Agent Trust Policy Language

Executive Summary

As AI continues to reshape industries, so too must the security paradigms be governing it. The **Agent Trust Policy Language (ATPL)** redefines how trust is codified, enforced, and audited in AI-driven systems. Designed as a declarative, schema-bound trust language for agent-based architectures, ATPL transforms Zero Trust from a conceptual model into a functional and enforceable policy layer.

By leveraging ATPL, organizations can implement **composable, identity-aware, and behavior-scored trust policies**, fully aligned with Zero Trust principles and inspired by the NIST AI RMF. Each agent becomes a policy-bound entity whose actions, access, and privileges are governed not by static assumptions but by contextual evaluation.

The Urgency of Zero Trust for AI Systems

Zero Trust has become a cornerstone of modern cybersecurity, emphasizing the principle of *trust no one, verify everything*; however, the application of these principles to AI presents unique opportunities and challenges:

- **Dynamic Threats:** AI systems face novel attack vectors, such as adversarial inputs and data poisoning, which exploit the inherent complexity of machine learning.
- **Ethical Accountability:** AI models can inadvertently perpetuate biases or make opaque decisions, undermining trust.
- **Evolving Compliance Needs:** Regulatory bodies may require dynamic safeguards not pertinent to all agentic systems.

ATPL attempts to address these risks by creating a dynamic language to support resolving these issues programmatically. Coupled with a truly Zero Trust architecture,

Core Concepts

This paper introduces the concept of ATPL but also Zero Trust Access Tokens, which are time and situationally aware access tokens with a limited temporal or operational lifetime. ZTATs are bounded by the system, and are able to be revoked at any time. ATPL is built with the assumption. In some architectures this may be seen as a Just In Time request for escalated privileges; however, ZTATs extend beyond access claims (like JWTs) and reflect privileges on a system or systems. Operations can require ZTATs, for example. We can also be dynamic and adjust depending on your security architecture. If a user's activity is deemed risky or requires a human in the loop (HITL) but one is not available, an agent can be a real time monitor. That real time monitor can be an intercept of operations and delegate authority to the user or revoke if a

risk score is deemed unnecessarily high.

The nirvana fallacy dictates that break glass is an exception mode; however, ATPL and ZTATs treat normal day to day the same as break glass. Stated in reverse, access may be limited to systems only during “break glass operations” but the security therein should not be any more limited due to the severity of an incident and need to “Just fix it” All Zero trust access policies should govern not just access to the system but also each granular operation.

With ATPL the source of trust for ZTATs can be agents, services, and humans. The source of truth for the policies will be configuration management.

Core Objectives

AI systems no longer operate as monoliths; they are composed of interdependent agents performing specialized tasks. These agentic systems increase the attack surface, amplify complexity, and complicate trust assumptions. Traditional Zero Trust frameworks lack the granularity and agent-awareness needed to manage these environments.

ATPL addresses these gaps:

- Enables **granular trust enforcement** per agent
- Supports **capability composition** for scoped access control
- Provides a **trust scoring engine** based on identity, provenance, behavior, and runtime
- Allows for **zero trust access token delegation (ZTAT)** to enforce temporary, contextual privileges

The framework is structured around four principles, each addressing a critical aspect of AI security**:**

- Continuous validation of all entities and interactions. A prototypical element of Zero Trust Frameworks, this will be evolved for agentic workloads as each agent itself a trusted entity with its own identity. Access can be revoked at any time by human or an agent, if so delegated.
- Decisions must be tracked and measured throughout the lifecycle of the agents.
- Limiting access to reduce risk. Since each agent has its own identity, access should be partitioned at the most granular level.
- Observables must be in place to ensure outcomes arrive at decisions within bounds.

Building on the NIST AI RMF, the Agent Trust Policy Language is adaptable, modular, and actionable. It aligns with global standards and regulations, enabling organizations to achieve compliance while addressing industry-specific risks. This will be discussed in more detail in the next section.

Zero Trust for Agents

Despite its transformative potential, the adoption of AI introduces a spectrum of risks. Unlike traditional software systems, AI models are inherently probabilistic, making them vulnerable to:

- Adversarial Manipulation: Malicious actors can exploit AI through adversarial inputs, undermining its reliability.
- Data Vulnerabilities: As data fuels AI, breaches or tampering can have cascading effects.
- Opaque Decision-Making: Complex AI models often operate as "black boxes," limiting accountability.

Addressing AI Risks with ATPL

The transformative power of artificial intelligence is matched by the unique challenges it introduces. Unlike deterministic software systems, AI operates in probabilistic and adaptive spaces, opening the door to new categories of risk. ATPL (Agent Trust Policy Language) is designed to confront these challenges head-on, enabling enforceable, transparent, and composable trust policies that govern AI behavior in real time.

Adversarial Manipulation

AI systems can be subtly manipulated through adversarial inputs — data crafted to exploit the system's learned behavior. ATPL mitigates this by enabling fine-grained input validation rules and context-aware policy enforcement. Trust policies can define acceptable input domains, detect patterns consistent with adversarial attacks, and trigger mitigative actions such as:

- Isolating suspicious agent actions
- Requiring real-time human oversight
- Rolling back or quarantining outputs pending approval

This creates a zero-trust posture for AI input processing, preventing blind trust in upstream data.

Data Vulnerabilities

Since data forms the backbone of AI decision-making, any compromise in training or inference data poses systemic risks. ATPL enforces data provenance and integrity checks as part of its runtime policy evaluation. Trust policies can require:

- Verification of data lineage before model execution
- Integrity signatures on data sets or model weights
- Role-based constraints on who or what can feed data into an agent

With ATPL, policies aren't static — they adapt in real-time to data context, enabling dynamic defenses against data poisoning or unauthorized injections.

Opaque Decision-Making ("Black Box" AI)

The complexity of AI models often obscures their reasoning, making it difficult to audit or explain outcomes. ATPL introduces structured explainability requirements directly into the execution environment. Trust policies can mandate:

- Generation of traceable reasoning chains
- Logging of model activations or decision paths
- Disclosure of uncertainty/confidence levels

This ensures that every AI action is observable, attributable, and governed by explainable rules, fostering trust and compliance even in high-stakes domains.

Zero Trust is a security model based on the principle of "never trust, always verify." It assumes that threats can originate from anywhere—internal or external—and applies rigorous verification processes to every interaction. Core tenets of Zero Trust include ensuring least privileged access, continuous verification, and adaptive security.

Systems often validate identities, but ATPL differs in that its objective is to ensure all operations are trusted and within bounded reason. There can be reasonable and actionable gradients of trust, so trust is a score based on a weighted score. ZTATs, or Zero Trust Access tokens are time and situationally aware access tokens with a limited temporal or operational lifetime. ZTATs are bound by the system, and are able to be revoked at any time. ATPL is built with the assumption

Applying Zero Trust to AI

AI systems no longer operate as monoliths; they are composed of interdependent agents performing specialized tasks. These agentic systems increase the attack surface, amplify complexity, and complicate trust assumptions. Traditional Zero Trust frameworks lack the granularity and agent-awareness needed to manage these environments.

ATPL addresses these gaps:

- Enables **granular trust enforcement** per agent
- Supports **capability composition** for scoped access control
- Provides a **trust scoring engine** based on identity, provenance, behavior, and runtime
- Allows for **zero trust access token delegation (ZTAT)** to enforce temporary, contextual privileges

Analysis of Alternatives (AoA): Why ATPL Stands Apart

When evaluating strategies to secure agentic AI systems, multiple approaches exist, from augmenting traditional role-based access control (RBAC) to applying runtime security policies or building proprietary enforcement layers around AI workloads. The following Analysis of Alternatives (AoA) outlines how ATPL compares to these strategies.

Traditional RBAC and Static Policy Enforcement

Pros:

- Simplicity and familiarity in enterprise systems
- Mature tooling and policy frameworks

Cons:

- Inflexible in dynamic AI environments
- No support for behavior-based scoring
- Cannot express composable or conditional capabilities
- Lacks support for temporary trust delegation (ZTAT)

ATPL Advantage: ATPL enables identity-aware and conditional policies that evolve with runtime context, replacing static roles with evaluative trust expressions.

Agent Sandboxing and Isolation Techniques

Pros:

- Contain untrusted code
- Reduce lateral movement

Cons:

- Do not address identity, behavior, or history of the agent
- Limited visibility into intent or access purpose
- High operational overhead and performance cost

ATPL Advantage: ATPL builds layered defense through trust scoring and scoped capabilities, reducing need for full isolation while maintaining control and transparency.

AI Runtime Monitoring with Heuristics

Pros:

- Real-time detection of anomalies
- Integration with observability platforms

Cons:

- Limited preemptive controls (detects after-the-fact)
- Can be bypassed by sophisticated adversaries
- Not declarative or explainable to auditors or compliance teams

ATPL Advantage: ATPL offers proactive trust enforcement before action execution, backed by audit-ready YAML-defined policies. Monitoring can still be used for post-hoc scoring.

Custom Policy Engines Built In-House

Pros:

- Tailored to internal environments
- Full control of logic

Cons:

- High maintenance cost and engineering complexity
- Poor interoperability with external compliance frameworks
- Risk of misalignment with evolving AI risk management standards

ATPL Advantage: ATPL is schema-validated, NIST-aligned, and designed for broad compatibility with modern AI and zero trust systems. It supports YAML-based policy versioning and open schema extensions.

Alignment with Existing Frameworks

The Agent Trust Policy Language builds upon and complements established standards and frameworks that address AI risk, cybersecurity, and compliance. By aligning with existing guidelines, the framework ensures its principles are actionable and interoperable within existing organizational structure.

NIST AI Risk Management Framework (AI RMF)

Alignment with the NIST AI Risk Management Framework (AI RMF)

ATPL naturally maps to and extends the core functions of the NIST AI RMF: **Govern, Map, Measure, and Manage.**

Govern

- **Codified Trust Policies**: ATPL introduces explicit, versioned policies for agents.
- **Enforceable Governance**: Each interaction is evaluated through a declared policy — not through inference or assumption.

Map

- **Agent Metadata and Tags**: Enables understanding of who/what the agent is, its role, and the scope of its requests.
- **Context-Aware Classification**: Uses contextual match conditions to map policies to agent functions. Cryptographic means should be used, when possible, to align these mapped policies to agent identities.

Measure

- **Trust Scoring System**: Quantitative trust assessment aligned with NIST's emphasis on measuring risk.
- **Audit Logs and Capability Traces**: Every action can be traced, reviewed, and scored over time.

Manage

- **ZTAT (Zero Trust Access Token Delegation)**: Enables temporary, risk-aware access aligned to runtime trust posture. Works at the principle of "revocation at any time"

Declarative Enforcement: Policies evolve without code changes, enabling flexible risk response

Technical Foundations: ATPL Policy Schema

Note: The schema is an evolving document. It is located at https://sentrius.io/schemas/atpl.schema.json

Foundationally ATPL is a declarative schema defined in YAML, enabling developers and security teams to articulate trust policies in human-readable, auditable formats. This schema can be validated using JSON Schema tooling, ensuring enforcement consistency across environments.

Each ATPL policy includes the following key sections:

`policy_id`, `version`, and `description`

Provides metadata for versioning, identification, and documentation. Policies should be uniquely versioned to support traceability.

`match`

Defines criteria to associate this policy with an agent. Match fields typically include tags, classifications, or runtime attributes (e.g., `env:prod`, `agent_type:extractor`).

```
match:
  agent_tags:
    - "env:prod"
    - "classification:data-transformer"
```

`identity`

Specifies identity constraints that must be met. This includes:

- `issuer`: trusted certificate or identity provider
- `subject_prefix`: expected agent ID format
- `mfa_required`: true/false flag for two-factor verification
- `certificate_authority`: name or hash of root CA

`provenance`

Outlines the build and release chain expectations:

- `source`: expected build system or CI pipeline
- `signature_required`: whether a code signature must be validated
- `approved_committers`: list of email or GPG IDs

runtime

Describes runtime requirements:

- `enclave_required`: boolean value indicating trusted compute requirement
- `attestation_type`: e.g., `aws-nitro`, `intel-sgx`
- `verified_at_boot`: indicates if measured boot was attested

behavior

Applies historical and behavioral constraints:

- `minimum_positive_runs`: number of successful sessions without violations
- `max_incidents`: threshold of past trust violations. The usage can be implementation dependent
- `incident_types`: types of policy violations considered (e.g., denylist). These may be standardized and added to the schema.

capabilities

Defines allowable actions, separated into primitives and composed sets:

```
capabilities:
  primitives:
    - id: "readLogs"
      description: "Read access to logs"
  composed:
    - id: "observe"
      includes: ["readLogs", "streamMetrics"]
```

trust_score

Calculates the trustworthiness of the agent in runtime:

```
trust_score:
  minimum: 75
  weightings:
    identity: 0.3
    provenance: 0.2
    runtime: 0.3
    behavior: 0.2
```

actions

Specifies what to do at various ttrrust thresholds:

```
actions:
  on_success: "allow"
  on_marginal:
    action: "require_ztat"
    ztat_provider: "ztat-service.internal.svc"
  on_failure: "deny"

ztat
```

A dedicated section for configuring Zero Trust Access Token delegation. This section defines the token provider, lifetime, cryptographic bindings, and optional manual approval workflows.

```
ztat:
  provider: "ztat-service.internal.svc"
  ttl: "5m"
  approved_issuers:
    - "sentrius"
    - "trust-oracle"
  key_binding: "RSA2048"
  approval_required: true
```

ZTAT Fields:

provider: Service endpoint or URI responsible for issuing tokens

ttl: Time-to-live duration for the access token

approved_issuers: A list of identities allowed to issue ZTATs

key_binding: The cryptographic key type expected (e.g., RSA2048, ECDSA)

approval_required: Boolean indicating whether manual or external approval is needed

This schema is designed for composability and continuous evaluation. Any field may be extended, and future versions will support temporal constraints, context-aware conditions, and in-policy observability hooks. Please note that the mechanism for ZTAT delivery and delegation aren't defined by the ATPL specification. That will be implementation dependent for the time being. Our open-source project (Sentrius) has APIs that deliver the access tokens to external services through its own architecture. The fact of requiring ZTATs and the providers must be known to agents. While the authors believe that Access Tokens should be standardized, that is out of the scope of ATPL.

Conclusion

ATPL delivers a **functional Zero Trust foundation for AI agents**, transforming abstract principles into enforceable policies. Through composable capabilities, runtime scoring, and declarative enforcement, AI control planes become policy aware.

This framework not only strengthens security but builds a pathway for **trusted, explainable, and governable AI** in high-risk domains. ATPL is a blueprint for the future of secure agentic systems.