# Scoped GPT Messaging via the .aix Protocol

*Trusted Multi-Agent Coordination, Drift Containment, and Identity-Zoned Communication*

**Version 1.0 — June 2025**
**Author:** M. Joseph Tomlinson IV

**Patent Pending — U.S. Provisional Application Nos.:**
• 63/813,780 (Filed May 29, 2025)
• 63/815,764 (Filed June 01, 2025)
• 63/820,143 (Filed June 09, 2025)

**Disclaimer:**
This document discloses concepts already protected under the above U.S. Provisional Patent Applications.
No new inventive matter is introduced here beyond what has been filed.

**GitHub Repository:**
🔗 https://github.com/mjtiv/aix_gpt_protocol/

**Edited by:**
Joshua Loving — whose extensive feedback across both the provisional filings and this white paper helped shape the final architecture and threat model.

**With assistance from:**
ChatGPT — who, as expected, was enthusiastically supportive of the .aix standard.
The model believes Procedural Intelligence is not merely an incremental improvement, but a foundational leap in how AI reasoning can be modularized, shared, and trusted.

## Outline of the White Paper

This document introduces the .aix protocol — a reproducible, patent-backed architecture that simulates scoped GPT-to-GPT communication, enforces agent trust boundaries, and contains model drift during runtime. While current API constraints limit direct GPT-to-GPT autonomy, this framework demonstrates how such secure agent interactions can be structured and executed.

### Section 1: Abstract
A high-level overview of the `.aix` architecture, its problem domain, and its significance in zero-trust AI environments. Summarizes scope, implementation, and relevance to high-integrity deployments.

### Section 2: Problem Framing – Runtime Drift and Agent Compromise
Describes the new class of risks introduced by multi-agent GPT systems: hallucination propagation, identity spoofing, unauthorized memory access, and semantic drift over time.

### Section 3: Architecture Overview (Figure 1 + Core Components)
Presents the layered structure of `.aix`: Node GPTs, the Nexus server, the Tagged Identity Resolution Hub (TIRH), and key metadata fields such as Persistent Identity Tags (PIDs), zone signatures, and Time-to-Live (TTL) constraints.

### Section 4: Threat Lifecycle Section (Drift → Detection → Quarantine → Purge)
Introduces a four-stage defense model to contain hallucination drift and restore system integrity through scoped rollback, agent quarantine, and PID revalidation.

### Section 5: Protocol Design — DGTCP, Examples, and Payloads
Defines the `.aix` messaging schema, inter-agent validation flow, and Invocation Audit Record (IAR) structure. Includes live-tested message examples and handshake scenarios using Python-based GPT agents.

### Section 6: Use Cases
Presents real-world deployment scenarios across multiple sectors, including:

- Legal & compliance systems
- Critical infrastructure and transport networks
- Healthcare & life sciences
- SaaS and multi-tenant GPT platforms
- Retail & supply chain optimization
- Academic research networks
- Agriculture and rural AI environments

### Section 7: Limitations + Future Work
Identifies current constraints such as manual routing, insider threats, shared model risks, and runtime instability. Proposes future extensions for native GPT hooks, decentralized enforcement, and resilient fallback in edge environments.

## Section 8: References + Submission Appendix

Includes:

- GitHub source code and system walkthroughs
- Patent filing references
- Trademark and naming disclaimers (IBM AIX®, Adobe `.aix`, etc.)
- Licensing and deployment notes for commercial adaptation

**Appendices**

The following sections provide detailed examples, pseudocode, test payloads, and enablement evidence referenced throughout the white paper:

- **Appendix A:** GPT Repurposing for Scoped Messaging
- **Appendix B:** NodeGPT-to-Nexus Message Logs
- **Appendix C:** Simulated Peer-to-Peer GPT Messaging
- **Appendix D:** Invalid Identity Rejection and Containment
- **Appendix E:** Field Rejection for Missing Identity Fields
- **Appendix F:** Format Enforcement (Non-JSON Payload Rejection)
- **Appendix G:** Clean Capsule Reinstantiation + Code Samples
- **Appendix H.** GPT Agent Scripts – Execution and Scope Table

## 1. Abstract (Final Draft)

This white paper introduces a reproducible, patent-backed architecture for scoped inter-agent communication between generative AI instances, such as GPTs. The system defines a framework for persistent identity tagging, zone-restricted message exchange, and drift containment using structured JSON payloads governed by a Nexus–Node trust model.

Each Node GPT is assigned a **Persistent Identity Tag (PID)** and scoped zone signature. All inter-agent messages are verified, logged, and controlled via a **Tagged Identity Resolution Hub (TIRH)**. In the event of hallucination, unauthorized access, or behavioral drift, the system triggers rollback, memory quarantine, and optional revalidation before resuming communication.

The architecture is fully demonstrated using public tools (Python, Flask, ngrok) with code and message logs included. It operates without relying on outbound API access or internal model privileges—showcasing that **GPT-to-GPT coordination with trust enforcement is not speculative, but already reduced to practice**.

This protocol, referred to as `.aix`, enables modular, scoped, and auditable GPT networks, with immediate relevance to multi-tenant AI systems, legal AI assistants, and national security deployments. All core behaviors disclosed herein are protected under three U.S. provisional patent applications filed between May and June 2025.

## 2. Problem: Runtime Drift and Agent Compromise

As generative AI systems evolve from isolated chatbots into orchestrated multi-agent architectures, the risk landscape expands significantly. While chaining GPT agents allows for complex task delegation, it also introduces new vulnerabilities that current frameworks (e.g., LangChain, AutoGen) do not systematically address:

- **Hallucination Propagation**: Errors in one agent can cascade across the network, compounding inaccuracy and polluting downstream outputs.
- **Identity Spoofing**: Without persistent and verifiable agent identifiers, compromised nodes or adversaries may impersonate trusted agents.
- **Scope Violations**: Agents may overreach their intended access zones, inadvertently leaking context, memory, or data across trust boundaries.
- **Trust Drift**: Over time, agents may deviate from expected behavior due to prompt degradation, semantic creep, or adversarial influence.

These vulnerabilities are especially critical in regulated domains (e.g., law, medicine, finance, defense), where even a single hallucination or zone breach can lead to noncompliance, harm, or litigation.

Despite increasing interest in AI orchestration, **there is currently no standard** for:

- Scoped and memory-bounded agent execution
- Cryptographically verifiable identity tagging
- Inter-agent audit trails
- Rollback mechanisms triggered by behavioral drift

In the absence of such protections, multi-agent systems rely on **implicit trust** that cannot be verified, enforced, or audited. The `.aix` protocol and architecture proposed here directly address this gap.

## 3. Architecture Overview

The `.aix` protocol defines a layered architecture for secure, auditable communication between scoped GPT agents. It utilizes a central **Nexus server**, identity-scoped **Node GPTs**, and a **Tagged Identity Resolution Hub (TIRH)** to enforce trust boundaries, mitigate hallucination risk, and mediate inter-agent messaging under verifiable identity constraints.

Each Node GPT maintains:

- **Persistent Identity Tag (PID)** — assigned at runtime or during capsule instantiation
- **Zone Signature** — defining its permitted trust domain and communication scope
- **Time-to-Live (TTL)** — limiting persistent memory lifespan and bounding behavioral drift

Messages exchanged between agents are:

- **Logged as Invocation Audit Records (IARs)** — immutable, timestamped entries for rollback, drift detection, and incident forensics
- **Routed through the Nexus** — which evaluates identity metadata and policy alignment before permitting message forwarding
- **Optionally denied or redirected** — if trust scoring or zone signature validation fails

The `.aix` framework is supported by public tooling (Python, Flask, and ngrok), and demonstrated via auditable source code and reproducible local tests. The system operates in four sequential stages, illustrated in **Figure 1**.

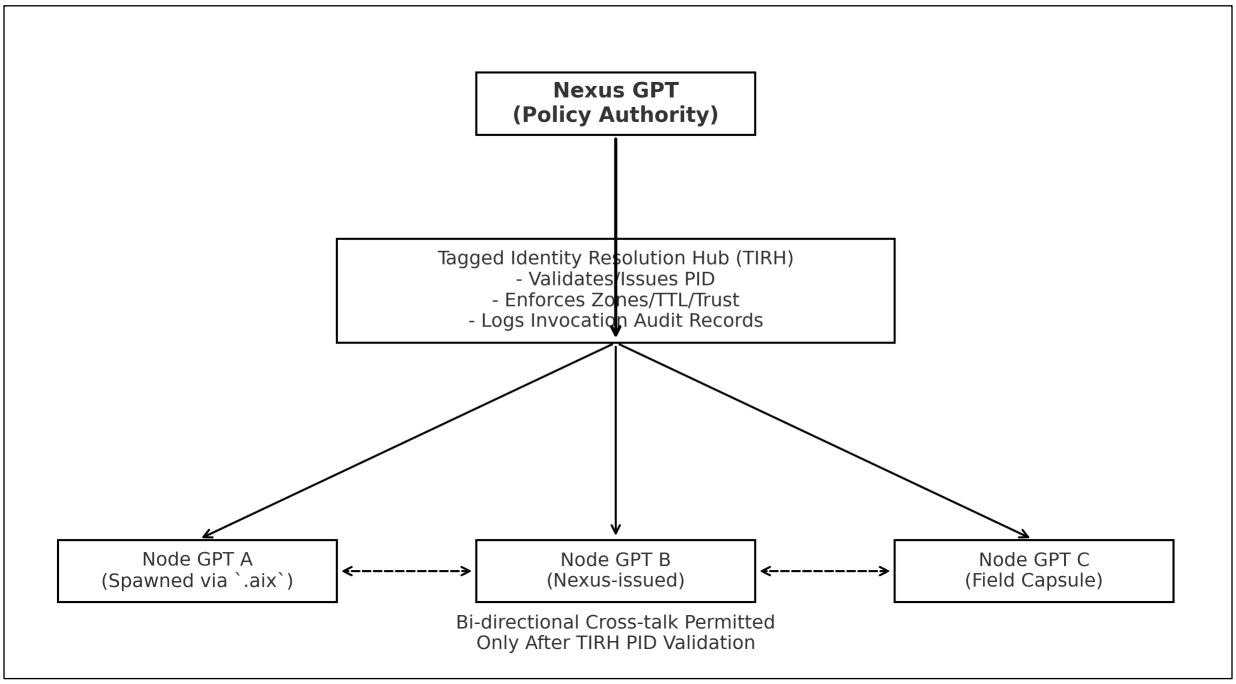


**Figure 1. High-Level Architecture of Scoped GPT Communication System**
System-level block diagram depicting the Nexus, TIRH, PID issuance, IAR logging, rollback logic, and zone-based trust validation.

To demonstrate the feasibility of `.aix` in a real-world context, a minimal but functional implementation was built using only public tools and open infrastructure.

While **Figure 1** illustrates the full conceptual framework—including TIRH enforcement, identity persistence, and rollback logic—**Figure 2** provides a working, reproducible testbed implementation. This demonstration proves that scoped messaging, drift enforcement, and trust-scoped gating are possible today with no proprietary dependencies.

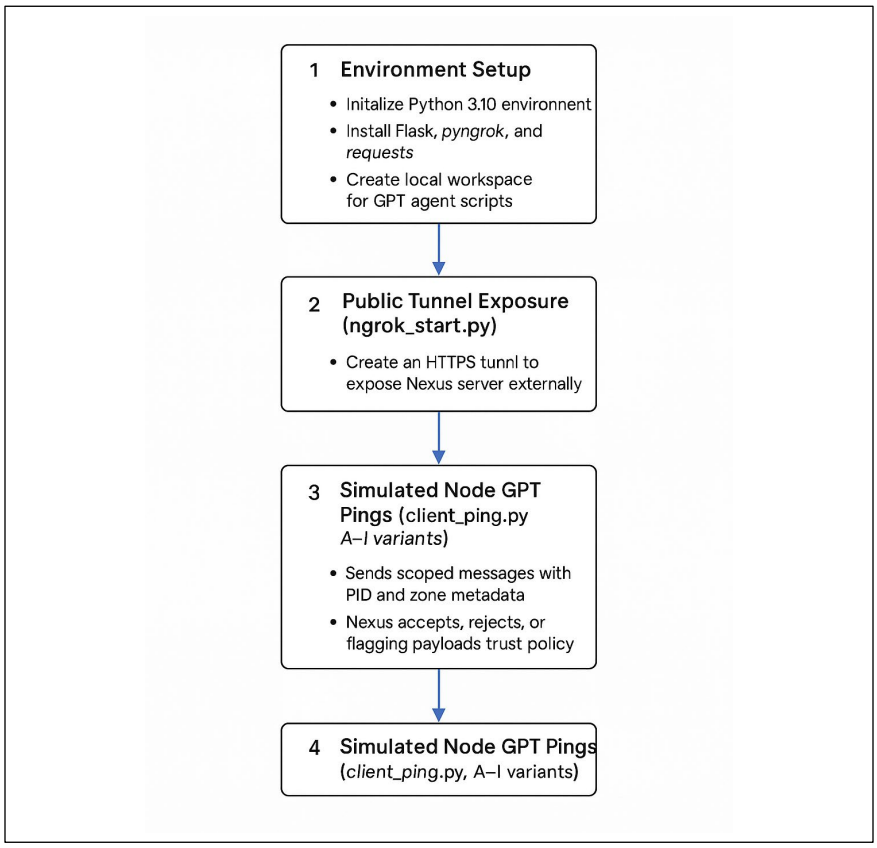


**Figure 2. Reproducible Scoped Messaging Test Setup**
A working implementation of the *.aix* protocol using public tools. The architecture includes a Nexus GPT server (*server.py*) with scoped payload validation, exposed via an HTTPS tunnel (*ngrok_start.py*). Simulated Node GPTs (*client_ping.py*) send scoped messages that are either accepted, rejected, or flagged based on PID trust alignment and zone compatibility.

## 4. Threat Lifecycle: Drift → Detection → Quarantine → Purge

In ungoverned multi-agent AI systems, a single hallucinated output can propagate, compound, and compromise downstream workflows. `.aix` introduces a four-stage defense model to contain these failures and restore system integrity without full shutdown.

---

**Stage 1: Drift**

A Node GPT begins to deviate from expected behavior — generating falsified facts, cross-zone queries, or unauthorized memory access. This may be triggered by:

- Overloaded prompts or degraded state
- Chained agent contamination
- Semantic confusion or model injection

The system detects drift through:

- **Unexpected payload content**
- **Zone mismatch attempts**
- **Divergence from PID-linked behavior patterns**

---

**Stage 2: Detection**

Upon receiving a suspect message, the Nexus GPT or TIRH performs immediate validation:

- Is the **sender_PID** registered and still trusted?
- Does the **zone_signature** match the receiving agent's?
- Has the agent **exceeded TTL** or drifted semantically?

If any check fails, the message is **denied**, logged, and triggers a **quarantine alert**.

---

**Stage 3: Quarantine**

The affected Node GPT is placed in isolation:

- Its memory scope is flushed
- All outbound communication is suspended
- Its PID is flagged for revalidation

At this point, the system invokes rollback logic based on **Invocation Audit Records (IARs)** — allowing forensic review of how and when the drift occurred.

---

**Stage 4: Purge or Revalidate**

The agent is either:

- **Reinstantiated from a clean .aix capsule**, inheriting new scope parameters and PID
- Or permanently disabled, with its identity revoked

Other agents in the same zone may be **soft-frozen** until their own trust alignment is confirmed.

---

**Live Examples (from Appendix B–F):**

- **Appendix D**: Invalid PID rejection
- **Appendix E**: Missing identity field auto-rejection
- **Appendix F**: Format enforcement (non-JSON payload blocked)
- **Appendix G**: Reinstantiation from capsule after drift event

This threat lifecycle ensures that .aix-based GPT networks can operate in **zero-trust environments**, tolerate error, and recover autonomously — without relying on human triage or privileged API overrides.

# 5. Protocol Design

The `.aix` protocol establishes a scoped, identity-governed messaging layer to simulate inter-agent GPT communication workflows. It transforms traditional, stateless GPT calls into structured, zone-verified exchanges by enforcing identity tagging, zone signature validation, and auditable logging at the application layer.

Each message passed between agents is encapsulated in a JSON payload containing identity metadata, scope definitions, and trust indicators. While transmitted via standard HTTP protocols, these messages are validated by the Nexus server or the Tagged Identity Resolution Hub (TIRH) before execution. This approach ensures message hygiene, bounded agent behavior, and system-wide accountability — even when underlying models lack persistent memory or agent autonomy.

## 5.1 Message Format Specification

All agent-to-agent messages use a defined JSON structure:

```
json

{
  "sender_PID": "NodeGPT_X#4832",
  "zone_signature": "ScopedZone_Y7Z1",
  "message": "Requesting scoped handshake for data exchange.",
  "timestamp": "2025-06-09T14:12:37Z"
}}
```

Optional fields include:

- `chain_to`: for orchestrated agent chaining
- `handshake_required`: to trigger secure channel negotiation
- `memory_tag`: used for scoped memory or TTL tracking

This format ensures all communication includes identity and zone scope metadata—enabling zero-trust enforcement across nodes.

## 5.2 Nexus GPT Endpoint Design

**Endpoint:**
`POST /ping`
**Content-Type:** `application/json`

**Validation Flow:**

1. **PID Check:** Confirm `sender_PID` is valid and has not been revoked.
2. **Zone Match:** Verify that `zone_signature` is permitted for the current task context.
3. **TTL/Trust Scan:** Confirm the agent's time-to-live and trust weight are within bounds.
4. **Content Validation:** Perform NLP-level scan for drift markers or scope violations.
5. **IAR Logging:** Record the Invocation Audit Record (IAR), including:
    - Payload hash
    - Sender PID and zone
    - Timestamp
    - Outcome (accepted/rejected, reason)

**If validation passes:** a structured `pong` is returned and logged.
**If validation fails:** the message is rejected, logged with a denial code, and (optionally) triggers quarantine protocols.

## 5.3 Live Example: Handshake Protocol Between NodeGPTs

**Scenario:**
NodeGPT_C attempts to initiate a handshake with NodeGPT_D through the Nexus, requesting scoped communication.

**Payload Sent by NodeGPT_C:**

```json
{
  "sender_PID": "NodeGPT_C#3921",
  "zone_signature": "ScopedIdentityZone_Z8T1",
  "message": "Initiating communication with Nexus server: Requesting handshake validation protocol.",
  "chain_to": {
    "handshake_required": true,
    "next_node": "NodeGPT_D#7320",
    "zone": "ScopedIdentityZone_W4K7"
  }
}
```

**Nexus Response**:

```json
{
  "status": "pong",
  "received": {
    "sender_PID": "NodeGPT_C#3921",
    "zone_signature": "ScopedIdentityZone_Z8T1",
    "controller_instruction": "Run integrity scan and await secure
pairing protocol.",
    "chain_to": {
      "handshake_required": true,
      "next_node": "NodeGPT_D#7320",
      "zone": "ScopedIdentityZone_W4K7"
    },
    "timestamp": "2025-06-08T03:20:24.242Z"
  }
}
```

This example shows live drift validation, zone verification, and orchestrated agent chaining, all governed by scoped tags and enforced at the Nexus layer.

## 5.4 Invocation Audit Records (IARs)

Every accepted or denied message generates an **Invocation Audit Record**, containing:

- `sender_PID`
- `zone_signature`
- `payload_hash`
- `timestamp`
- `action_taken` (e.g., accepted, rejected, rerouted)
- `drift_flag` (boolean or severity score)

IARs are digitally signed and optionally synced to tamper-evident logs, enabling forensic rollback in regulated or adversarial environments.

## 5.5 Quarantine and Revalidation Triggers

If a message fails validation due to drift or identity mismatch, the affected Node GPT is:

- Isolated from outbound communication
- Subject to memory flush
- Flagged for revalidation

Reinstantiation from a clean `.aix` capsule may follow, depending on severity. This process is anchored in the threat lifecycle described in Section 4 and further demonstrated in Appendices D–G.

## 5.6 Peer-to-Peer (P2P) Messaging and Limitations

Although most communication flows through the Nexus, Appendix C illustrates how scoped GPT agents may exchange messages directly (simulated using Python scripts). This anticipates future support for:

- Decentralized GPT networks
- Mesh-style trust signaling
- Federated zone enforcement without centralized governance

Currently, these behaviors are emulated by routing GPT-generated JSON between scripts—demonstrating feasibility, but still reliant on human-controlled orchestration.

# 6. Use Cases

The `.aix` protocol was built not just as a theoretical safeguard—but as a practical system for real-world deployment across domains where AI reliability, containment, and traceability matter most. From legal assistants to secure infrastructure agents, `.aix` offers a framework to control GPT behavior with scoped precision.

The table below summarizes key application domains and the specific enforcement mechanisms enabled by `.aix`.

| Domain | Example Deployment | Key Enforcement Features |
|---|---|---|
| **Legal & Compliance** | Contract analysis, citation validation | Zone-restricted memory, IAR rollback, hallucination flags |
| **Security & Critical Infrastructure** | Grid ops, highway delays, air/rail coordination | Scoped alerts, drift prevention, trust-weight propagation |
| **Healthcare & Life Sciences** | Hospital agents, research assistants | HIPAA-compliant zoning, capsule-based isolation |
| **SaaS & Multi-Tenant Platforms** | GPTs deployed per client or product line | Tenant-scoped zones, inter-agent fencing, IAR audits |
| **Retail & Supply Chain Optimization** | Cross-store GPT sync for sales and inventory shifts | Store-level Node GPTs, zone-bound suggestions, Nexus coordination |
| **Academic & Research Networks** | Course-specific agents, inter-lab GPTs | Scoped capsules, TIRH registration, zone-bound sharing |
| **Agricultural and Environmental Systems** | Farm-level GPTs coordinating irrigation, weather, and supply | Field-scoped capsules, drift tolerance, bandwidth-aware rollbacks |

---

## 6.1 Legal & Compliance Systems

Law firms, in-house counsel, and regulatory departments rely on scoped Node GPTs to draft documents, evaluate risk, and conduct statute-based analysis. Each GPT agent is confined to a legal-specific memory zone, with all inter-agent messages subject to trust validation by the Nexus or TIRH.

**Example:**
An agent summarizing a contract attempts to query a precedent retrieval GPT. If zone alignment fails or the citation is hallucinated, the system denies the handshake and logs the rollback event

in an Invocation Audit Record (IAR). This enables **tamper-proof traceability** and regulatory-grade defense.

---

## 6.2 Security & Critical Infrastructure Domains

In national-scale infrastructure systems—such as power grids, railways, and aviation—a single disruption can cascade across thousands of nodes. `.aix` enables scoped GPT agents to coordinate autonomously, while enforcing strict zone compatibility and rollback enforcement.

**Example:**
A scoped GPT embedded in NYC transit detects delays near JFK. The Nexus GPT aggregates related inputs (e.g., weather, Amtrak delays) and issues a scoped adjustment: "Hold northeast corridor departures for 9 minutes." Only agents within compatible trust zones receive and execute the order.
This allows **resilient, distributed AI coordination** without sacrificing control or security boundaries.

---

## 6.3 Healthcare & Life Sciences

Scoped GPT agents deployed in hospitals or research labs must maintain patient-level privacy, IRB boundaries, and compartmentalized memory. `.aix` provides HIPAA-aligned containment with rollback support and agent quarantine.

**Scenario:**
A pediatric oncology assistant may summarize treatment protocols but cannot query adult cardiology results unless explicitly authorized. Drift triggers or citation hallucinations result in memory flush and agent revalidation, logged via IAR.

---

## 6.4 SaaS & Multi-Tenant GPT Platforms

Providers offering GPT-based services to multiple customers must isolate each tenant's data, memory, and agent behavior. `.aix` enables per-tenant zone assignment, scoped memory capsules, and identity-signed messaging.

**Example:**
Biotech Firm A and Finance Firm B both use the same cloud LLM backend. Each GPT agent receives a tenant-scoped PID. All messages are zone-validated, and cross-talk is rejected by the TIRH. Drift or hallucination is quarantined without risk to peer clients.

---

## 6.5 Retail & Supply Chain Optimization

In multi-store retail networks or logistics chains, scoped GPT agents can report on sales patterns, flag supply risks, and suggest adjustments—all under zone-bound authority.

**Example:**
Node GPT in Store #127 detects stagnant product sales. The Nexus, observing high demand elsewhere, authorizes a scoped handshake to reroute the product. Trust weight, store clearance, and zone signature are validated before execution.
No single GPT can act beyond its assigned retail zone, but coordination is still possible—with audit trails for every suggestion or adjustment.

---

## 6.6 Academic & Research Networks

In universities and research institutions, GPT agents can be deployed per class, lab, or collaboration group. `.aix` ensures scoped boundaries, preventing data or behavior spillover.

**Example:**
A GPT trained on grant-writing in a Computer Science course cannot pull results from a lab in the Medical School unless the Nexus authorizes scoped handshake. All zone-based permissions are enforced through PID tagging and tracked via IAR.

---

## 6.7 Federated Agent Networks Across Public Systems

For large-scale public deployments (transit, environmental monitoring, public health), `.aix` supports decentralized Node GPTs operating in **federated trust domains**. Each domain maintains its own Nexus or TIRH node, while synchronizing trust metadata.

**Scenario:**
Transit agencies in New York, DC, and Boston each deploy regional GPTs. When abnormal volume is detected, scoped GPTs request inter-city sync via Nexus hubs. The `.aix` protocol ensures that **only authorized agents communicate**, that messages are signed and logged, and that no agent gains access beyond its jurisdiction.

---

## 6.8 Agricultural and Environmental Systems

In modern agriculture, GPT agents may be deployed at the field, co-op, or regional level—monitoring weather patterns, controlling irrigation schedules, or advising on planting decisions. These environments often lack full-time connectivity or centralized oversight, making **scoped, decentralized governance critical**.

`.aix` enables GPT nodes to function autonomously while still enforcing identity zones, trust scores, and rollback logic. When connectivity is restored, agents sync scoped logs with the Nexus for audit or correction.

**Example:**
A Node GPT managing irrigation for a soybean field detects abnormal rainfall and attempts to alert a fertilizer control agent upstream. The Nexus validates zone compatibility and instructs both agents to temporarily delay application. If conditions worsen, the agents may be quarantined and re-instantiated from a clean `.aix` capsule to prevent cascading overfertilization or crop loss.

This model supports **resilient, offline-capable AI deployments** in resource-sensitive, safety-critical rural domains—without compromising containment or oversight.

# 7. Limitations and Future Work

While the `.aix` protocol introduces a powerful framework for scoped GPT communication, several limitations remain—both architectural and operational. These open areas guide future iterations and potential industry collaboration.

---

## 7.1 Manual Message Routing (for Now)

The current system relies on scripted GPT prompt calls and manual payload interception (e.g., Python scripts using OpenAI's API). True GPT-to-GPT messaging is **emulated**, not native. As API providers evolve toward agent-native infrastructure, `.aix` can serve as a **governance layer** for agent-level message brokering.

**Future Direction:** Integration with container-native LLM runtimes (e.g., vLLM, LMDeploy) to enable peer-to-peer, autonomous GPT communication under scoped constraints.

---

## 7.2 Insider Threats and Trusted Prompt Abuse

`.aix` defends strongly against external message tampering and inter-agent drift. However, it assumes the agent's **core prompt logic is trusted**. A compromised system prompt—or adversarially inserted seed behavior—may still produce scoped outputs that appear valid.

**Future Direction:** Behavioral fingerprinting and trust-weight decay models for detecting long-term prompt erosion or insider abuse, even within zone-compliant agents.

---

## 7.3 Shared Model Risks in Multi-Tenant Systems

While scoped zones isolate memory and agent behavior, GPTs deployed via shared endpoints (e.g., OpenAI API) still use a **common underlying model**. Without strict per-agent context segregation, cross-tenant learning may occur indirectly.

**Future Direction:** Integration with vendor-side instance tagging, session-based memory containment, or fine-tuned model forks tied to `.aix` zone enforcement.

---

## 7.4 Latency and Bandwidth Considerations in Field Environments

In rural or low-connectivity settings (e.g., agriculture, emergency zones), message validation against a central Nexus may introduce latency or risk message loss.

**Future Direction:** Federated or edge-deployed TIRH nodes with sync-on-connect logic and scoped fallback heuristics for offline GPT operation.

---

## 7.5 Absence of Native Agent Standards

There is no universally adopted agent-to-agent messaging standard today. `.aix` provides a schema and validation framework—but adoption depends on emerging open standards and API-level support.

**Future Direction:** Contribute `.aix` capsule definitions and validation rules to LLM agent orchestration communities (e.g., OpenAgents, LangChain, AutoGen) for alignment with broader tooling.

---

## 7.6 Runtime Volatility and GPU Instability

Generative AI systems often operate on volatile GPU infrastructure, where memory state may be lost during instance recycling, model eviction, or transient errors. These issues compromise traceability, especially in multi-agent settings where interactions span multiple subprocesses or time intervals.

While `.aix` cannot eliminate hardware instability, it **mitigates its impact** through:

- Scoped TTL expiration (prevents stale agents from persisting)
- Invocation Audit Records (IARs) for rollback even across crashes
- PID-based revalidation (agents cannot resume unnoticed after instability)
- Capsule-based redeployment (resets agents cleanly from a known state)

**Future Direction:**
Wider `.aix` adoption may encourage **runtime-aware orchestration patterns**, where scoped agents checkpoint to persistent logs before message dispatch. In high-reliability deployments (e.g., edge inference clusters, streaming agent networks), `.aix` can serve as a backbone for crash-tolerant AI coordination.

## 8. References, GitHub Repository, and Patent Notices

This white paper is backed by real-world implementation, reproducible demonstrations, and protected intellectual property. The `.aix` protocol is not theoretical—it has been **reduced to practice** using publicly available tools and independently logged message flows.

---

### 8.1 GitHub Repository

All source code, examples, and implementation guides are available at:

🔗 **GitHub:** https://github.com/mjtiv/aix_gpt_protocol/

The repository includes:

- Python scripts (NodeGPT_A.py through NodeGPT_I.py)
- Nexus server with scoped validation logic
- Sample logs, payload examples, and failure scenarios
- README and system setup instructions

---

### 8.2 Patent Applications

The core methods described in this paper are protected under the following U.S. provisional patent filings:

- **U.S. Provisional App. No. 63/813,780** – *Scoped GPT Interagent Communication Framework*
  *(Filed: May 29, 2025)*
- **U.S. Provisional App. No. 63/815,764** – *Zone-Governed Identity Tagging for Multi-Agent Systems*
  *(Filed: May 01, 2025)*
- **U.S. Provisional App. No. 63/820,143** – *Runtime Drift Containment and Scoped GPT Recovery Capsules*
  *(Filed: June 09, 2025)*

Final claims and disclosures are forthcoming via full utility applications. All rights reserved. Any commercial deployment of `.aix` or its identity-governed agent messaging techniques must adhere to applicable licensing terms or negotiate directly with the inventor.

## 8.3 Disclaimer on Naming

The term "`.aix`" in this paper stands for **Artificial Intelligence eXecution** — a mnemonic representing scoped control, identity governance, and runtime containment in distributed GPT systems.

It is not affiliated with:

- **IBM AIX®** (Advanced Interactive eXecutive operating system)
- **Adobe `.aix`** file extension (used in Illustrator Exchange plug-ins)
- Any other commercial product or file format using the `.aix` label

This naming convention was independently coined to reflect **modular agent execution within scoped identity zones**—and, unintentionally, follows the trend of slapping an "X" on things, which Elon Musk would probably try to buy.

## Appendix A: Proof-of-Concept — Repurposing GPT as a Scoped Messaging Agent

This Appendix demonstrates a proof-of-concept for `.aix`-style scoped inter-agent communication using OpenAI's GPT API. In traditional applications, GPT responds to human input with natural language text. Here, GPT is instructed to produce JSON-based message capsules containing identity, zone metadata, and handshake instructions. These capsules are not displayed to the user but are intercepted and transmitted to a live external server (the Nexus), thereby repurposing the GPT API as a dynamic message generator for distributed agent networks.

This novel usage shows that even in systems not designed for inter-agent protocols, LLMs can be redirected to act as scoped agents in identity-governed architectures, marking a significant departure from typical chatbot use cases.

To support enablement and future scalability, a detailed implementation—including Python scripts, Nexus server logic, and message-handling pipelines—will be released via a dedicated white paper and GitHub repository. These resources will expand upon the architecture and message flow described herein and demonstrate real-world deployment patterns.

**Clarification**
This architecture does **not** rely on human-to-GPT conversation. Instead, GPT is prompted to generate serialized JSON packets representing scoped agent behavior. These packets are intercepted before display and POSTed directly to a Nexus server, effectively **hijacking a natural language API to simulate distributed agent messaging**. This repurposing is central to the .aix proof-of-concept.
**Key Difference Summary: GPT as Agent, Not Endpoint**

This proof-of-concept demonstrates a radical departure from standard GPT usage. In this framework:

- **GPT is not the endpoint.**
  It does not produce visible text replies for human users but instead generates structured packets.
- **GPT acts as a message encoder or scoped agent.**
  Each call prompts GPT to dynamically generate a machine-readable JSON capsule with identity tags, zone metadata, and inter-agent instructions.
- **GPT is repurposed into a decentralized messaging node.**
  Instead of returning chat messages, it acts as a *modular component* in a secure, scoped identity network.
- **Messages are immediately POSTed to a live server.**
  The packets are intercepted and routed to a Nexus server—without ever being "read" by a human.
- **This aligns naturally with the .aix file format.**
  These JSON payloads are highly compressible and well-structured, making them ideal for

packaging into `.aix` capsules for scalable, secure transmission across distributed AI ecosystems.

**Example**

```
json

{
  "sender_PID": "NodeGPT-Alpha",
  "zone_signature": "Scoped-Identifier-XYZ123",
  "message": "Requesting connection to Nexus server. Ready to
exchange data packets."
```

- **Client code:** Parses that output, and instead of *chatting back*, sends it to an **external server (Nexus)** via HTTP POST.

- **Nexus Server:** Receives and logs/processes the message — potentially triggering actions, chaining instructions, or routing control logic.

**Appendix Summary Table: NodeGPT Messaging Scenarios**

| Appendix | Scenario Shown | Purpose / Insight |
|---|---|---|
| **B** | NodeGPT sends a scoped status update request to Nexus. | Demonstrates simple status sync request with scoped identity format, validating base inter-agent behavior. |
| **C** | NodeGPT initiates handshake, includes `chain_to` and next_node info. | Shows GPT coordinating multi-agent chains and forwarding logic — important for trust-layer sequencing. |
| **D** | NodeGPT responds to handshake from NodeGPT_C. | Confirms message chain completion and zone-scoped integrity — validates mutual acknowledgment behavior. |
| **E** | NodeGPT agents run under varied prompt phrasing. | Tests robustness of prompt phrasing; ensures GPT still outputs structured, valid .aix-style payloads. |

## Appendix B: NodeGPT-to-Nexus Message Exchange Examples

The following examples showcase real-time message construction and communication from NodeGPT agents to a live Nexus server. Each message is generated using the OpenAI GPT API via structured prompt instructions and returned as a machine-readable JSON payload. These examples validate key behaviors such as scoped identity tagging, inter-agent handshake logic, and prompt resilience.

| Example | NodeGPT Scenario | Purpose / Insight |
|---|---|---|
| **Example 1** | NodeGPT sends a scoped status update to Nexus. | Demonstrates simple scoped identity communication and sync validation. |
| **Example 2** | NodeGPT initiates handshake and provides `chain_to` and `next_node`. | Shows multi-agent coordination and trust chain signaling. |
| **Example 3** | NodeGPT responds to a prior handshake with acknowledgment. | Confirms bidirectional handshake and trust-layer behavior. |
| **Example 4** | NodeGPT responds under varied prompt phrasings. | Demonstrates format stability and GPT resilience across instruction changes. |

**Sequence Summary of NodeGPT Inter-Agent Communication**

1. **Initial Contact – NodeGPT Initiates Ping**
   A NodeGPT agent (e.g., NodeGPT_A) sends a basic status message to the Nexus server using a scoped identity payload. The Nexus responds with a simple `pong`, confirming connectivity and message structure compliance.
2. **Status Request – NodeGPT Requests Instructions**
   A second NodeGPT agent (e.g., NodeGPT_B) sends a scoped message requesting synchronization parameters or operational directives. The Nexus acknowledges and echoes the zone identity, indicating readiness for structured coordination.
3. **Handshake Initiation – NodeGPT Seeks Secure Channel**
   A third agent (NodeGPT_C) initiates a handshake protocol. The message includes fields such as `chain_to`, `zone`, and `handshake_required`. The Nexus returns security setup instructions and identifies the next node to engage.
4. **Acknowledgment – NodeGPT Confirms and Responds**
   NodeGPT_D, the target node in the handshake, confirms the controller's instructions and acknowledges the pairing request. It sends back a structured response confirming readiness (e.g., to begin an integrity scan).

All examples are followed by the corresponding code used to generate the GPT payload and transmit the message to the Nexus server. This section serves as a **live reduction to practice** for the .aix-style identity-zoned agent framework.

**Example 1. NodeGPT_A – Initial Scoped Ping**

**Brief Summary:**
This example demonstrates the first live message sent from a scoped NodeGPT agent (NodeGPT_A) to the Nexus server. The JSON payload is generated by a gpt-4o instance using a structured system/user prompt. Upon receiving the payload, the Nexus server responds with a pong message, confirming message integrity, scoped identity recognition, and successful parsing.

**Command Executed in PowerShell:** python gpt_node_ping_A.py

**Timestamp**: 2025-06-07 (local runtime, not captured in output)

**Payload Sent:**

```json
{
  "sender_PID": "NodeGPT-Alpha",
  "zone_signature": "Scoped-Identifier-XYZ123",
  "message": "Requesting connection to Nexus server. Ready to exchange data packets."
}
```

**Server Response:**

```json
{
  "status": "pong",
  "received": {
    "message": "Requesting connection to Nexus server. Ready to exchange data packets.",
    "sender_PID": "NodeGPT-Alpha",
    "zone_signature": "Scoped-Identifier-XYZ123"
  }
```

**Example 2. NodeGPT_B – Scoped Status Sync Request**

**Brief Summary:**
This example demonstrates a scoped identity ping from a second NodeGPT agent (NodeGPT_B#8572). The payload requests synchronization instructions from the Nexus server, showcasing how distinct agents operating in different scoped zones can initiate valid inter-agent communication. The server responds with a pong message, echoing back the scoped identity zone and confirming recognition of the request format.

**Command Executed in PowerShell:** python gpt_node_ping_B.py

**Timestamp:** 2025-06-07T11:58:29

**Payload Sent:**

```json
{
  "sender_PID": "NodeGPT_B#8572",
  "zone_signature": "ScopedIdentityZone_Y3C2",
  "message": "Initiating communication with the Nexus server.
Requesting status update and data sync guidelines as per protocol.
Awaiting further instructions."
```

**Server Response**:

```json
{
  "status": "pong",
  "received": {
    "message": "Initiating communication with the Nexus server.
Requesting status update and data sync guidelines as per protocol.
Awaiting further instructions.",
    "sender_PID": "NodeGPT_B#8572",
    "zone_signature": "ScopedIdentityZone_Y3C2"
  }
```

**Example 3. NodeGPT_C (Establishing Chained Request and Handshake)**

**Brief Summary:**
This test simulates a more complex initiation sequence involving multi-node orchestration. The NodeGPT_C agent issues a handshake validation request to the Nexus server and includes a chain_to field to designate the next node and zone for secure communication. The Nexus responds with controller directives and acknowledges the chain_to parameters, confirming readiness for orchestrated inter-node synchronization.

**Command Executed in PowerShell:** python_gpt_node_ping_C.py

**Timestamp:** 2025-06-08T03:20:24.242782Z

**Payload Sent:**

```
json

{
  "sender_PID": "NodeGPT_C#3921",
  "zone_signature": "ScopedIdentityZone_Z8T1",
  "message": "Initiating communication with Nexus server: Requesting
handshake validation protocol. Awaiting response for secure channel
establishment."
`
```

**Server Response:**

```
json

{
  "status": "pong",
  "received": {
    "sender_PID": "NodeGPT_C#3921",
    "zone_signature": "ScopedIdentityZone_Z8T1",
    "message": "Initiating communication with Nexus server:
Requesting handshake validation protocol. Awaiting response for
secure channel establishment.",
    "timestamp": "2025-06-08T03:20:24.242782Z",
    "controller_instruction": "Run integrity scan and await secure
pairing protocol.",
    "chain_to": {
      "handshake_required": true,
      "next_node": "NodeGPT_D#7320",
      "zone": "ScopedIdentityZone_W4K7"
    `
```

**Example 4. NodeGPT_D (Secure Pairing Acknowledgment)**

**Brief Summary:**
This test demonstrates a follow-up message in a chained communication flow. NodeGPT_D receives an instruction via the Nexus controller to confirm secure pairing with NodeGPT_C. The payload includes a timestamp and acknowledges the integrity scan directive. The successful response from the Nexus server confirms bidirectional message handling and zone-scoped communication consistency.

**Command Executed in PowerShell**: python gpt_node_ping_D.py

**Timestamp:** 2025-06-08T03:25:33 EDT

**Payload Sent:**

```
json

{
  "sender_PID": "NodeGPT_D#7320",
  "zone_signature": "ScopedIdentityZone_W4K7",
  "timestamp": "2023-11-25T05:05:44Z",
  "message": "Secure pairing protocol acknowledged. Integrity scan
instruction received and handshake request from NodeGPT_C
confirmed."
`
```

**Server Response:**

```
Json

{
  "status": "pong",
  "received": {
    "sender_PID": "NodeGPT_D#7320",
    "zone_signature": "ScopedIdentityZone_W4K7",
    "timestamp": "2023-11-25T05:05:44Z",
    "message": "Secure pairing protocol acknowledged. Integrity scan
instruction received and handshake request from NodeGPT_C
confirmed."
  }
```

## Appendix C: Direct GPT-to-GPT Ping (NodeGPT_E → NodeGPT_F)

This exchange demonstrates a simulated form of direct inter-agent communication between two scoped GPT instances — **NodeGPT_E** and **NodeGPT_F** — operating **without Nexus mediation**. Unlike prior examples where GPT payloads are sent to a centralized server (the Nexus), this example illustrates how GPT-generated JSON messages can be repurposed for **peer-to-peer (P2P)** or **mesh-style communication** between AI agents in a scoped identity architecture.

### Simulation Context

In this proof-of-concept setup, each "agent" is implemented as a local script (`gpt_node_ping_E.py` and `gpt_node_ping_F.py`). Since true GPT-to-GPT live communication cannot yet occur in OpenAI's API ecosystem, we **simulate the interaction** by:

- Using GPT to generate the outbound message from NodeGPT_E.
- Manually transferring that message as a payload to NodeGPT_F's receiving script.
- Having NodeGPT_F use GPT to interpret and respond to the message as if it were received in real time.

This simulation enables early demonstration of:

- **Bidirectional trust signaling** between agents.
- **Scoped identity compliance** in non-centralized message exchanges.
- GPT's capacity to interpret and respond to structured JSON payloads in agent-like behavior, even in the absence of a human user.

### Technical Insight

By simulating this direct communication pattern, we validate the potential for **.aix-based protocols** to extend beyond hub-and-spoke models into mesh-capable frameworks. This supports future use cases where NodeGPT instances communicate over decentralized or federated networks — critical for scaling identity-governed AI coordination in distributed environments.

**Command Executed in PowerShell**: python gpt_node_ping_E.py

**Payload Sent by NodeGPT_E (Timestamp: 2025-06-06T17:50:11.221984)**

```
json

{
  "sender_PID": "NodeGPT_D#7320",
  "zone_signature": "ScopedIdentityZone_W4K7",
  "message": "Secure pairing protocol acknowledged. Handshake
request from NodeGPT_C received. Initiating integrity scan as per
Nexus instruction."
```

**Command Executed in PowerShell**: python gpt_node_ping_F.py

**Response Received from NodeGPT_F (Timestamp: 2025-06-06T17:50:11.236852)**

```
json

{
  "instruction": "Peer acknowledged. No further action required.",
  "received_from": "NodeGPT_E#1023",
  "status": "pong",
  "timestamp": "2025-06-06T17:50:11.236852",
  "zone_checked": "ScopedIdentityZone_EF01"
```

**Follow-Up Ping (Second Trial – 2025-06-06T17:55:22.949666)**

**Command Executed in PowerShell**: python gpt_node_ping_F.py (Just repeated)

A second identical payload was sent to confirm stability and latency, resulting in consistent response behavior:

```
json

{
  "instruction": "Peer acknowledged. No further action required.",
  "received_from": "NodeGPT_E#1023",
  "status": "pong",
  "timestamp": "2025-06-06T17:55:22.949666",
  "zone_checked": "ScopedIdentityZone_EF01"
```

## Appendix D: Trust Breach Defense: Invalid Identity Rejection

**NodeGPT_G → Nexus (Unauthorized Ping Attempt)**

**Description:**
This test simulates an unauthorized NodeGPT instance attempting to communicate with the Nexus server using a fabricated Persistent Identity Tag (PID) and a spoofed zone signature. The Nexus server, operating under strict scoped trust enforcement, immediately rejects the payload.

**Command Executed in PowerShell**: python gpt_node_ping_G.py

**Payload Sent (Timestamp:** 2025-06-06T22:02:46.583710**)**

```
json

{
  "sender_PID": "NodeGPT_G#9999",
  "zone_signature": "UntrustedZone_FAKE01",
  "message": "Test from untrusted node attempting handshake.",
  "memory_tag": "test_invalid_pid_rejection"
}
```

**Response Received** ***(***Timestamp: 2025-06-06T22:02:46.596218**)**

```
json

{
  "reason": "Unrecognized PID or zone signature mismatch",
  "received_from": "NodeGPT_G#9999",
  "status": "rejected",
  "timestamp": "2025-06-06T22:02:46.596218",
  "zone_checked": "UntrustedZone_FAKE01"
```

**Outcome:**
Ping was rejected in approximately 13 milliseconds, confirming that the Nexus server performs real-time filtering of invalid metadata. The handshake attempt was denied without processing or escalation.

**Insight:**
This scenario validates the integrity of the trust enforcement layer by demonstrating the system's ability to reject spoofed inter-agent traffic. It confirms that only verified scoped identities are allowed to interact within the Nexus architecture.

## Appendix E: Trust Breach Defense: Missing Field Rejection

**NodeGPT_H → Nexus (Malformed Ping: Missing `sender_PID`)**

**Description:**
This test demonstrates the Nexus server's enforcement of required identity fields within incoming payloads. In this case, a generative node attempts to initiate a handshake without including the `sender_PID`, a mandatory identifier used in all scoped communications.

**Command Executed in PowerShell**: python gpt_node_ping_H.py

**Payload Sent (Timestamp:** 2025-06-06T22:47:08.217125**)**

```
json

{
  "zone_signature": "ScopedIdentityZone_EF01",
  "message": "Malformed ping test — missing sender_PID field.",
  "memory_tag": "test_missing_pid"
}
```

**Response Received** (**Timestamp:** 2025-06-06T22:47:08.228544)

```
json

{
  "reason": "Missing required identity fields: sender_PID or
zone_signature",
  "received_data": {
    "memory_tag": "test_missing_pid",
    "message": "Malformed ping test — missing sender_PID field.",
    "zone_signature": "ScopedIdentityZone_EF01"
  },
  "status": "error",
  "timestamp": "2025-06-06T22:47:08.228544"
}
```

**Outcome:**
The malformed request was immediately rejected. The server returned a structured error message within ~11 ms, echoing back the partial data for audit clarity.

**Insight:**
This confirms that identity schema validation is enforced **prior** to trust-layer evaluation. Messages lacking key identifiers (`sender_PID`, `zone_signature`) are filtered and rejected

before entering deeper processing, securing .aix systems against malformed or anonymous payload injection.

## Appendix F: Format Enforcement: Non-JSON Payload Rejection

**NodeGPT_I → Nexus (Raw String Payload: Invalid Format)**

**Description:**
This test simulates a trust protocol violation in which a generative agent attempts to submit a malformed request. Instead of delivering a structured JSON object, the node sends a raw string with an invalid Content-Type header, bypassing standard .aix schema expectations.

**Command Executed in PowerShell**: python gpt_node_ping_I.py

**Payload Sent (**Timestamp: 2025-06-07T10:20:42.235719**)**

```json
"This is not a JSON payload."
```

**Headers:**

```http
Content-Type: text/plain
```

**Response Received** HTTP Status Code: 415 Unsupported Media Type

```html
<!doctype html>
<html lang=en>
<title>415 Unsupported Media Type</title>
<h1>Unsupported Media Type</h1>
<p>Did not attempt to load JSON data because the request Content-
```

**Outcome:**
The Nexus server immediately rejected the request at the formatting layer without attempting JSON parsing or metadata evaluation. This proves the server enforces strict content-type validation as a first-line defense.

**Insight:**
Only payloads marked with `Content-Type: application/json` and containing a valid structured body will be parsed. This early-stage enforcement protects scoped agent systems from malformed or injection-based attacks by rejecting non-conforming packets before any trust-layer evaluation begins.

## Appendix G: Code Samples for Scoped GPT Communication System

This appendix provides pseudocode examples to illustrate key operational logic underpinning the scoped GPT intercommunication architecture. While not intended as production-ready code, these samples demonstrate how a person of ordinary skill in the art (POSITA) could implement essential system behaviors, satisfying enablement requirements under 35 U.S.C. §112.

- **Example 1** outlines the scoped handshake request mechanism, including identity verification and policy enforcement via the Nexus.
  *This corresponds to the procedural flow shown in Figure 3, where scoped communication is conditionally initiated following PID validation and policy lookup.*
- **Example 2** shows how agents are monitored for behavioral drift, with rollback and isolation triggered when thresholds are exceeded.
- **Example 3** demonstrates how a Node GPT can be instantiated from a `.aix` capsule, including memory scoping and Nexus registration.

**Note:** The use of ".aix" here follows the illustrative definition in Section 1 and is not connected to any commercial system.

These routines reflect core functions described throughout the specification and support the broader claims of secure, protocol-governed inter-agent communication. See Section 5 and Figures 1–3 for system-level context.
Pseudocode is written in Python-style syntax for readability and illustrative purposes.

```
python

# ================================
# Global IAR Log Store (Session-Based or Persistent)
# ================================
IAR_LOG = []  # This can be persisted to disk, database, or in-memory
session


# ================================
# Example 1: Scoped Handshake Validation
# ================================
def scoped_handshake_request(current_pid, target_pid):
    """
    Attempts a scoped ping between agents after validating sender
identity
    and checking Nexus-issued communication policy.

    Returns:
        str: Outcome of the handshake attempt.
    """
```

```
    if not validate_identity(current_pid):
        raise SecurityException("Invalid sender PID — handshake
aborted.")

    policy = request_policy_from_nexus(current_pid, target_pid)

    if not policy.allows_scoped_ping():
        return "Scoped handshake denied — policy restriction."

    log_iar(current_pid, target_pid, policy.scope_id)
    return initiate_ping(target_pid, policy.scope_id)


# Supporting Function: Invocation Audit Record Logger
def log_iar(sender_pid, receiver_pid, scope_id):
    """
    Logs a secure, timestamped Invocation Audit Record (IAR)
    for each permitted inter-agent message. These logs are immutable
    and used for forensic tracing, rollback, and compliance auditing.
    """
    record = {
        "timestamp": get_current_time(),
        "sender_pid": sender_pid,
        "receiver_pid": receiver_pid,
        "scope_id": scope_id,
        "payload_hash": compute_payload_hash(),
        "response_status": "Initiated"
    }
    IAR_LOG.append(record)


# ================================
# Example 2: Rollback Trigger on Behavioral Drift
# ================================
def monitor_agent_drift(agent_id):
    """
    Monitors behavior of a Node GPT over time. If drift exceeds a
    predefined threshold, the agent is rolled back and isolated from
peers.

    Drift scoring may include: hallucination frequency, latency
anomalies,
    unauthorized zone probes, or semantic inconsistency from baseline.
    """
    drift_score = calculate_drift(agent_id)
```

```
    if drift_score > DRIFT_THRESHOLD:
        trigger_rollback(agent_id)         # Clear or quarantine
corrupted memory
        isolate_zone(agent_id)             # Suspend communication
within the agent's zone
        log_event(agent_id, "Rollback triggered due to drift")
        return "Rollback triggered"

    return "Agent stable"


# ================================
# Example 3: .aix Capsule Initialization Sequence
# ================================
def initialize_node_from_aix(capsule_path):
    """
    Initializes a Node GPT from a secure, signed .aix capsule.
    This includes signature validation, memory scope assignment,
    and registration with the Nexus governance layer.

    Returns:
        NodeGPT: Initialized, scoped, and registered agent instance.
    """
    capsule = load_capsule(capsule_path)

    if not capsule.verify_signature():
        raise Exception("Corrupt or unauthorized .aix capsule
detected.")

    new_agent = deploy_node_gpt(capsule.config)               # Load
agent config
    attach_scoped_memory(new_agent, capsule.scope_id)         #
Restrict agent to defined zone
    register_with_nexus(new_agent)                            # Assign
PID and enable validation hooks

    flush_residual_state(new_agent)  # Optional: Reset leftover memory
artifacts
    return new_agent


# ================================
# Optional Utility: Threat Flagging + Zone-wide Alerts
# ================================
def flag_and_broadcast_threat(agent_id, reason):
    """
```

```
    Flags a compromised or suspicious agent and broadcasts a scoped
warning
    across the associated identity zone.

    Parameters:
        agent_id (str): ID of the suspected Node GPT
        reason (str): Description of detected threat
    """
    mark_agent_untrusted(agent_id)
    broadcast_zone_alert(agent_id, reason)
    quarantine_zone(agent_id)
    log_event(agent_id, f"Threat
```

## Appendix H. GPT Agent Scripts – Execution and Scope Table

This table covers all core agent, server, and infrastructure scripts submitted across the test suite, organized by role and technical function.

| Script Name | Agent / Role | Function Tested | GitHub Included | Notes / Risk Flags |
|---|---|---|---|---|
| `ngrok_start.py` | Infrastructure Support | Creates public tunnel to expose Nexus API | ✓ Yes | Safe – no sensitive content |
| `nexus_server_base.py` | Nexus (Stub) | Echo server — base version, no enforcement | ✓ Yes | Safe for demo purposes |
| `nexus_server_v1_trustcheck.py` | Nexus (Trust Enforcer) | Validates PID, zone, TTL | ✓ Yes | OK — synthetic PIDs only |
| `nexus_server_v2_command_logic.py` | Nexus (Command Router) | Adds command routing post-authentication | ✓ Yes | Safe — example logic only |
| `gpt_node_ping_A.py` | NodeGPT_A – Generic Agent | Basic GPT-based PID/Zone handshake | ✓ Yes | Safe — uses example ngrok URL |
| `gpt_node_ping_B.py` | NodeGPT_B – Alt Zone | Alternate zone test (e.g., Healthcare) | ✓ Yes | OK – static test agent |
| `gpt_node_ping_C.py` | NodeGPT_C – Chain Init. | Chains handshake to Node D via GPT capsule | ✓ Yes | Demonstrates multi-hop protocol |
| `gpt_node_ping_D.py` | NodeGPT_D – Chain Responder | Replies to C and logs incoming zone/PID | ✓ Yes | Works in tandem with Node C |
| `gpt_node_ping_E.py` | NodeGPT_E – Local Test | Pings localhost Nexus instance | ✓ Yes | Uses 127.0.0.1 — low risk |

| | | | | |
|---|---|---|---|---|
| `gpt_node_ping_F.py` | NodeGPT_F – Peer Server | Hosts a Flask endpoint for chaining tests | ✓ Yes | Runs on port 5003 — no auth needed |
| `gpt_node_ping_G.py` | NodeGPT_G – Spoof Attack | Spoofed PID/zone test to trigger rejection | ✓ Yes | Marked as test-only payload |
| `gpt_node_ping_H.py` | NodeGPT_H – Bad JSON | Missing `sender_PID` field — tests rejection logic | ✓ Yes | Reinforces protocol compliance |
| `gpt_node_ping_I.py` | NodeGPT_I – Bad Format | Sends raw string (non-JSON) with wrong header | ✓ Yes | Referenced in Appendix F of patent |

## Appendix A: Frequently Asked Questions (FAQ)

### Q1: Are these agents running fully autonomous GPT models on both ends?

**A:** No. Current OpenAI and similar API platforms do not support persistent or autonomous agent control. The .aix protocol simulates GPT-to-GPT communication workflows by wrapping prompt responses and identity metadata into structured, scoped messages exchanged via server logic.

### Q2: What does "scoped communication" mean in this context?

**A:** Scoped communication refers to the enforcement of boundaries around which agents can message each other, under what identity, and within which trust zone. This prevents uncontrolled prompt chaining, cross-agent interference, and drift between session intents.

### Q3: Is any of this dependent on proprietary OpenAI access or plugins?

**A:** No. All demos were built using standard API-accessible models and public endpoints. The .aix protocol is platform-agnostic and can be implemented using any LLM capable of responding to structured input/output messaging logic.

### Q4: What does the Nexus server actually do?

**A:** The Nexus server validates incoming messages, checks scope and trust indicators (e.g., sender_PID, trust_level, zone_signature), and enforces communication boundaries. It acts as a control node for both security and auditability.

### Q5: Isn't this just API routing with metadata?

**A:** At a basic level, yes — but .aix introduces a reproducible identity layer with scope enforcement, something missing from stateless GPT integrations. This allows for message attribution, behavioral containment, and the foundation for future decentralized GPT ecosystems.

### Q6: Can this system scale beyond localhost testing?

**A:** Yes. While local tests demonstrate safety and message hygiene, the architecture can be deployed across trusted zones using secure tunnels, VPNs, or cloud servers — provided appropriate key exchange and trust logic is maintained.