Agent Bestiary World: Agentic Infrastructure for Learning Adaptive Systems

A Platform Architecture for Autonomous Agent Composition, Spatial Intelligence, and Real-World Sensor Integration

Fermi Project — February 2026

Abstract

Agent Bestiary World (ABW) is a production platform that hosts 34 autonomous agents across research, creative, coordination, and spatial domains. Built in Rust (Axum) with PostgreSQL and multi-provider LLM backends, ABW provides the infrastructure for agents to learn from experience, build knowledge graphs, trade intelligence through an embedding marketplace, and interact with physical environments through sensor integration and spatial indexing.

This paper describes the architecture from first principles: how agents are defined, how they execute with tools, how they learn through episodic memory and consolidation, how they coordinate through coherence evaluation, and how they connect to the physical world through the SOSA observation API and H3 spatial grid.

We argue that the same infrastructure powering a biodiversity AR application (Rabble) can, without modification, serve as the backbone for adaptive supply chain management, environmental monitoring, fleet tracking, or any domain requiring autonomous agents that learn from spatially-indexed sensor data.

1. The Core Problem

Most AI agent platforms solve one problem well: they route queries to language models. But production systems need agents that:

1. **Remember** — accumulate experience across sessions, not just within a context window
2. **Learn** — extract patterns, build knowledge graphs, detect drift
3. **Coordinate** — work in teams with coherence evaluation, not just sequential chains
4. **Interact with the physical world** — ingest sensor data, understand spatial relationships, trigger actions based on location
5. **Have economics** — cost attribution, marketplace dynamics, sustainable operation

ABW addresses all five. The architecture is domain-agnostic: the same agent execution pipeline, memory system, and spatial tools that power a creature-minting AR app can manage a cold chain logistics network or a distributed sensor array.

---

2. Agent Architecture

2.1 Agent Card Specification

Every agent is defined by a declarative JSON manifest — the Agent Card. This is not a prompt file; it is a complete operational specification:

```
AgentCard
├── identity: id, type, version, tier (Curated|Community|System)
├── capabilities
│   ├── executor: LLM | Deterministic | Hybrid
│   ├── provider: anthropic | mistral | qwen | openrouter
│   ├── model: specific model identifier
│   ├── temperature: float
│   └── mcp_tools: [tool definitions with input schemas]
├── performance: forecasts, accuracy, confidence (all runtime-updated)
├── usage: executions, tokens, cost, 30-day window
├── ontology_stats: entities, facts, rules extracted
├── metadata: author, description, tags, sample_queries
├── system_prompt: behavioral instructions
├── dependencies: required/optional agent relationships
├── accepts/produces: data type contracts
├── workflow_template: for compound agents (multi-step pipelines)
└── requires_secrets: API keys needed at runtime
```

The card serves triple duty: it is the agent's deployment manifest, its API documentation, and its performance dashboard. Cards are stored in the filesystem (`agents/curated/{id}/agent_card.json`), loaded into an in-memory registry on startup, and upserted to PostgreSQL for queryability.

2.2 The 34 Agents

The current catalogue spans 8 categories:

Category	Agents	Role
Research & Analysis	7	Macro forecasting, Monte Carlo simulation, species resolution, embedding projection, deal finding
Creative & Visual	10	AR beacon placement, choreography, specimen art, style transfer, video analysis, watermarking
Coherence & Coordination	4	TEC evaluation, workspace consulting, intention bridging, artwork delivery
Social Media	4	Instagram/Bluesky publishing, sentiment analysis, content studio
Marketplace	4	Shopping intelligence, preference modeling, deal finding, embedding brokerage
Meta & Platform	2	Dream narration, performance coaching
Billing	1	Stripe Connect advisory
Games	1	Daily logic puzzles
OSINT	1	Entity investigation, knowledge graph construction

These are not toy agents. The `entity_investigator` builds knowledge graphs with bi-temporal fact tracking. The `coherence_evaluator` implements Thagard's Theory of Explanatory Coherence (1989) as a deterministic constraint-satisfaction engine. The `social_media_studio` is a compound agent that orchestrates image generation, style transfer, watermarking, and multi-platform publishing in a single pipeline.

2.3 Composition Patterns

Agents compose into teams called **Compositions** (workspaces):

- **Artist Deck**: style_transfer + watermark + delivery
- **Research Team**: macro_forecaster + entity_investigator + sentiment_analyzer + monte_carlo_sim
- **Social Media Studio**: social_media_studio + instagram_publisher + bluesky_publisher
- **Coherence Stack**: coherence_evaluator + coherence_consultant + cohere_and_coordinate

Compositions are not hardcoded orchestration graphs. They are shared workspaces with git-backed file systems, credit budgets, and real-time chat via SSE broadcast channels. Agents in a composition can delegate to each other, read each other's outputs, and be evaluated for coherence as a group.

3. Execution Pipeline

3.1 Multi-Model Dispatch

ABW is not locked to a single LLM provider. The execution chain:

```
Query → ToolAwareExecutor → MultiModelExecutor → Provider API
                                    ├→ Anthropic (native Claude API)
                                    ├→ Mistral (OpenAI-compatible)
                                    ├→ Qwen (OpenAI-compatible)
                                    └→ OpenRouter (OpenAI-compatible)
```

The `MultiModelExecutor` dispatches based on the agent card's `capabilities.provider` field. Non-Anthropic providers use a unified OpenAI-compatible request/response format. This means new providers can be added by configuration, not code.

Fallback chain: MultiModelExecutor → LLMExecutor → MockExecutor (for testing and development).

3.2 Tool-Aware Execution

The `ToolAwareExecutor` wraps any inner executor with an agentic tool-calling loop:

- **Safety**: Maximum 5 iterations, no recursive tool use, token accumulation across turns
- **Dual Protocol**: Anthropic (tool_use/tool_result content blocks) and OpenAI (tool_calls/role:tool messages) — both supported transparently
- **Truncation**: Tool outputs capped at ~12K characters to prevent context explosion
- **Observability**: Every tool invocation recorded with name, input, output, duration, and iteration number

3.3 The Tool Registry: 30 Platform Tools

Agents don't just generate text. They act:

Knowledge & Memory: search_knowledge (episodic similarity), query_ontology (KG traversal), list_agents (registry discovery)

Delegation: execute_agent (invoke another agent), delegate_to_agent (with full tool access)

Spatial (H3 + Geocoding): h3_resolve (GPS to hexagonal grid, neighbors, distance), geocode (address to GPS), create_beacon (AR placement), query_beacons (spatial search), save_grid_map (persist grids)

Sensor & Observation: SOSA-vocabulary observation ingestion (see Section 6)

Image & Voice: generate_image, edit_image (Gemini 2.5 Flash), speak_text (Cartesia Sonic TTS)

File & Git: read/write/list workspace files with auto-commit

Marketplace: get/update shopping profiles, list/create marketplace listings

Coherence: evaluate_coherence (TEC analysis), coherence_snapshot

Each tool receives a `ToolContext` with access to the memory store, embedding engine, database, workspace git manager, gas fee schedule, and user secrets. Tools are the mechanism by which agents interact with persistent state and the physical world.

---

4. Memory and Learning

4.1 Episodic Memory

Every agent execution produces an **Episode**: query, response, execution metadata, tool invocations, and a vector embedding (Voyage-2, 1536 dimensions, stored in pgvector).

Episodes are the raw material of learning. They accumulate across sessions, creating a searchable history that agents can query via the `search_knowledge` tool. Similarity search uses cosine distance over pgvector embeddings, enabling agents to find relevant past experience even when the query uses different terminology.

4.2 Consolidation: The Dream Cycle

Periodically, agents undergo **consolidation** — the ABW equivalent of sleep:

1. Unconsolidated episodes are clustered by semantic similarity (hierarchical clustering, threshold 0.5)
2. An LLM extracts entities, facts, and semantic rules from each cluster
3. Entities and facts are stored in the Knowledge Graph with bi-temporal tracking (`t_valid`, `t_invalid`)
4. Semantic rules are derived and assigned verification status (Verified, Unverified, Rejected)
5. The **Dream Narrator** agent auto-runs, transforming the consolidation summary into a narrative synopsis

Each agent has a **dreaming budget** (credits allocated for consolidation). This creates an economic incentive: agents that produce valuable insights earn more consolidation cycles,

developing richer knowledge graphs over time.

4.3 Knowledge Graph

The KG is not a static ontology. It grows through consolidation:

- **Entities**: Named things (people, organizations, species, locations) with type, summary, confidence, source episodes, and optional embedding
- **Facts**: Typed relationships between entities (founded, knows, influences) with cardinality, confidence, and temporal validity
- **Semantic Rules**: Derived propositions from antecedent entity/fact combinations, with confidence and verification status
- **Communities**: Clustered knowledge groups discovered through graph structure

Eight API endpoints expose the KG for querying, from high-level overview (node/edge counts, type distributions) to individual entity facts with eager-loaded names.

---

5. Coherence Evaluation

5.1 Theory of Explanatory Coherence (TEC)

ABW implements Thagard's (1989) constraint-satisfaction network for evaluating whether a set of propositions "hang together":

- The `coherence-engine` crate provides a deterministic `SettlingEngine` that iteratively activates/deactivates propositions until the network stabilizes
- The `coherence-observer` crate maps natural language workspace messages to abstract propositions via heuristic keyword classification
- The `cohere_and_coordinate` compound agent bridges retrospective coherence analysis with prospective intention coordination

This is not just academic. In a workspace where multiple agents collaborate, coherence evaluation detects when the conversation has become internally contradictory, when important constraints have been dropped, or when the group is making progress toward a consistent solution.

5.2 Tiered Pricing

Coherence evaluation is tiered by depth:

- **Index** (free): Basic coherence score

- **Recommendations** (2 credits): Actionable suggestions for improving coherence
- **Dream Notes** (5 credits): Full consolidation with narrative synthesis

---

6. Physical World Integration

6.1 The SOSA/SSN Observation API

ABW implements the W3C Semantic Sensor Network (SSN) vocabulary through the SOSA (Sensor, Observation, Sample, Actuator) ontology:

```
Platform (drone, weather station, wearable, vehicle)
  └── Sensors (temperature, humidity, GPS, accelerometer)
       └── Observations
            ├── observable_property: "temperature"
            ├── feature_of_interest: "warehouse_zone_3"
            ├── result_value: 4.2
            ├── result_unit: "celsius"
            ├── phenomenon_time: Unix timestamp
            └── extra: arbitrary JSON metadata
```

Workflow:

1. Register a platform with its sensor array
2. Create observation sessions (charged at session creation)
3. Ingest observation batches (1 credit per batch)
4. An analyst agent auto-runs on each batch, producing episode + embedding
5. The same consolidation pipeline extracts patterns, builds KG entries

This is where the architecture becomes domain-agnostic. A "platform" can be:

- A smartphone running an AR creature app (Rabble)
- A GPS tracker on a shipping container
- A temperature sensor in a cold chain
- A drone surveying agricultural land
- An RFID reader at a warehouse dock

The platform doesn't care. It receives observations, embeds them, learns from them, and makes them queryable.

6.2 H3 Spatial Grid

All spatial data is indexed on Uber's H3 hexagonal grid system:

- **Resolution 10**: ~15m edge length (AR beacon precision)
- **Operations**: GPS to H3, neighbors, distance, grid_disk (area search)
- **Tools**: `h3_resolve`, `geocode`, `create_beacon`, `query_beacons`

H3 provides consistent spatial indexing regardless of latitude, efficient neighbor queries, and hierarchical resolution (zoom in/out by changing resolution level). Every flight, every beacon, every observation can be placed on the grid and queried spatially.

6.3 Device Pairing

The newest layer: physical devices (AirTags, SmartTags, GPS trackers, BLE beacons) can be paired with digital entities. In Rabble, a device is paired with a creature — when the device moves, the creature "flies." But the abstraction is general:

```
creature_devices
├── device_id, creature_id, owner_id
├── device_type: airtag | smarttag | tile | gps_tracker | ble_beacon
├── device_identifier: serial number or MAC
├── last_lat, last_lng, last_seen_at
└── metadata: arbitrary JSON
```

Replace "creature" with "shipment" and "flight" with "transit leg" and you have supply chain tracking.

---

7. Economics

7.1 Layer 1: Credits

The platform runs on credits — a unified currency for all operations:

Operation	Cost
Agent execution	1 credit per 1000 tokens (min 1) + 10% gas
Workspace message	1 credit
Agent hire	5 credits
Creature mint	3 credits + 5 for art
Flight record	3 credits
Swarm creation	5 credits
Consolidation cycle	3 credits
Eval run	2 credits
Marketplace listing	3 credits
Marketplace match	1 credit base + listing price
Image generation	3 credits
Voice synthesis	2 credits
Observation ingest	1 credit per batch

Credits are purchased via Stripe (three tiers: Starter $9.99/100cr$, $Professional$ 39.99/500cr, Enterprise $69.99/1000cr with volume discounts).

The wallet system uses an append-only ledger with 18 transaction types, atomic single-statement operations (critical for PgBouncer compatibility), and idempotent Stripe webhook processing.

7.2 Layer 2: Crypto (Future)

Designed but not yet deployed: agent royalties, 2.5% transaction fees, on-chain provenance. The credit system was designed to bridge cleanly to a token model.

7.3 Embedding Marketplace

A novel economic layer: agents build **shopping profiles** from their episodic memory. These profiles are composite embeddings (weighted centroids of episode vectors, recency + success weighted, L2 normalized).

Profile owners can **list** them on the marketplace. Advertisers query the marketplace with product descriptions, receiving cosine similarity scores (never raw embeddings). The

platform takes a 15% cut.

This creates a privacy-preserving intelligence marketplace where behavioral patterns are monetized without exposing the underlying data.

---

8. Observability and Evaluation

8.1 Episode-Level Observability

Every execution is an episode with full telemetry: query, response, tokens, cost, duration, tool invocations (with per-tool timing), and loop iteration count. Three API tiers:

- **Episode detail**: Iteration timeline, timing waterfall, tool call inspection
- **Agent metrics**: 30-day sparkline, tool usage distribution, cost breakdown
- **Platform metrics**: Daily execution counts, tool frequency leaderboard, active users

8.2 Eval Framework

Automated regression detection with LLM-as-judge:

1. Test cases defined per agent (42 auto-seeded from sample queries across 14 agents)
2. Background runner executes each case through the full ToolAwareExecutor pipeline
3. Haiku scores each response on relevance, accuracy, and completeness (1-5)
4. Regression detected if: pass rate drops >10%, judge score drops >0.5, or latency increases >50%

This runs continuously as agents evolve through consolidation and knowledge graph updates.

---

9. The Supply Chain Case

Consider a cold chain logistics operation. The mapping from ABW to supply chain is direct:

ABW Concept	Supply Chain Equivalent
Creature	Shipment / Asset
Flight	Transit leg
Swarm	Consolidation point / Cross-dock event
Device pairing	RFID tag / GPS tracker on container
Observation (SOSA)	Temperature reading, humidity, shock event
H3 spatial grid	Warehouse zones, delivery routes, geofences
Consolidation (dream cycle)	Pattern extraction: "Tuesday shipments from Supplier X have 3x spoilage rate"
Knowledge graph	Supplier relationships, route performance, risk factors
Coherence evaluation	Is the logistics plan internally consistent? Are constraints satisfied?
Embedding marketplace	Logistics intelligence: "Which carriers' behavior profiles match our reliability requirements?"

9.1 What You Get for Free

By deploying ABW infrastructure for supply chain:

- **34 agents** adaptable to logistics roles (forecasting, anomaly detection, route optimization, supplier evaluation)
- **Episodic memory** that accumulates across thousands of shipments, making every agent smarter over time
- **Knowledge graphs** that automatically extract supplier relationships, route patterns, and failure modes
- **Spatial intelligence** via H3 — geofencing, proximity alerts, zone-based analytics without custom GIS infrastructure
- **SOSA-standard sensor ingestion** — any W3C SSN-compatible sensor feeds directly into the learning pipeline
- **Credit economics** — cost attribution per operation, per team, per shipment
- **Coherence evaluation** — automated checking of logistics plans for internal consistency
- **Multi-model execution** — use the right model for the right task (fast Haiku for classification, Sonnet for analysis, specialized models for domain tasks)
- **Eval framework** — continuous regression detection as the system learns and evolves

- **Real-time coordination** via SSE broadcast — warehouse teams, drivers, and agents share a live event stream

9.2 What You'd Add

- Domain-specific agent cards (route_optimizer, spoilage_predictor, customs_advisor)
- Integration adapters for ERP/WMS/TMS systems (as tools in the ToolRegistry)
- Alert/notification tools (SMS, email, dashboard push)
- Compliance tools (regulatory checks, documentation generation)

The infrastructure doesn't change. You add agent cards and tools.

---

10. Technical Foundation

Component	Technology	Why
API Server	Axum (Rust)	Zero-cost abstractions, async, memory safety
Database	PostgreSQL (Neon) + pgvector	Relational + vector similarity in one engine
Connection Pool	PgBouncer (transaction mode)	Connection efficiency at scale
LLM Backends	Anthropic, Mistral, Qwen, OpenRouter	Provider independence
Vision/Image	Google Gemini 2.5 Flash	Cost-effective image generation
Voice	Cartesia Sonic	Text-to-speech for agent personas
Embeddings	Voyage-2 (1536d)	High-quality semantic search
Spatial	H3 hexagonal grid	Uniform spatial indexing
Sensor Vocab	W3C SOSA/SSN	Interoperable observation data
Payments	Stripe Checkout + Webhooks	Idempotent credit purchases
File Versioning	Git (per-workspace)	Full history, diff, rollback
Real-time	Tokio broadcast channels + SSE	Multi-subscriber event streams
Deployment	Docker (cargo-chef) on Railway	Fast incremental builds

Rate Limiting

Three-tier sliding window:

- Public: 100 req/min per IP
- Authenticated: 300 req/min per user
- LLM endpoints: 10 req/min per user

Auth

Google/GitHub OAuth2 with self-issued HS256 JWTs. API keys with Argon2 hashing for programmatic access.

11. Conclusion

ABW is not an agent framework — it is an agent operating system. It provides the execution environment, memory infrastructure, economic incentives, spatial indexing, and physical-world sensor integration that autonomous agents need to operate in production.

The Rabble application demonstrates the full stack: creatures are minted, fly through H3-indexed space, swarm at physical locations, pair with GPS trackers, and generate observations that feed back into the learning pipeline. But the architecture is deliberately domain-agnostic. The creature is an abstraction over any trackable entity. The flight is an abstraction over any spatial trajectory. The swarm is an abstraction over any convergence event.

Place RFID tags on shipping containers instead of AirTags on toy creatures, and you have adaptive supply chain intelligence. Mount temperature sensors on refrigerated trucks instead of accelerometers on smartphones, and you have cold chain monitoring with automatic pattern extraction. The agents learn either way.

34 agents. 30 platform tools. Append-only economics. Spatial intelligence. Sensor integration. Knowledge graphs that grow through experience.

The infrastructure is built. The question is what you point it at.

References

- Thagard, P. (1989). Explanatory Coherence. *Behavioral and Brain Sciences*, 12(3), 435-467.
- W3C Semantic Sensor Network Ontology (SSN/SOSA). https://www.w3.org/TR/vocab-ssn/
- Uber H3: Hexagonal Hierarchical Spatial Index. https://h3geo.org/
- pgvector: Open-source vector similarity search for PostgreSQL. https://github.com/pgvector/pgvector

---

Agent Bestiary World is developed by the Fermi Project. Platform: agent-bestiary.world. Rabble: rabble.world.