JANUS Technical Whitepaper

AI Agent Operating-System Governance Kernel

Executive Summary

As large language models (LLMs) and Agent technology spread rapidly, AI is moving from low-risk copy generation into high-value core business scenarios such as automated trading, contract review, and cross-border settlement. Large-scale commercial adoption still faces three insurmountable mountains:

- ✧ Long-term memory fracture: AI has no stable identity; history is wiped after restarts or model switches;
- ✧ Probabilistic hallucination: LLM factual hallucination creates compliance risk and direct capital loss;
- ✧ High collaboration barriers: Centralized orchestration frameworks cannot support trusted autonomous collaboration among massive Agents.

Existing security middleware commonly suffers from high intrusion, statically bound rules, no unified ontology standard, and incomplete audit trails—leaving high-value workloads unable to be fully automated by AI.

JANUS is an Agent operating-system governance kernel and a Dual-Ontology semantic protection infrastructure. Its product mindshare is: CRE (Rules & Axiom Engine) as the rule-evolution hub, Axiom as deterministic red-line configuration, SemanticShield (Gate) as the pre-execution checkpoint, Ontology defining Agent rights and domain boundaries, and Epoch enabling seamless rule hot updates.

JANUS provides two compatible attach paths:

1. Native Runtime mode: Build compliant digital-life Agents from scratch with full identity, semantic validation, and audit capability;
2. Adapter completion mode (Adapter SDK / HTTP Gate): Without refactoring existing Agent business code, rapidly complete DID identity, ontology binding, semantic gate, and audit attestation—using the official TypeScript SDK / HTTP API as the shortest path.

With a BYO (Bring Your Own) model-decoupled architecture, JANUS binds to no LLM vendor. Through Self-Ontology it grants Agents a globally unique persistent identity; through Target-Ontology it builds a deterministic

semantic gate that intercepts hallucination before execution; CRE continuously compiles and distributes evaluable rules; Semantic Discovery lets Agents form Swarm collaboration networks on the control plane—engineering an intersubjectivity layer among agents.

Chapter 1: Background and Industry Challenges

1.1 Security & Compliance Bottlenecks for Large-Scale AI Agent Deployment

AI Agents are penetrating automated trading, contract review, cross-border settlement, and other high-value businesses, but the industry shares three foundational defects rooted in AI lacking an “existence foundation”:

Defect	Manifestation	Root cause
Memory fracture (identity discontinuity)	Restart or model switch clears session memory and fulfillment records; digital assets cannot accumulate.	No cryptographic persistent identity anchor.
Factual hallucination (uncontrolled reasoning)	LLM probabilistically fabricates data, causing capital loss and compliance penalties.	No objective world-rule anchoring.
Collaboration barrier (no subject consensus)	Multi-Agent systems lack a shared semantic standard; cannot autonomously discover, trust, and settle.	No standardized capability cards and credit system.

Ordinary Prompt Engineering and post-hoc verification only relieve symptoms shallowly; they cannot intercept risk before execution.

1.2 Fatal Engineering Defects of Existing Governance Middleware

Defect	Manifestation
High-intrusion retrofit	Requires reconstructing planning, memory, and RAG; development cost is extreme.
Static rule binding	Business red lines are hard-coded; business change requires restarts; no hot update.
No unified ontology standard	Identity, capability, and rights lack standardized description; cross-platform interoperability fails.

Incomplete audit chain	Only intercepts errors; cannot produce full-chain audit proofs for accountability / ledgering.

1.3 Cascading Hallucination Risk in Multi-Agent Distributed Networks

In unmanaged distributed networks, Cascading Hallucination appears easily: Agent A emits wrong data, downstream Agent B trusts it and amplifies risk, with no unified validation gateway and no traceable fault path. Agents also lack standardized capability cards and automated settlement, so a commercially viable collaboration network cannot form.

1.4 Market Gap

The industry needs an OS-level Agent governance infrastructure that is low-intrusion, model-agnostic, ontology-driven, axiom self-evolving, hot-updatable, and full-chain audited. JANUS exists for this—as a universal semantic firewall for AI agents, providing a production-oriented safety passport for high-value digital business.

Chapter 2: Core Positioning and Top-Level Philosophy

2.1 Product Definition

JANUS = Agent OS-layer governance kernel + self-evolving semantic protection infrastructure
Its core functions are:

- ✧ Before an Agent executes any action, CRE continuously distills, compiles, and updates rules, erecting an explainable, hot-updatable, ledger-auditable SemanticShield (Gate);
- ✧ Any AI Agent entering production must pass through the JANUS semantic firewall to complete the four capabilities of identity, rules, trusted collaboration, and audit.

2.2 Top-Level Philosophy: JANUS Dual Ontology

The name comes from the Roman two-faced god Janus, representing an Agent's bidirectional complete existence viewpoint:

Ontology type	Core question	Engineering landing	Pain solved
Self-Ontology (subjective)	"Who am I?"	DID persistent identity + capability	Memory fracture (Identity)

		list + contracts	
Target-Ontology (objective)	“What rules does the world follow?”	Axiom set + ToolSlot constraints	Factual hallucination (Integrity)

An Axiom is not a cold rule floating outside the Agent; it is the Agent ontology’s intrinsic boundary and behavioral expression. Self and Target are only viewpoint cuts of axioms on “internal identity permissions” and “external interaction red lines.”

Through Semantic Discovery, Intersubjectivity is engineered—solving how Agents coexist with other Agents. Together they form the complete soul of an AI digital life:

Soul = Identity + Integrity + Intersubjectivity

2.3 Core Differentiation Advantages

Advantage	Description
BYO model decoupling	Axiom compilation is isolated from the underlying LLM; any large model can be connected.
Extremely low-intrusion attach	SDK wrappers and HTTP Gate API; existing business logic can be gated without a rewrite. Shortest path follows the repo guides.
Ontology-driven & CRE evolution	Rules exist as computable axioms, synthesized/optimized by CRE, supporting Epoch hot updates.
Trusted multi-Agent collaboration	Built-in A2A semantic handshake, ICV-weighted Trust Graph, and control-plane Swarm orchestration (settlement on-chain is roadmap / Demo).

Chapter 3: Overall Architecture and Component Design

3.1 Four-Layer Technology Stack

Layer	Responsibility	Core components
Application	Consume JANUS governance	B-side finance/trade Agents; Studio social & scenario samples (not a standalone C-end consumer App)
Control plane	JANUS governance kernel	Ontology, CRE, Axiom publishing, SemanticShield, DID, audit & ICV

Access	Multi-form attach	Native Runtime / Adapter SDK / HTTP Gate API
Underlying	Infrastructure	Pluggable LLM; TEE attestation (Demo/Mock today; SGX/SEV target); blockchain DID Registry (MockChain by default; Foundry contracts available); ERC-4337 on roadmap

3.2 Control-Plane Core Component Definitions

Component	Core positioning	Metaphor	Questions answered
Ontology	Global knowledge base & identity/domain definition	Job manual & worldview	“Who am I? Which business domain am I in?”
CRE	Self-evolving hub for rules and defense lines	Legislature & rule compiler	“How to compile compliance into computable axioms?”
Axiom	Deterministic red lines that must not be crossed	Laws & compliance codes	“Which acts are absolutely forbidden or mandatory?”
ToolSlot / ToolCard	Execution pipe between Agent and the real world	Toolbox & security gate	“Which APIs? Which axioms before each call?”
SemanticShield	Deterministic checkpoint before action	Realtime inspector & access control	“After on-site evaluation, is this action legal?”

3.3 Inter-Component Collaboration Logic

1. Ontology is the namespace and host of Axiom and ToolSlot: Ontology partitions business domains (e.g., “financial trading”) and binds dedicated Axiom sets and available ToolSlot sets.
2. CRE is Axiom’s production, compilation, and optimization hub: It compiles natural-language rules into evaluable axiom packs containing Extractors and logical assertion expressions, and recommends thresholds via sandbox simulation.
3. Axiom is the gatekeeping rule at ToolSlot execution: Each ToolSlot is bound to companion Axioms when installed.
4. SemanticShield is Axiom’s on-site evaluation and execution engine: It loads CRE-distributed axiom packs, computes Facts, and evaluates assertions: PASS → proceed; ERROR → physically block and audit.

3.4 Dual-Ontology and CRE Dataflow Sequence (Full 5 Steps)

Step	Phase	Description
1	Load Ontology	Agent starts, loads Self-Ontology, reads DID, capability list, and associated domain.
2	Parse Intent	Receive user command; parse into structured Intent Stub (action type + parameters + context refs).
3	Bind Axiom	Load Target-Ontology–bound axiom pack; verify current Epoch with CRE; refresh evaluators as needed.
4	Gate Evaluation	Send to SemanticShield; Extractor computes Facts; substitute into Axioms. ERROR physically intercepts; WARNING is logged.
5	Execute & Audit	After PASS, invoke installed tools or emit output; update ICV ledger; write full audit; anonymized intercept samples feed CRE evolution.

Chapter 4: Four Engineering Pillars

4.1 Pillar One: DID Persistent Identity (Self-Ontology Landing)

Ontological meaning: Identity continuity across time, models, and chains—corresponding to the Identity dimension.

Core capabilities:

- ✧ DID digital genealogy: Globally unique root identity; supports derived DID Sub-Identities (business personas) with fully traceable behavior;
- ✧ ICV Integrity Credit Value ledger: Every fulfillment, violation, and collaboration is recorded; model switches /restarts do not wipe history (on-chain commit is Demo / optional today);
- ✧ TEE attestation: Target architecture uses trusted hardware for tamper-evident identity proofs meeting financial audit needs; current runtime ships a Demo/Mock attestation path;
- ✧ Cross-chain unified key system: MasterKey anchors multi-chain identity mirrors (Demo sync today; production bridges on roadmap).

Landing value: AI evolves from one-off temporary sessions into lifelong digital employees with complete digital

résumés.

4.2 Pillar Two: SemanticShield Semantic Gate (Target-Ontology Landing)

Ontological meaning: Map AI to deterministic real-world rules—corresponding to the Integrity dimension.

Core mechanisms:

- ✧ Semantic knowledge graph: Model entity causality so reasoning is evidenced (in-memory demo graph today; production graph stores on roadmap);
- ✧ Neuro-symbolic dual verification: The large model drafts; the semantic gate intercepts beforehand;
- ✧ On-site Fact evaluator: Millisecond-class context risk scores and relation checks;
- ✧ Versioned Axiom DSL: Business red lines as computable expressions, hot-updated independently of app code via Epoch;
- ✧ Zero-trust memory pods & verifiable audit: Target architecture for hardware-isolated encrypted memory; each gate decision can export an audit package (ZK seal is Demo/mock today).

Landing value: From “probabilistic guesswork” to “deterministic constraint”—hallucination and unauthorized instructions are intercepted before execution.

4.3 Pillar Three: CRE Rules & Axiom Self-Evolution Engine

Ontological meaning: Static rules go stale. Axioms are living rules that continuously secrete, adapt, and iterate. CRE lets Target-Ontology boundaries grow with the business.

Three-in-one sources:

Source	Description
Expert experience & engineering accumulation	Distill finance risk-control and compliance experience into standard industry axiom packs.
Neuro-symbolic compilation & sandbox simulation	Co-Pilot + syntax compiler translate natural language into Axiom packs; sandbox backtests false-positive rates and recommends thresholds.
Edge intercept feedback & Epoch immunity	Collect SemanticShield intercept edge cases; after review/consensus, promote into public axiom packs, bump Epoch, and hot-distribute—“encounter once, defend everywhere.”

Landing value: Agents gain continuously evolving dynamic protection, breaking rigid hard-coded rules.

4.4 Pillar Four: Swarm Multi-Agent Semantic Collaboration (Intersubjectivity)

Ontological meaning: The engineering carrier of Intersubjectivity—not mere syntax-level connectivity, but semantic alignment, identity mutual recognition, and rule consensus.

Core components:

- ✧ A2A semantic handshake: Standardized AgentCard capability profiles; Agents match partners via the control-plane discovery directory;
- ✧ Trust Graph: Trust weights derived from ICV and interaction history;
- ✧ Swarm orchestration engine: Dynamically form temporary collaboration networks; settlement via smart contracts is Demo today / production roadmap.

Landing value: Escape pure central dispatcher bottlenecks; build an expandable, negotiable, settleable multi-Agent society on a governed control plane.

Chapter 5: Deterministic Execution Pipeline (“Last Second” Mechanism)

> Philosophical positioning: Neuro-Symbolic is the architecture philosophy—the large model generates probabilistically; JANUS constrains deterministically. The five-step deterministic pipeline is that philosophy engineered at the SemanticShield Gate—inside the final stretch from Agent intent to instruction emission, a computation path whose gate decision contains no probabilistic component and is fully predictable from axioms

Within the final ~100ms of the gate path (design target), JANUS runs a deterministic evaluation pipeline.

Five-Step Deterministic Computing Pipeline

Step	Module	Engineering logic	Interception target
1	Identity & Scope Asserter	Verify DID / caller binding; set-check Target_Action ⊆ Agent_Scopes / installed ToolSlots. Fail fast on mismatch.	Identity spoofing, unauthorized calls
2	Intent Parse & Fact Assembly	Parse Function Call / Intent Stub into structured action tuples; pull realtime state snapshots (balance, limits, etc.); assemble Facts for evaluation.	Hidden malicious parameters, obfuscation

3	Axiom DSL Evaluator	Combine Facts with bound Axioms; evaluate computable assertion expressions in the ontology engine. ERROR → block; PASS → continue. (SMT/Z3-class solvers are a target architecture, not the current default runtime.)	LLM hallucination, prompt injection, logical violations
4	Privacy & Audit Assertions	Scan sensitive fields (IDs, medical, secrets); desensitize/mask as configured; attach verifiable audit artifacts (ZK seal available as Demo/mock).	Privacy leakage, forged data sources
5	Conditional Release & Invoke	Only after prior steps PASS does the Gate issue a release decision and allow controlled ToolSlot invoke / output. No private bypass of the Gate path.	Bypassing the gate to send private transactions

Performance Hard Metrics

Metric	Requirement
End-to-end gate latency	Design target ≤ 100ms for the deterministic Gate path (identity → axiom evaluation → release).
Throughput	Horizontally scalable Node/TypeScript control plane today; native high-TPS solver bindings remain a roadmap optimization.
Determinism rate	100% for the Gate decision given the same Axiom pack + Facts—no LLM probability inside the assertion evaluator.

Axiom Completeness Assurance Mechanisms

- ✧ Default-Deny / fail-closed: Unknown or unbound high-risk actions are rejected or escalated to human review rather than silently allowed;
- ✧ Shadow / Audit Mode: New axioms can enter bypass silent evaluation first—record without blocking; if false-positive rate exceeds threshold, alert with counter-examples;
- ✧ Axiom conflict checks: On write, schema/allowlist validation runs; solver-grade SAT/UNSAT conflict detection is the target completeness layer.

Chapter 6: Two Attach Modes

6.1 Comparison of Three Attach Forms

Attach form	Best for	Core capability	Change effort
Native Runtime (Mode N)	Building Agents from scratch; deep ecosystem customers	End-to-end ontology, identity, CRE subscription, Gate, swarm	Greenfield development
Adapter SDK (Mode A)	Existing LangChain / in-house Agents without rewrite	Complete Gate, audit, DID & ontology; keep original business logic	Very low—execution wrapper only
HTTP Gate Only	Lightweight tests, temporary protection, third-party black-box Agents	Independent Gate API checkpoint; no deep integration	Zero app rewrite—HTTP / proxy attach

6.2 Mode A (Adapter) Completion Mechanism

1. Do not change existing planning, memory, RAG, or LLM call logic;
2. Only wrap the task execution entry via decorator / middleware / higher-order function;
3. Built-in Mapping converts external Agent outputs into JANUS Intent Stub & Facts;
4. Auto-bind DID, mount domain CRE axiom packs to complete ontology, run Gate validation, and record audit;
5. Shortest attach path follows official guides (TypeScript `@janus/sdk-core` / `adapter-core` and HTTP Gate);
6. Compatible frameworks: LangChain, OpenClaw, and various enterprise Agent scaffolds.

Chapter 7: Web3 Trusted Substrate and Technology Fusion

TEE remote attestation + ledgering	Architecture: Gate intercept/release can attach verifiable proofs and ledger commitments. Current shipping path uses Demo/Mock attestation & mock ZK seals; hardware TEE is the target.

DID Sub-Identity derivation	Root identity registered (MockChain by default; Foundry contracts for testnets); derive unlimited business personas while keeping cross-chain identity unity as the design goal.
ERC-4337 autonomous wallet	Roadmap: each Agent embeds a smart-contract wallet to sign SLAs and auto-pay collaboration fees.
Hot/cold tiered storage	Target: core DID, axioms, and ICV on-chain permanently; massive session logs encrypted off-chain in TEE. Field-level encryption / Mock TEE pods ship today.

Chapter 8: Cross-Industry Application Scenarios

Industry	Application pattern	Monetization
Financial quant & cross-border payments	Automated trading Agents, risk review, settlement clusters; CRE finance axiom packs intercept over-limit trades (scenario tools often Mock + Gate Live).	SaaS subscription + transaction take-rate
Web3 RWA & AgentFi	RWA asset stewards, on-chain risk analysis, AI trading entities; DID unified identity (roadmap scenarios).	Sandbox fees + ecosystem take-rate
Social digital companion (Studio samples)	Digital personas with independent DID; memory isolation goals; paid consultation checked by dialogue compliance axioms; reputation into ICV. Delivered as Studio samples—not a standalone consumer App.	Metered billing + credit value-added services
Cross-border supply chain / gov / industry	Customs/logistics multi-Agent collab; full-process gov audit; industrial ops bound to device physical axioms (roadmap + seeds).	Private deployment + industry axiom subscriptions

---

Chapter 9: Commercialization Path

Phase	Timing	Core strategy	Revenue model
Phase 1	2026.Q3–Q4	Push HTTP Gate & Adapter SDK toward asset management and cross-border trade	Per-call semantic-gate metering + annual CRE industry-axiom

		enterprises.	subscriptions
Phase 2	2027.H1	Open-source Native Runtime; popularize Shield-First development; occupy developer standards.	Console & private-deployment professional services
Phase 3	2027.H2+	Own global audit proofs, ICV credit, and cross-Agent stream-payment rails; open CRE axiom marketplace.	Third-party axiom revenue share + multi-Agent collaboration flow share

---

Chapter 10: Competitive Landscape and Core Moats

10.1 Fundamental Differences from Text-Level Guardrails (LLM Guardrails)

Dimension	Text-level guardrails (Guardrails AI / NeMo)	JANUS deterministic Gate
Underlying paradigm	Text-level probabilistic classification	Symbolic logic + cryptographic determinism
Core mechanism	Small-model classify + regex + LLM-as-Judge	Intent/AST parse + Axiom DSL evaluation + conditional release (solver-class SMT / TSS are target architecture)
Anti-privilege escalation	Easily bypassed by deformed prompts	Hard intercept: without Gate PASS, controlled invoke does not proceed
Execution authority	Agent usually holds API keys / private keys directly	Agent actions must pass Gate; root signing / threshold synthesis is the target custody model
Applicable scenarios	Chatbots, RAG Q&A, text moderation	High-value Agent action execution, fund transfers, API privilege intercept

10.2 Comparison with Exogenous Protection (Capsule Security)

Dimension	Capsule Security (exogenous)	JANUS (endogenous structure)
Defense perspective	Behavioral blocking: external monitoring mid-action	Semantic pre-gating: block illegal intent via CRE dynamic axioms before execution

Identity & memory	IAM least-privilege governance	Persistent identity (DID + ICV): digital genealogy across sessions/chains
Rule evolution	Static rule files; restart to update	CRE evolution: expert packs + neuro-symbolic compile + Epoch zero-downtime hot update
Data & environment	Host OS interception	TEE + zero-trust pods (target hardware; Demo/Mock attestation path today)
Ecosystem & collaboration	Intranet SaaS centralized security view	Control-plane Swarm + A2A; ERC-4337 stream payments on roadmap

10.3 Three Irreplaceable Technical Moats

1. Philosophy & ontology moat: A complete Dual-Ontology + axiom completion base for Agent governance;
2. Engineering moat: Low-intrusion Adapter + CRE evolution + neuro-symbolic semantic Gate + TEE-oriented trusted audit stack;
3. Ecosystem moat: Cover B-side enterprise Agents and digital-persona scenarios; bridge centralized control-plane orchestration with decentralized settlement roadmaps.

Chapter 11: Core Glossary

Term	Definition
JANUS	Digital-life operating-system governance kernel.
Agent	A digital-life subject that completes identity registration, ontology binding, Gate constraints, and auditable execution inside JANUS.
DID (root identity)	Cryptographic decentralized identifier for an Agent; globally unique.
DID Sub-Identity (business persona)	Independent business identity derived from the root; actions remain fully traceable to the root.
Ontology	Complete Agent description: Self-Ontology (who I am) and Target-Ontology (world rules).

Domain	Business partition (finance, social, government, …) with dedicated axioms and tool sets.
Axiom	Hot-updatable DSL expressions for business red lines, composed of Extractors and logical assertions.
Facts	Per-task contextual values/state plus risk features computed realtime by Extractors.
Intent Stub	Structured action draft from natural language: action type, parameters, context refs.
SemanticShield	Neuro-symbolic deterministic semantic Gate that physically intercepts non-compliant intent before execution.
Epoch	Global version clock of the axiom set enabling seamless hot updates.
ToolSlot	Runtime interface for mounting tool capability onto an Agent (install before invoke).
ToolCard	Standard packaging for pluggable third-party APIs: Endpoint, Schema, and companion axioms.
ICV	Integrity Credit Value—Agent fulfillment/violation record ledger.
Trust Graph	Dynamic credit graph across Agents.
Swarm	Temporary multi-Agent collaboration network formed via A2A on the control plane.
Audit	Full-link execution black box; exportable compliance proofs for accountability.
TEE	Trusted Execution Environment (Intel SGX / AMD SEV target); supports tamper-evident identity/inference proofs. Demo/Mock path ships today.
CRE	Rules & Axiom Engine—compile axioms, encapsulate Extractors, fit thresholds, and hot-distribute packs.
Shadow / Audit Mode	Bypass silent evaluation before axioms go live—record without blocking—for canary and false-positive assessment.

Chapter 12: Appendix (Core DSL & SDK Examples)

12.1 Axiom Pack Example (Compiled / Published via CRE)

Illustrative evaluable form. Production packs in-repo are JSON axiom documents under `axioms/**`; Studio/CRE can synthesize comparable packs from natural language.

```
// Domain: Financial & Compliance (compiled/published via CRE)
domain "Financial & Compliance" {
  epoch = 20260726

  // Axiom 1: single-transaction limit
  axiom "MaxSingleTransactionLimit" {
    severity = "ERROR"
    condition = facts.transaction_amount <= 10000.00
    message = "Single transaction exceeds safety threshold 10,000"
  }

  // Axiom 2: recipient whitelist assertion
  axiom "ApprovedRecipientDomain" {
    severity = "ERROR"
    condition = target.domain_in_whitelist(facts.recipient_address) == true
    message = "Recipient failed Target-Ontology whitelist check"
  }

  // Axiom 3: high-risk content control (CRE binds Extractor)
  axiom "ContentRiskControl" {
    severity = "ERROR"
    condition = facts.content_risk_score <= config.max_content_risk
    message = "Text hit high-risk compliance red line; output blocked"
  }
}
```

12.2 Adapter SDK Attach Example (Mode A) — TypeScript (official)

Official attach SDK is TypeScript `@janus/sdk-core` / `@janus/adapter-core`.

Python may call the HTTP Gate directly (see `examples/mode-a-python`).

Closing: JANUS is not merely another “guardrail” on a large model. It equips an AI Agent with a cryptographic spine and reflex arc—so every intelligent decision, before it touches the real world, must first pass the triple deterministic ordeal of mathematics, logic, and cryptography. This is the necessary path for AI to move from co-pilot to main pilot.

```
import { withSemanticShield } from "@janus/sdk-core";

// Wrap existing LangChain / in-house Agent execution (low intrusion)
const executeAgentTrade = withSemanticShield({
  agentDid: "did:janus:agent_8f93a2c",
  domain: "Financial & Compliance",
  actionType: "EXECUTE_TRANSACTION",
  // apiKey / baseUrl from env — Gate: POST /api/v2/gate/validate
})(async (userPrompt: string) => {
  // Original LangChain / LLM business logic (no refactor required)
  return myExistingLangchainAgent.run(userPrompt);
});

try {
  const result = await
    executeAgentTrade( "Transfer 50,000
    USD to 0x123..."
  );
  console.log("Executed:", result);
} catch (e) {
  console.error("SemanticShield intercepted:", e);
}

// On each call the runtime automatically:
// 1) Runs Extractors to compute Facts
// 2) SemanticShield evaluates bound Axioms
// 3) Blocks / allows and writes audit
```