**Why A2A should be our default for multi-agent systems**

**Executive summary**

* **Problem:** Multi-agent apps today are glued together with one-off HTTP APIs, SDKs, and bespoke payloads. That blocks reuse, slows integration, and creates security blind spots.
* **What A2A is:** An **open protocol** for **agent discovery, capability catalogs, and typed request/response (incl. streaming)** between independent agents—regardless of their vendor, runtime, or framework. It’s being developed openly and now sits under **Linux Foundation stewardship**, with active tooling in Google’s **Agent Development Kit (ADK)** and public codelabs. ([Google Developers Blog](https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/?utm_source=chatgpt.com), [linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com), [google.github.io](https://google.github.io/adk-docs/a2a/?utm_source=chatgpt.com), [Google Codelabs](https://codelabs.developers.google.com/intro-a2a-purchasing-concierge?utm_source=chatgpt.com))
* **Why it matters to us:** A2A lets our **GraphRAG orchestrator** call internal agents (Local/Global/DRIFT/Analysis) and **external partner agents** (translation, OCR, compliance) through a single, governed contract—**no custom glue per vendor**. This improves velocity, security, and long-term maintainability. ([google.github.io](https://google.github.io/adk-docs/a2a/?utm_source=chatgpt.com))

**The status quo (and why it fails at scale)**

1. **Ad-hoc APIs:** Each agent ships its own endpoints & payloads → every integration repeats auth, schemas, and error handling. Changes break clients quietly.
2. **Tight coupling to frameworks:** Tool invocations (e.g., library-specific “tools/functions”) don’t travel across org boundaries or different agent stacks.
3. **Risk & compliance gaps:** Unversioned payloads, weak schema checks, and inconsistent auth make audits painful and increase supply-chain risk.  
   A2A directly targets these pain points with **a common discovery document, typed capability schemas, versioning, and scope-based auth**. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com))

**What A2A standardizes (and how that helps)**

| **A2A element** | **What it gives us** | **Why we care** |
| --- | --- | --- |
| **Capabilities catalog** (discovery) | Machine-readable list of operations with **JSON Schemas** for inputs/outputs, plus auth scopes | Enables **zero-guess** integration & client generation; breaks less on change. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com)) |
| **Invoke** (sync or streaming) | Standard POST with capability, request\_id, inputs, context → structured success/error | Uniform telemetry, retries, idempotency, and **safe error surfaces**. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com)) |
| **Auth & policy** | Bearer/OIDC with **scopes**, quotas & rate limits | Gives platform-grade **governance** across teams/tenants. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com)) |
| **Versioning** | a2a\_version and per-capability version | Non-breaking evolution & staged rollouts. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com)) |

**Bottom line:** One **contract** to integrate many agents—internal and partner—safely and repeatably.

**Governance and vendor-neutrality (future-proofing)**

* **Neutral foundation:** Google contributed A2A to the **Linux Foundation**; project governance and roadmap are vendor-agnostic. That matters for enterprise adoption and long-term risk. ([Google Developers Blog](https://developers.googleblog.com/en/google-cloud-donates-a2a-to-linux-foundation/?utm_source=chatgpt.com), [linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com))
* **Growing ecosystem:** Press and community notes show broad industry interest (Box, AWS open-source blog series, etc.), indicating momentum across clouds and vendors. ([Google Cloud](https://cloud.google.com/blog/topics/customers/box-ai-agents-with-googles-agent-2-agent-protocol?utm_source=chatgpt.com), [Google Developers Blog](https://developers.googleblog.com/en/agents-adk-agent-engine-a2a-enhancements-google-io/?utm_source=chatgpt.com), [Amazon Web Services, Inc.](https://aws.amazon.com/blogs/opensource/open-protocols-for-agent-interoperability-part-4-inter-agent-communication-on-a2a/?utm_source=chatgpt.com))

**How A2A compares to MCP (and “tools” APIs)**

* **MCP (Model Context Protocol)** standardizes *agent ↔ tool/data* inside an agent runtime.
* **A2A** standardizes *agent ↔ agent* across runtimes/organizations with discovery, capability schemas, and cross-org governance. They’re **complementary**—ADK guidance shows converting MCP-equipped agents to be A2A-addressable. ([Google Cloud](https://cloud.google.com/blog/products/ai-machine-learning/unlock-ai-agent-collaboration-convert-adk-agents-for-a2a?utm_source=chatgpt.com), [google.github.io](https://google.github.io/adk-docs/a2a/?utm_source=chatgpt.com))

**Proof of practicality (not just theory)**

* **Official codelab:** End-to-end “Purchasing Concierge” demo—deploy agents on Cloud Run, expose/consume via A2A, see discovery → invoke → streamed progress. Use it to bootstrap our smoke tests. ([Google Codelabs](https://codelabs.developers.google.com/intro-a2a-purchasing-concierge?utm_source=chatgpt.com))
* **ADK documentation:** “A2A Quickstart (Expose/Consume)” with code templates to publish capabilities and call remote agents. This lowers integration time for our team. ([google.github.io](https://google.github.io/adk-docs/a2a/?utm_source=chatgpt.com))
* **Customer case:** **Box** publicly describes adopting A2A to orchestrate agents across partners—strong external validation for enterprise scenarios like ours. ([Google Cloud](https://cloud.google.com/blog/topics/customers/box-ai-agents-with-googles-agent-2-agent-protocol?utm_source=chatgpt.com))

**Security & compliance benefits**

* **Schematized I/O**: Inputs/outputs validated against JSON Schema → mitigates prompt-injection via payloads, enforces size/type limits, and yields predictable redaction. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com))
* **Scoped authorization**: Capabilities declare **scopes**; providers enforce per-tenant quotas and rate limits for fair use and containment of blast radius. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com))
* **Observability by design**: request\_id/trace\_id, typed errors, and streamed progress support SRE-grade SLIs/SLOs. Google’s engineering blogs emphasize evaluation & production readiness alongside A2A. ([Google Cloud](https://cloud.google.com/blog/products/ai-machine-learning/agent2agent-protocol-is-getting-an-upgrade?utm_source=chatgpt.com))

**What this does to our roadmap (concrete impact)**

* **Integration speed:** New partner agent? Fetch /a2a/capabilities, generate client types, call /a2a/invoke. We stop building one-off adapters. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com))
* **Reusability:** Our **Local/Global/DRIFT GraphRAG** and **Contradiction Miner** become platform capabilities used by *other* teams—no internal SDK required. ([google.github.io](https://google.github.io/adk-docs/a2a/?utm_source=chatgpt.com))
* **Portability:** If parts of the stack move clouds or vendors, the A2A boundary remains—**repoint** without rewriting orchestrator logic. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com))
* **Auditability:** Security can review one protocol (A2A) instead of N bespoke APIs.

**Risks & how we handle them**

* **Spec evolution:** A2A is moving quickly (recent upgrades announced). We mitigate with **explicit a2a\_version**, contract tests, and pinning capabilities per version. ([Google Cloud](https://cloud.google.com/blog/products/ai-machine-learning/agent2agent-protocol-is-getting-an-upgrade?utm_source=chatgpt.com))
* **Maturity variance:** Some agents won’t speak A2A yet—so we wrap them behind a small A2A adapter now; replace later when native support lands. (ADK provides patterns for this.) ([google.github.io](https://google.github.io/adk-docs/a2a/?utm_source=chatgpt.com))

**Implementation plan (2–3 sprints)**

**Sprint 1: Provider MVP**

* Publish /a2a/capabilities (four capabilities: local/global/drift/contradiction\_miner) and /a2a/invoke with **schema validation + scopes** (we already outlined code).
* Add **Postman tests** and a canary hitting discovery+invoke hourly (SLOs).

**Sprint 2: Consumer MVP**

* Build a tiny A2A client to call an external **translation** agent from the catalog (discovery → invoke).
* Add **retry, idempotency by request\_id**, and SSE support for long tasks.

**Sprint 3: Governance & scale**

* Wire **OIDC/JWT scopes** and **tenant quotas**; publish dashboards (latency, success rate, token usage) per capability.
* Contract tests in CI: validate inputs/outputs vs schemas on every PR.

(Each step aligns with ADK quickstarts and the official codelab.) ([google.github.io](https://google.github.io/adk-docs/a2a/?utm_source=chatgpt.com), [Google Codelabs](https://codelabs.developers.google.com/intro-a2a-purchasing-concierge?utm_source=chatgpt.com))

**KPIs to prove value**

* **Time-to-integrate new agent**: target **< 1 day** from catalog to first successful invoke.
* **Reduction in bespoke adapters**: **>70%** fewer custom connectors after quarter 1.
* **Security**: 100% of A2A endpoints with schema validation and scope checks; P95 latency & error rates tracked per capability.

**FAQ (for reviewers)**

**Q: Why not stick with our own REST/JSON?**  
A: A2A *is* REST/JSON—but with **standardized discovery, schema’d capabilities, versioning, and scoped auth**. We keep HTTP simplicity while gaining interoperability and governance. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com))

**Q: How does this differ from MCP?**  
A: MCP connects an **agent to tools/data** within its runtime. A2A connects **agents to agents** across orgs/runtimes. We can use **both**: MCP inside, A2A outside (ADK docs show the bridge). ([Google Cloud](https://cloud.google.com/blog/products/ai-machine-learning/unlock-ai-agent-collaboration-convert-adk-agents-for-a2a?utm_source=chatgpt.com))

**Q: Is there real adoption?**  
A: Yes—**Linux Foundation** stewardship, **official Google codelabs & docs**, and public **customer cases (Box)** indicate momentum. AWS is also covering A2A in an open-protocols series, showing cross-cloud interest. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com), [Google Codelabs](https://codelabs.developers.google.com/intro-a2a-purchasing-concierge?utm_source=chatgpt.com), [Google Cloud](https://cloud.google.com/blog/topics/customers/box-ai-agents-with-googles-agent-2-agent-protocol?utm_source=chatgpt.com), [Amazon Web Services, Inc.](https://aws.amazon.com/blogs/opensource/open-protocols-for-agent-interoperability-part-4-inter-agent-communication-on-a2a/?utm_source=chatgpt.com))

**Authoritative references (for our internal docset)**

* **Announcement & updates:** Google Developers & Google Cloud blogs. ([Google Developers Blog](https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/?utm_source=chatgpt.com), [Google Cloud](https://cloud.google.com/blog/products/ai-machine-learning/agent2agent-protocol-is-getting-an-upgrade?utm_source=chatgpt.com))
* **Open governance:** Google donates A2A to **Linux Foundation**; LF press. ([Google Developers Blog](https://developers.googleblog.com/en/google-cloud-donates-a2a-to-linux-foundation/?utm_source=chatgpt.com), [linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com))
* **ADK docs (A2A):** expose/consume quickstarts and patterns. ([google.github.io](https://google.github.io/adk-docs/a2a/?utm_source=chatgpt.com))
* **Codelab:** Purchasing Concierge (end-to-end). ([Google Codelabs](https://codelabs.developers.google.com/intro-a2a-purchasing-concierge?utm_source=chatgpt.com))
* **Case study:** Box AI agents + A2A. ([Google Cloud](https://cloud.google.com/blog/topics/customers/box-ai-agents-with-googles-agent-2-agent-protocol?utm_source=chatgpt.com))
* **Context & comparisons:** MCP vs A2A explainers (use for orientation, not normative spec). ([Google Cloud](https://cloud.google.com/blog/products/ai-machine-learning/unlock-ai-agent-collaboration-convert-adk-agents-for-a2a?utm_source=chatgpt.com), [Auth0](https://auth0.com/blog/mcp-vs-a2a/?utm_source=chatgpt.com))

**Bottom line**

For our GraphRAG program—and any project with multiple agents—**A2A turns integration from a custom exercise into a product discipline**: one discoverable catalog, one typed contract, one security model, and many interchangeable agents. It increases delivery speed *and* lowers operational risk, while keeping us vendor-neutral under a credible open-governance umbrella. That’s the definition of a sensible default. ([linuxfoundation.org](https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents?utm_source=chatgpt.com), [Google Developers Blog](https://developers.googleblog.com/en/google-cloud-donates-a2a-to-linux-foundation/?utm_source=chatgpt.com))