

AIR Blackbox Compliance Report

LlamaIndex Framework Analysis

Framework	LlamaIndex (run-llama/llama_index)
Scanner Version	air-blackbox v1.2.2
Scan Date	March 13, 2026
Articles Covered	EU AI Act Articles 9, 10, 11, 12, 14, 15
Detection Method	95% automated, 5% hybrid/manual
License	MIT (LlamaIndex) / Apache 2.0 (AIR Blackbox)

15	11	11	37
Passing	Warnings	Failing	Total Checks

Executive Summary

LlamaIndex is a leading data framework for building LLM-powered applications, with 4,143 Python files and over 463,000 lines of code across a comprehensive monorepo spanning core libraries, integrations, and community packs. This report evaluates LlamaIndex against EU AI Act requirements across 6 articles using the AIR Blackbox open-source compliance scanner.

LlamaIndex demonstrates strong foundational compliance patterns including robust input validation (643 files with Pydantic/dataclass enforcement), extensive observability infrastructure (406 files with tracing patterns and a dedicated instrumentation module), and solid prompt injection defenses (72 files). The framework's AgentMesh integration includes cryptographic trust verification with Ed25519 signing, which is notably advanced for agent identity management.

Key areas for improvement include missing governance documentation (RISK_ASSESSMENT.md, DATA_GOVERNANCE.md), error handling coverage in LLM call paths (46% coverage), and docstring coverage at 40% (below the 50% threshold for comprehensive technical documentation under Article 11). The 11 failing checks are primarily infrastructure-level (gateway, vault, signing key) and documentation-level, not code-quality issues.

Key Findings (Code-Level)

Metric	Value	Assessment
--------	-------	------------

Python files scanned	4,143	Comprehensive monorepo
Input validation (Pydantic/dataclass)	643 files (16%)	PASS
PII-aware patterns	341 files	PASS
Structured logging	398 files (10%)	WARN
Tracing / observability	406 files + dedicated instrumentation module	PASS
Human-in-the-loop patterns	56 files	PASS
Retry / backoff logic	176 files	PASS
Fallback / recovery patterns	262 files	PASS
Prompt injection defense	72 files	PASS
Output validation / parsing	137 files	PASS
Rate limiting / budget controls	231 files	PASS
LLM call error handling	127/275 files (46%)	WARN
Docstrings coverage	5,803/14,339 (40%)	WARN
Type annotations	5,778/11,767 (49%)	WARN
Agent action audit trail	5 files	PASS
User identity binding	25 files	PASS

Article 9 - Risk Management

Check	Status	Type	Evidence
Risk assessment document	FAIL	Human	No RISK_ASSESSMENT.md found. Fix: Create documenting identified risks, likelihood, impact, and mitigations
Risk mitigations active	FAIL	Human	0/4 mitigations active (guardrails.yaml, signing key, vault, OTel). Infrastructure-level, not code issue
LLM call error handling	WARN	Auto	127/275 files with LLM calls have error handling (46%). Missing coverage primarily in llama-index-packs community contributions
Fallback/recovery patterns	PASS	Auto	Fallback patterns found in 262 files. Strong recovery architecture across core and integrations

Analysis: LlamaIndex has excellent fallback/recovery patterns (262 files) suggesting mature error architecture. The error handling gap is concentrated in community-contributed packs rather than core modules. The missing governance documents and infrastructure mitigations are common across all frameworks scanned and represent organizational maturity rather than code quality issues.

Article 10 - Data Governance

Check	Status	Type	Evidence
PII detection in prompts	FAIL	Auto	Gateway not reachable. Runtime check requires gateway deployment
Data governance documentation	FAIL	Human	No DATA_GOVERNANCE.md found. Fix: Document data sources, consent, quality measures, retention
Data vault (controlled storage)	FAIL	Auto	No vault configured. Infrastructure-level requirement
Input validation / schema enforcement	PASS	Auto	643/4,143 Python files use Pydantic, dataclass, or similar validation. Strong schema enforcement across the codebase
PII handling in code	PASS	Auto	PII-aware patterns found in 341 files. Indicates systematic awareness of personal data handling

Analysis: LlamaIndex's Pydantic-first architecture (643 files with validation) is a significant compliance strength. The 341 files with PII-aware patterns suggest the framework considers personal data handling. The failing checks are all infrastructure/documentation requirements, not code deficiencies.

Article 11 - Technical Documentation

Check	Status	Type	Evidence
System description (README)	PASS	Human	Comprehensive README.md found with architecture documentation
Runtime system inventory (AI-BOM)	FAIL	Auto	No traffic data. Requires gateway for runtime model inventory generation
Model card / system card	WARN	Human	No MODEL_CARD.md found. Run: air-blackbox discover --generate-card
Code documentation (docstrings)	WARN	Auto	5,803/14,339 public functions/classes have docstrings (40%). Below 50% threshold
Type annotations	WARN	Auto	5,778/11,767 public functions have type hints (49%). Close to 50% threshold

Analysis: Docstring coverage at 40% and type annotations at 49% are both close to passing thresholds. Given the massive codebase (14,339 public entities), these percentages represent substantial documentation effort. The monorepo structure with community packs naturally dilutes percentages, as core modules likely have significantly higher coverage.

Article 12 - Record-Keeping

Check	Status	Type	Evidence
Automatic event logging	FAIL	Auto	Gateway not reachable. Runtime logging requires gateway deployment
Tamper-evident audit chain	FAIL	Auto	No TRUST_SIGNING_KEY set. Requires HMAC-SHA256 signing configuration
Log detail and traceability	FAIL	Auto	No logged records. Route traffic through gateway for automatic capture
Application logging	WARN	Auto	Logging found in 398/4,143 files (10%). Present but not pervasive
Tracing / observability	PASS	Auto	Tracing patterns in 406 files. Dedicated llama-index-instrumentation module with span handlers, event dispatchers, and OpenTelemetry support
Agent action audit trail	PASS	Auto	Action-level audit logging found in 5 files

Analysis: LlamaIndex has a standout observability architecture. The dedicated llama-index-instrumentation package provides span handlers, event dispatchers, and a full instrumentation API. Combined with callback integrations for W&B, OpenInference, Opik, and PromptLayer, this gives operators extensive runtime visibility. The failing checks are all gateway-dependent infrastructure requirements.

Article 14 - Human Oversight

Check	Status	Type	Evidence
Human-in-the-loop mechanism	WARN	Auto	No traffic data for runtime analysis. Gateway required for oversight pattern detection
Kill switch / stop mechanism	FAIL	Auto	Gateway not running. Runtime kill switch requires gateway deployment
Operator documentation	WARN	Manual	No OPERATOR_GUIDE.md found. Recommended for Article 14 compliance
Human-in-the-loop patterns	PASS	Auto	Human oversight patterns found in 56 files. Agent workflow includes confirmation and approval mechanisms
Usage limits / budget controls	PASS	Auto	Rate limiting or budget controls found in 231 files. Strong resource governance
Agent-to-user identity binding	PASS	Auto	User identity binding found in 25 files (user_id, memory store, delegation tracking)

Check	Status	Type	Evidence
Token scope / permission validation	PASS	Auto	Scope or permission validation found in 1 file
Token expiry / execution bounding	PASS	Auto	Execution boundary patterns found in 25 files (max steps, timeouts, iteration limits)
Agent action boundaries	PASS	Auto	Action boundary controls found in 6 files

Analysis: Article 14 is LlamaIndex's strongest area with 6 of 9 checks passing. The framework provides comprehensive human oversight mechanisms including budget controls (231 files), execution boundaries, and identity binding. The AgentMesh integration adds cryptographic trust verification with Ed25519 signing for inter-agent identity, which is advanced among the frameworks we've scanned.

Article 15 - Accuracy, Robustness & Cybersecurity

Check	Status	Type	Evidence
Prompt injection protection	FAIL	Auto	Gateway not running. Runtime injection filtering requires gateway
Error resilience	WARN	Auto	No traffic data to measure resilience metrics
API access control	WARN	Human	No API keys detected in scan environment
Adversarial robustness testing	WARN	Manual	No red team / adversarial testing evidence found
Retry / backoff logic	PASS	Auto	Retry/backoff patterns in 176 files. Robust fault tolerance across integrations
Prompt injection defense	PASS	Auto	Injection defense patterns found in 72 files. Code-level sanitization present
Unsafe input handling	WARN	Auto	Possible raw user input in prompts in 15 files (primarily community packs and vector store integrations)
LLM output validation	PASS	Auto	Output parsing/validation found in 137 files. Strong output contract enforcement

Analysis: LlamaIndex demonstrates solid cybersecurity foundations with 72 files containing prompt injection defenses and 137 files with output validation. The 15 files flagged for unsafe input handling are primarily in community packs and vector store integrations, suggesting the core framework handles input safely but some integrations may bypass sanitization.

Notable Compliance Patterns

AgentMesh Cryptographic Trust

LlamaIndex's AgentMesh integration includes a cryptographic trust verification system using Ed25519 signatures. This includes TrustPolicy configuration with minimum trust scores, capability-based access control, audit query logging, and automatic blocking of unverified agents. The CMVKIdentity system provides cryptographic agent identity that goes beyond what most frameworks offer for inter-agent trust management. This is highly relevant to Article 14 (Human Oversight) and Article 15 (Cybersecurity) requirements.

Instrumentation Architecture

The dedicated llama-index-instrumentation package provides a pluggable observability architecture with span handlers, event dispatchers, and event handler abstractions. This design allows operators to plug in any observability backend (OpenTelemetry, W&B, PromptLayer, OpenInference, Opik) without modifying application code. This is a strong Article 12 (Record-Keeping) compliance pattern that enables production audit trails.

Multi-Agent Workflow Engine

The core agent workflow module includes ReAct agents, function agents, CodeAct agents, and multi-agent workflows with explicit context management. The agent_context.py module maintains conversation state and tool call history, which supports audit trail requirements under Article 12.

Questions for LlamaIndex Maintainers

The scanner uses static pattern matching, which has limitations. We'd appreciate validation on these specific findings:

- 1. AgentMesh trust verification** - The scanner detected TrustPolicy with require_verification, min_trust_score, and block_unverified flags. Is this actively used in production multi-agent deployments, or is it experimental/optional?
- 2. User identity binding (25 files)** - The scanner found user_id patterns across memory stores and agent contexts. Is this primarily for memory retrieval scoping, or does it serve as actual identity-based access control?
- 3. Callback manager instrumentation** - The callback_manager pattern appears in 406+ files. Is this the primary mechanism for production tracing, or has it been superseded by the newer llama-index-instrumentation package?
- 4. Unsafe input in community packs** - 15 files were flagged for potentially passing raw user input into LLM prompts without sanitization. These are mostly in community-contributed packs. Is there a review process or security policy for pack contributions?
- 5. Execution boundaries** - The scanner matched max_iterations and similar patterns in 25 files. Do these serve as hard execution limits to prevent runaway agents, or are they configuration suggestions that can be overridden?

How to Reproduce

This scan was performed using AIR Blackbox v1.2.2, an open-source EU AI Act compliance scanner for Python AI frameworks. Apache 2.0 licensed, runs entirely locally - no code leaves your machine.

```
pip install air-blackbox
git clone https://github.com/run-llama/llama_index.git
air-blackbox comply --scan ./llama_index -v
```

18 code-level checks + 5 OAuth delegation pattern checks + 14 infrastructure checks. The scanner maps findings to EU AI Act Articles 9, 10, 11, 12, 14, and 15.

Generated by AIR Blackbox v1.2.2 | airblackbox.ai | github.com/airblackbox | PyPI: [air-blackbox](https://pypi.org/project/air-blackbox)
Report generated: March 13, 2026 | Scanner: Apache 2.0 | Framework: MIT