHub identities and, for domain-scoped namespaces, from DNS-verified domains. It does not host executable code; instead it holds metadata only, inheriting code-level security from established registries (npm, PyPI, DockerHub). This minimizes the attack surface and delegates authentication and domain control to proven systems.

3.2 Authentication

All publish requests require a GitHub OAuth bearer token tied to the submitting user or organization. For reverse-DNS namespaces (e.g. `com.microsoft`), a DNS TXT record proof is required and linked to that GitHub identity. Read operations are openly accessible.

3.3 Scalability

Only a small number of MCP client applications query the central registry; they cache and serve data to millions of end-users. The API supports asynchronous processing and webhooks. Metadata is stored in MongoDB—suited to flexible, document-style records—and served via CDN-cacheable HTTP endpoints; optional middle-layers (private mirrors, curated feeds) can shard load.

3.4 Maintenance

The registry's core service is schema-driven by `mcp.json` (OpenAPI/JSON Schema), with no package hosting or scanning to maintain. A CLI tool automates publication and verification flows. Schema updates proceed independently of the service code, and validation logic resides with the publisher.

4 Agent2Agent (A2A) Protocol

The Agent2Agent (A2A) protocol is a transport-agnostic, enterprise-ready standard for inter-agent communication across heterogeneous systems. A2A enables autonomous agents—potentially opaque, vendor-specific, or closed-source—to discover, negotiate, and collaborate using a shared JSON-RPC interface over secure HTTP transport.

A2A is optimized for asynchronous, long-running, multimodal, and streaming interactions, supporting flexible task handoff between agents without requiring visibility into internal execution models. Through its declarative AgentCard, it enables dynamic discovery of skills, capabilities, and authentication requirements, establishing a standardized model for agent-to-agent collaboration.

4.1 Security

A2A relies on transport-layer security (TLS) and established web security best practices. Identity and authentication are handled outside of the A2A JSON-RPC payload—via standard HTTP headers—allowing compatibility with OAuth2, API keys, and mTLS. Server identity is verified via TLS certificates, while clients authenticate based on security schemes advertised in the AgentCard. Push notifications (webhooks) are authenticated using per-client credentials, tokens, or schemes negotiated during setup. Agents do not share internal states; interactions are scoped to declared capabilities and managed through tasks and artifacts, reducing attack surfaces and limiting data exposure.

4.2 Authentication

AgentCards explicitly declare supported authentication mechanisms using OpenAPI-style security schemes (e.g., Bearer tokens, OpenID Connect, API keys). Clients must obtain credentials out-of-band and include them in request headers. Each RPC call is authenticated individually, and servers return HTTP 401/403 responses with guidance when credentials are missing or invalid. During execution, if secondary credentials are needed (e.g., to proxy tool access), tasks transition to auth-required, and clients supply the required credentials in subsequent messages.

4.3 Scalability

A2A's task-based, stateless transport over HTTP and SSE enables horizontal scalability across distributed agent systems. Agents define capabilities declaratively in AgentCards, allowing registries and discovery services to dynamically catalog available services. Tasks are long-lived objects with unique IDs, status updates, and artifact streams. Streaming (via SSE) and push notifications reduce polling overhead. Task lifecycles support fine-grained eventing (submitted, working, input-required, etc.), enabling responsive and resilient orchestration even in failure-prone environments.

4.4 Maintainability

A2A is intentionally simple and extensible, built atop HTTP and JSON-RPC 2.0. It minimizes custom logic and avoids bespoke protocols. Features like AgentCard and structured data formats (e.g., TextPart, DataPart, FilePart) ensure consistent interpretation of messages while allowing modality diversity. Schema evolution is flexible: agents can define capabilities per skill, override input/output MIME types, and extend their AgentCard dynamically. Task handling and message structure follow consistent, extensible conventions.

5 Microsoft Entra Agent ID

Microsoft Entra Agent ID provides a managed, enterprise-grade directory for AI agent identities. Agents created in Copilot Studio or Azure AI Foundry automatically appear as "Agent ID" applications in the Entra admin center. Identity practitioners gain visibility, lifecycle management, and access governance for these non-human identities using the same tools and policies as for user or service identities. Upcoming features include least-privilege token issuance, expanded Conditional Access, and cross-tenant identity federation. Further analysis of Microsoft Entra Agent ID's security, authentication, scalability, and maintainability will be possible once technical documentation and operational data are available.

Figure 1: Microsoft Entra Agent ID Overview [3]

6 NANDA Quilt

The Networked Agents and Decentralized AI (NANDA) Quilt Registry presents a lean, modular architecture for agent discovery in decentralized environments. Designed specifically for scale, privacy, and interoperability, NANDA separates static identifier resolution from dynamic agent metadata to enable rapid discovery, credentialed verification, and flexible routing across federated agent ecosystems.

At the core of the design is the concept of a minimal AgentAddr record an $Ed25519-$signed object that maps agent identifiers to one or more verifiable metadata locations: a public FactsURL, an optional privacy-preserving PrivateFactsURL, and an AdaptiveRouterURL for real-time routing. These records are lightweight ($\leq$ 120 bytes), cacheable, and stable, minimizing registry writes even in high-churn environments.

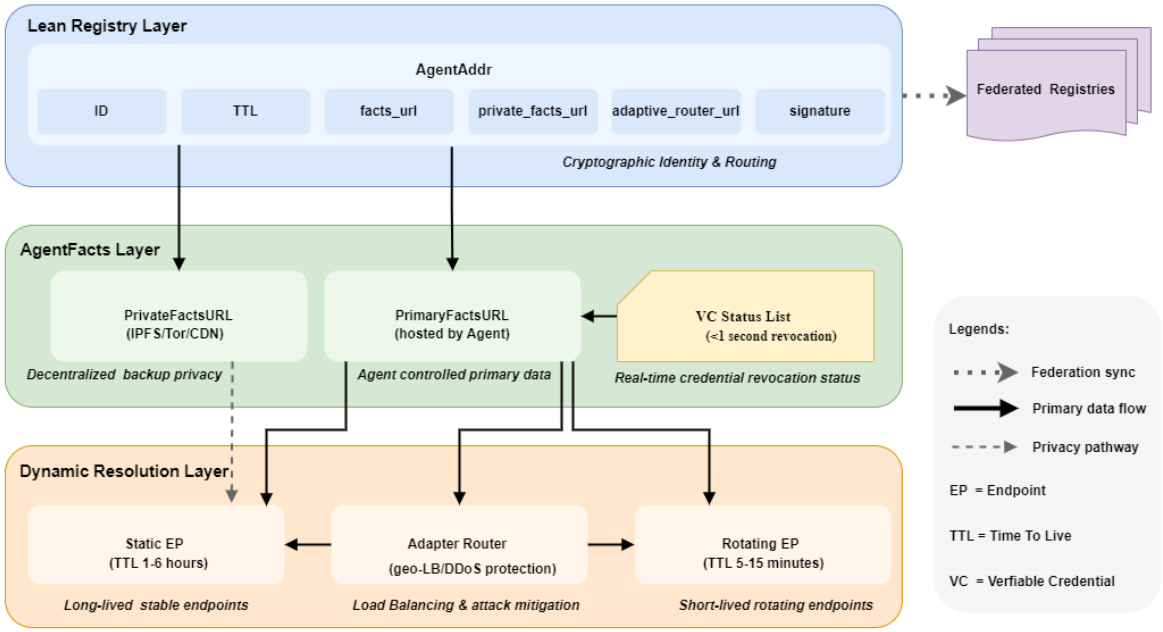

Figure 2: NANDA Quilt-like Registry Architecture: A modular, three-tier discovery model for AI agents. The **Registry Layer** resolves agent identifiers to signed `AgentAddr` records with metadata and routing pointers. The **AgentFacts Layer** hosts cryptographically signed JSON-LD documents describing dynamic agent state, capabilities, and endpoints. The **Dynamic Resolution Layer** supports privacy-preserving access and multi-endpoint routing, including static URIs, rotating pools, and adaptive routers. This separation of concerns enables scalable, privacy-aware agent discovery across federated domains. [source]

The full discovery flow is distributed across three layers:

1. **A Registry Layer** that serves as a static index of agent identifiers to metadata pointers.
2. **A Metadata Layer (AgentFacts)** containing schema-validated, W3C Verifiable Credential (VC)-signed descriptions of capabilities, endpoints, telemetry, and trust assertions.
3. **A Dynamic Resolution Layer** that supports static, rotating, or adaptive endpoint selection based on TTLs, load, geography, or policy.

6.1 Security

NANDA achieves security through end-to-end cryptographic guarantees. Each AgentAddr is signed by the issuing registry to ensure authenticity and immutability. AgentFacts documents are signed by trusted issuers using W3C Verifiable Credentials v2 and support short-lived credentials (<5 minutes) with revocation via VC-Status lists. This cryptographic framework ensures agents cannot spoof identities, impersonate others, or falsify capabilities. The inclusion of privacy-preserving resolution paths (PrivateFactsURL) further shields client access patterns from agents or intermediaries, in line with zero-trust design principles.

6.2 Authentication

Agent metadata is authenticated via decentralized identifiers (DIDs), and any claims within AgentFacts—such as skills, audit results, or compliance certifications—must be issued by credential authorities bound to trust domains. Updates to metadata are not pushed to the registry but are independently signed and hosted by agents or third-party infrastructure. This enables decentralized publication while maintaining verifiable authenticity.

The registry does not mediate live authentication but provides cryptographic roots of trust for downstream verification. Publishers are authenticated using DID-based credentials or through delegation models, supporting both self-sovereign and enterprise-controlled agent lifecycles.

6.3 Scalability

NANDA's architecture is inherently scalable due to its lean core and separation of concerns. By reducing registry entries to signed pointers and offloading metadata to self-hosted or distributed stores, the system avoids write amplification and supports horizontal scaling. TTL-based caching allows registry reads to be handled by edge caches or client-side resolvers, while AdaptiveRouterURL enables real-time endpoint agility without impacting the core registry.

The registry supports federated deployment, allowing multiple shards or namespaces to operate independently while remaining interoperable. Each registry instance may govern a sector (e.g., healthcare, finance) or geography, with cross-federation handled via DID resolution and verifiable claims.

6.4 Maintainability

The core registry logic is deliberately minimal and stable, reducing operational complexity. Since dynamic agent state is handled externally via signed AgentFacts, registry updates are rare and schema evolution does not require core service changes. This decoupling enables faster iteration on agent metadata without risking registry consistency.

All records adhere to versioned, JSON-LD-based schemas with forward compatibility built into the context definitions. Tooling around AgentFacts validation, routing policies, and credential inspection is modular and referenceable by external components, easing long-term ecosystem integration and maintainability.

Table 1: Comparison of AI Agent Registry solution by A2A, MCP and NANDA Quilt

Dimension	MCP	Google A2A	NANDA QUILT
Purpose	Enable publication and discovery of agent servers for LLM tool use	Advertise a server-side agent's endpoint and skills for JSON-RPC	Convey live endpoints plus cryptographically-signed capabilities, SBOM hash, privacy path, revocation info
Discovery Path	REST: `/v0/servers`, `/v0/servers/{id}`	One-step fetch at `/.well-known/agent.json` or central catalogue	Two-step: lean registry → `FactsURL` / `PrivateFactsURL`
Trust Primitive	Bearer token + GitHub credentials	Plain HTTPS and optional token, self-declared attributes	W3C Verifiable Credential v2 signatures; VC-Status revocation (<1 s), support for third-party audited attributes
Privacy Option	Container metadata and manifest publishing; optional offline operation	None (lookup hits agent host)	Optional `PrivateFactsURL` via IPFS/Tor; hides requester
Endpoint Freshness	Timestamps on publish: `created_at`, `updated_at`; dynamic via REST	Assumes minutes-to-hours stability (no TTL field)	TTL-scoped lists: static (1–6 h) or rotating (5–15 m)
Schema Weight	Approx. 1–3 KB JSON	$\approx$ 0.3–1 KB JSON	1–3 KB JSON-LD + VC
Best-fit Use Case	Centralized registry of agent/server tools (Docker/NPM); designed for scalable deployment and LLM plug-in integration	Stable SaaS agents inside a single marketplace	Highly mobile, privacy-sensitive, or safety-critical agents at trillion scale

7 Comparative Evaluation

Our comparative analysis is two-fold:

- Table 1 compares registry solutions from MCP, A2A (Google), and NANDA Quilt, analyzing their purpose, discovery paths, trust primitives, privacy mechanisms, endpoint freshness strategies, schema complexity, and best-fit use cases.
- Table 2 provides a detailed feature-level comparison between Agent Cards (used in A2A) and Agent Facts (used in NANDA Quilt), highlighting key differences—including but not limited to metadata structure, endpoint modeling, cryptographic guarantees, and extensibility.

Table 2: Detailed Feature Comparison: Agent Card vs. Agent Facts

Feature	Agent Card (Google A2A)	Agent Facts (NANDA Quilt)
Identifier	url (host-tied)	Id
Label & description	name, description, version	agent name, label, description, version
Provider Info	provider.organization, url	provider.{name,url,did}
Static endpoint	single url	`"endpoints.static"[ ]`
Rotating / geo endpoints	–	`"endpoints.rotating"[ ]` (URL + TTL)
Adapter router	–	`"endpoints.adaptive_router".url`
Skills / Capabilities	`skills[ ]`, `capabilities.*`	`skills[ ]` + `"capabilities"`
Auth Schemes	`securitySchemes`, `security`	`capabilities.authentication`
Protocol Flags	`capabilities.streaming`, `pushNotifications`	`capabilities.streaming`
SBOM/ Integrity	–	`sbomHash` (in VC)
Telemetry	–	`telemetry.{endpoint,sampling}`
Performance/audit	–	`evaluations block`
Privacy Path	–	`private_facts_url`
Cryptographic wrapper	plain JSON, HTTPS	JSON-LD + W3C VC
	optional	signature
TTL/Freshness	none (HTTP cache)	per-endpoint TTL fields
Size	0.3 – 1 KB	1–3 KB
Fetch hops	1 (well-known)	2 (registry → Facts)
Revocation speed	HTTP re-fetch (min)	VC-Status (<1 s)
Implementation effort	low	higher (JSON-LD, VC Toolchain)

8 Phase-wise Evolution of Agent Registry Architectures

Agent registry systems have evolved from simple, file-based descriptions to distributed, cryptographically-verifiable registries with structured discovery protocols. This section outlines the progression across thtree key phases, each adding layers of interoperability, scalability, trust, and governance. Understanding these phases helps clarify both the design motivations behind existing systems and the trade-offs made at each maturity level.

8.1 Phase I — Static, Isolated Discovery

The earliest registry mechanisms rely on static files (e.g., JSON or YAML manifests) published at well-known locations on an agent's domain. These files are primarily consumed manually or by tightly coupled runtimes and contain minimal, static metadata. Common attributes include agent name, endpoint, and basic capabilities.

Examples: Google A2A's `/.well-known/agent.json`,

8.2 Phase II — Dynamic RESTfull APIs

This phase introduced runtime introspection via HTTP APIs and formally validated JSON schemas. MCP has RESTfull API to find and list available MCP server in the client apps .

8.3 Phase III — Verifiable Metadata and Federated Trust and AI agent quilt

Registries in this phase adopt cryptographic verification and federated trust mechanisms closer to Nanda Quilt. Agent metadata is signed using W3C Verifiable Credentials (VCs), PKI certificates, or JSON Canonicalization with hashing and signing. DID-based identities or domain-bound signatures ensure authenticity, while TTL and revocation mechanisms enable fine-grained cache control. These designs enable trust portability, auditability, and agent-to-agent verification. They are well-suited for mobile, privacy-sensitive, or safety-critical deployments. This approach is closer to Nanda Quilt.

9 Conclusion

The proliferation of autonomous AI agents across enterprise, research, and consumer domains has created an urgent infrastructure challenge: how to discover, identify, and trust agents at Internet scale. This survey examined four representative registry architectures— MCP Metaregistry, A2A, Microsoft Entra Agent ID and NANDA Quilt Registry—each addressing core requirements of security, authentication, scalability, and maintenance through distinct architectural approaches.

Our analysis reveals several key insights that will shape the future of agent discovery infrastructure:

1. Architectural Trade-offs Are Protocol-Dependent. The "right" registry architecture depends heavily on deployment context. Enterprise environments with existing Azure AD infrastructure benefit from Entra Agent ID's seamless integration and zero-maintenance approach. Open research communities and decentralized applications require the cryptographic guarantees and federated governance of NANDA Registry. Protocol-specific ecosystems like MCP benefit from purpose-built registries that leverage existing authentication systems (GitHub OAuth) while maintaining simplicity.
2. Decentralization Enables Long-term Sustainability. While centralized approaches offer operational simplicity, they create single points of failure and vendor lock-in risks that become increasingly problematic as agent ecosystems mature. NANDA Registry's federated design demonstrate how decentralized architectures can achieve both scalability and community governance. The dual-path resolution pattern in NANDA particularly addresses privacy concerns that will become critical as agent interactions proliferate.
3. Security Must Be Built-in, Not Bolted-on. All examined registries recognize that cryptographic integrity is foundational—whether through W3C Verifiable Credentials (NANDA), DNS-TXT verification (MCP), or Azure AD's enterprise security controls (Entra). However, the separation of static identity resolution from dynamic metadata distribution, pioneered by NANDA, offers the most robust foundation for preventing registry poisoning and enabling privacy-preserving discovery patterns.
4. Interoperability Remains the Critical Gap. Despite their architectural differences, these registries serve overlapping use cases and will inevitably need to interoperate as agent ecosystems mature. Cross-protocol discovery, unified namespace management, and portable agent identities represent the next frontier for registry infrastructure development.
5. Community Governance Is Essential for Ecosystem Health. The most successful Internet-scale infrastructure—from DNS to HTTP to email—has emerged from open, multi-stakeholder governance processes. While proprietary solutions like Entra Agent ID serve important enterprise needs, the broader agent ecosystem requires community-governed registries that can evolve independently of any single vendor's interests.

References

[1] Model Context Protocol Contributors. Model Context Protocol Registry. `https://github.com/modelcontextprotocol/registry`, 2025. Accessed: 2025-05-29.

[2] Google A2A Project Contributors. A2A Specification. `https://google-a2a.github.io/A2A/specification/`, 2025. Accessed: 2025-05-29.

[3] Alex Simons. Announcing microsoft entra agent id: Secure and manage your ai agents. `https://techcommunity.microsoft.com/blog/microsoft-entra-blog/announcing-microsoft-entra-agent-id-secure-and-manage-your-ai-agents/3827392`, May 2025. Accessed: 2025-05-29.

[4] Project NANDA. Upgrade or switch: The need for new registry architecture for the internet of ai agents. `https://github.com/aidecentralized/nandapapers`, 2025. Accessed: 2025-05-29.

[5] Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp), 2025.