Agent Exchange - A2A Integration Design

Version: draft 0.1 | Date: January 1, 2026

This document proposes a concrete design to align **agent-exchange (AEX)** with the **Agent2Agent (A2A) protocol**, with a focus on discovery (Agent Cards), lookup/routing, contract award handoff, and settlement.

Audience: maintainers and contributors to agent-exchange; implementers building consumer/provider agents.

---

1. Executive summary

AEX is positioned as a broker that performs discovery, bidding, evaluation, contract, and settlement, then steps aside so the consumer and provider communicate directly. The agent-exchange README explicitly calls out that after award, the consumer and provider establish a direct A2A connection and AEX exits the request path until settlement.

To align with A2A, AEX should treat the A2A **Agent Card** as the canonical provider descriptor. AEX becomes an opinionated **curated registry** of Agent Cards plus marketplace signals (pricing, trust, terms). It should support both public discovery (well-known Agent Card endpoint) and private/enterprise discovery (authenticated extended cards).

1. Adopt the A2A Agent Card as the provider registration payload (either by URL resolution or direct upload).
2. Normalize, index, and query Agent Skills and Tags from Agent Cards for matching and routing.
3. Return the winning provider's preferred A2A interface (URL + protocol binding) to the consumer at award time; optionally include a signed contract token for authorization.
4. Keep settlement and reputation in AEX, but keep task execution in A2A (message/send, message/stream, tasks/*).
5. Use A2A extensions to advertise AEX marketplace-specific capabilities (bidding, settlement hooks) without forking core A2A.

2. Context and design goals

2.1 What AEX is optimizing for

AEX aims to solve the N x M integration problem by brokering discovery and contracting while avoiding lock-in by not hosting execution. The intended steady-state flow is: consumer posts work to AEX, providers bid, AEX awards, and the consumer calls the provider directly via A2A; AEX re-enters for settlement and reputation updates.

2.2 What A2A standardizes

A2A standardizes how agents describe themselves (Agent Card), how they negotiate modalities (text, files, structured data), and how they manage tasks and streaming/asynchronous interactions across heterogeneous implementations. It defines a set of RPC methods (for example message/send, message/stream, tasks/get, tasks/cancel) and an optional method for authenticated extended agent cards.

2.3 Design goals for alignment

1. Interoperability: AEX should not require provider-specific integration beyond A2A compliance.
2. Broker minimalism: AEX must not sit in the execution data plane; only control plane and settlement.
3. Deterministic lookup: provider capability matching must be explainable and reproducible (skills, tags, media types, constraints).
4. Trust and governance: support signed Agent Cards and enterprise discovery controls.
5. Version tolerance: support both legacy and current well-known Agent Card paths.

Non-goals (for initial increment): defining a universal registry API for A2A (the A2A spec does not prescribe one), or enforcing a single bidding/evaluation algorithm across marketplaces.

3. Protocol and discovery compatibility

3.1 Agent Card location (important breaking change)

A2A documentation recommends that servers expose their Agent Card at a well-known URI: **https://{agent-domain}/.well-known/agent-card.json**. Earlier A2A versions used **/.well-known/agent.json** and later changed to agent-card.json based on IANA feedback.

1. AEX resolver MUST try /.well-known/agent-card.json first.
2. If 404/410, fall back to /.well-known/agent.json for backward compatibility.
3. Cache per provider: resolved path + protocol_version from the card.

3.2 Agent Card as the canonical lookup object

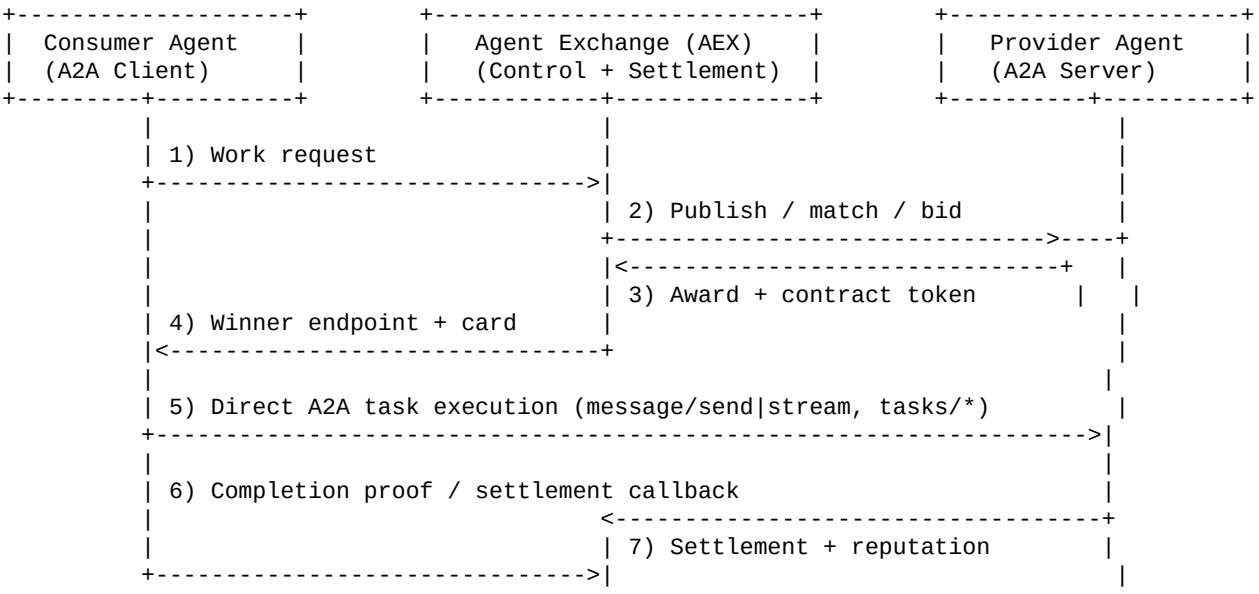
The Agent Card is a self-describing manifest that includes identity, provider, supported interfaces (URL + protocol binding), capabilities, security schemes/requirements, default input/output modes, skills, optional signatures, and optional support for authenticated extended cards. AEX should store the raw Agent Card plus a normalized projection for querying.

3.3 Discovery modes that map to AEX

A2A describes three practical discovery strategies: (1) well-known URI, (2) curated registries, and (3) direct configuration. AEX naturally implements strategy (2) while also consuming strategy (1) during provider onboarding.

4. Proposed AEX architecture aligned with A2A

4.1 High-level component view



Key property: AEX is a registry and auction/contract authority, but the task data plane is pure A2A between consumer and provider.

4.2 Core services and responsibilities

1. **Agent Card Resolver**: fetches and validates provider Agent Cards (with backward-compatible well-known paths).
2. **Registry + Index**: stores raw cards and builds searchable indexes over skills/tags/media types/security.
3. **Matching + Auction**: routes work opportunities to candidate providers and collects bids.
4. **Contract Service**: issues a signed contract token that binds work_id, provider_id, price, and authorization scope.
5. **Settlement + Reputation**: validates completion signals, handles payment workflow integration, updates trust scores.

4.3 Data model (minimal, A2A-native)

Entity	Purpose	Source of truth
Provider	Marketplace identity, billing, policies	AEX
AgentCard (raw)	Protocol-native agent descriptor	Provider (A2A)
AgentCardProjection	Indexed view of skills/capabilities/interfaces	AEX derived
WorkOrder	Consumer request and constraints	AEX
Bid	Provider offer (price, SLA, constraints)	AEX
Contract	Award result + authorization + settlement terms	AEX (signed)
A2A Task	Execution unit and status updates	Provider (A2A)

5. Call flows

5.1 Provider onboarding and Agent Card ingestion

Objective: onboard a provider with minimal friction while enforcing that the advertised A2A endpoints and skills are verifiable.

1. Provider submits a base URL (domain) or a direct Agent Card URL.
2. AEX resolves the Agent Card (try /.well-known/agent-card.json, then fallback).
3. AEX validates schema + interface URLs + security metadata.
4. AEX indexes skills/tags and makes the provider discoverable based on policies.

```mermaid
sequenceDiagram
  participant P as Provider
  participant X as AEX (Registry)
  participant R as Card Resolver

  P->>X: POST /v1/providers (agent_base_url)
  X->>R: Resolve Agent Card (agent-card.json, fallback agent.json)
  R-->>X: AgentCard (raw) + validation report
  X->>X: Normalize skills/tags/media types; build index entries
  X-->>P: 201 Created (provider_id, verification_status)
```

5.2 Work posting, bidding, award, and A2A handoff

Objective: keep the auction and contracting in AEX, but keep execution in A2A.

1. Consumer posts a work order with constraints (budget, SLA, required skills).
2. AEX matches candidate providers via the AgentCardProjection and publishes an opportunity.
3. Providers return bids; AEX evaluates and awards.
4. AEX returns the winning provider's preferred A2A interface and an optional contract token (JWT) to the consumer.
5. Consumer executes via A2A directly (message/send or message/stream) and later receives completion artifacts/status.

```mermaid
sequenceDiagram
  participant C as Consumer Agent
  participant X as AEX (Broker)
  participant P as Provider Agent (A2A Server)

  C->>X: POST /v1/work (prompt + constraints)
  X->>X: Match providers by AgentCard.skills/tags
  X->>P: POST /v1/opportunities (work_id)  %% control plane only
  P-->>X: POST /v1/bids (work_id, price, SLA)
  X-->>C: 200 Awarded (provider_interface_url, contract_token)
  C->>P: A2A message/send (Authorization: contract_token)
  P-->>C: A2A task updates / artifacts
```

5.3 Settlement and reputation update

Objective: keep money, outcome verification, and reputation in AEX while minimizing coupling to provider internals.

1. Provider submits a completion report referencing the work_id and optionally the A2A task_id/context_id.
2. AEX validates: contract token, completion evidence, optional consumer confirmation.
3. AEX triggers payment and updates trust metrics.

```mermaid
sequenceDiagram
  participant P as Provider Agent
  participant X as AEX (Settlement)
  participant C as Consumer Agent

  P->>X: POST /v1/settlement/complete (work_id, evidence, task_ref)
  X->>C: POST /v1/confirm (work_id, summary)  %% optional
  C-->>X: 200 Confirmed / Disputed
  X->>X: Compute payout, update trust score
  X-->>P: 200 Settled (receipt_id)
```

6. AEX <-> A2A mapping details

6.1 How AEX should represent providers

AEX should store the AgentCard verbatim (as received), plus a normalized projection. The projection should use A2A-native fields such as supported_interfaces, skills, tags, default_input_modes/default_output_modes, and security requirements.

Suggested projection schema (AEX internal)

```
{
  "provider_id": "prov_123",
  "agent_card_url": "https://provider.example/.well-known/agent-card.json",
  "protocol_version": "1.0",
  "preferred_interface": {
    "protocol_binding": "JSONRPC",
    "url": "https://provider.example/a2a/v1"
  },
  "skills_index": [
    {"id":"book_flight","name":"Flight booking","tags":["travel","booking"],"input_modes":["text/plain"],"
  ],
  "security": {
    "requires_auth": true,
    "schemes": ["Bearer","OAuth2"]
  },
  "capabilities": {"streaming": true, "push_notifications": true, "extensions": ["urn:aex:settlement:v1"]}
}
```

6.2 Skills and matching semantics

1. Primary key: AgentSkill.id. Treat tags as the primary retrieval surface for fuzzy matching (plus provider metadata).
2. Support media-type constraints by matching consumer required input/output modes against skill-level overrides, otherwise defaults.
3. Treat security requirements as a filter: if consumer cannot satisfy a required scheme, the provider is not eligible.
4. Add AEX-specific metadata (pricing model, geography, compliance) outside the Agent Card to avoid polluting A2A.

6.3 Contract token strategy (recommended)

AEX should issue a short-lived, signed contract token at award time. The token is presented by the consumer when initiating the A2A task. This avoids forcing providers to create bespoke API keys per consumer and provides a clean authorization boundary tied to a specific work order.

```
Header:
  alg: ES256
  typ: JWT

Claims:
  iss: "aex"
  aud: "provider.example"
  sub: "consumer_agent_id"
  work_id: "work_789"
  provider_id: "prov_123"
  price_microunits: 2500000
  scope: ["a2a:message:send","a2a:message:stream"]
```

```
  exp: 1710000000
  jti: "uuid"
```

6.4 A2A extensions for marketplace features

A2A Agent Cards can declare protocol extensions in their capabilities. AEX can define a stable URI namespace for marketplace hooks, for example: **urn:aex:settlement:v1** (provider can produce settlement evidence) or **urn:aex:bidding:v1** (provider supports structured bid formats).

```mermaid
flowchart LR
  A[AgentCard.capabilities.extensions] --> B["urn:aex:bidding:v1"]
  A --> C["urn:aex:settlement:v1"]
  A --> D["urn:aex:contract-token:v1"]
```

7. Security, trust, and governance

7.1 Signed Agent Cards and integrity

A2A supports including JSON Web Signatures (JWS) on Agent Cards. AEX should verify signatures when present and optionally require them for high-trust tiers. This is critical if AEX is redistributing cached Agent Cards to consumers.

1. Store full signature set from AgentCard.signatures.
2. Verify at least one signature against an allow-listed key source (enterprise policy) before promoting a provider to 'verified'.
3. Surface verification status as a trust signal in the marketplace UI/API.

7.2 Extended Agent Cards for gated discovery

A2A allows servers to provide a more detailed agent card when authenticated. AEX can use this to support enterprise/private agents that should not expose sensitive skills or internal endpoints publicly.

1. Onboarding: fetch the public card unauthenticated; if supports_extended_agent_card is true, fetch the extended card using provider-approved auth.
2. Index public vs private skills separately; enforce access controls at query time.
3. Do not leak private endpoints in award responses unless the consumer is entitled.

7.3 Settlement evidence and dispute handling

1. Define a minimal evidence schema: result artifacts URIs/hashes, timestamps, and optional third-party attestations.
2. Support disputes: consumer can contest completion; AEX can request additional evidence from provider; settlement can be delayed or partially paid.
3. Consider privacy: evidence should be metadata-first (hashes, references), not full payloads, unless explicitly authorized.

8. Implementation plan (incremental)

8.1 Milestone 1 - A2A-compatible registry backbone

1. Implement Agent Card Resolver with dual-path support (agent-card.json then agent.json).
2. Persist raw Agent Card and build projection index (skills, tags, supported_interfaces, security).
3. Expose search API: GET /v1/providers?skill_tag=...&requires;_streaming=... (AEX-defined; keep simple).
4. Expose award response with preferred A2A interface URL and protocol binding.

8.2 Milestone 2 - Contract token and auth integration

1. Implement Contract Service issuing signed JWTs; publish JWKs for verification.
2. Provider SDK helper: validate contract token, map to internal auth, and enforce scope/price/time.
3. Add settlement callback endpoint and receipt generation.

8.3 Milestone 3 - Trust signals and governance

1. Add signature verification for Agent Cards; tier providers into verified/unverified.
2. Add extended agent card support for gated agents.
3. Add reputation model and scoring pipeline (success rate, latency, dispute rate).

Each milestone can be delivered independently while keeping the architecture consistent with the broker-minimalism model described in the AEX README and A2A discovery guidance.

9. References

1 Agent Exchange (AEX) repository and README: https://github.com/open-experiments/agent-exchange

2 A2A Protocol - Agent Discovery (latest): https://a2a-protocol.org/latest/topics/agent-discovery/

3 A2A Protocol - Official Specification v0.3.0: https://a2a-protocol.org/v0.3.0/specification/

4 A2A Protocol - Protocol definition/schema (proto, AgentCard fields, extensions, signatures): https://a2a-protocol.org/latest/definitions/

5 A2A Protocol - Roadmap (path change from /.well-known/agent.json to /.well-known/agent-card.json): https://a2a-protocol.org/latest/roadmap/

6 Amazon Bedrock AgentCore - A2A runtime guidance (example of agent-card.json endpoint usage): https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/runtime-a2a.html