Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 5, 2026

D. Larsson
Agentflow
Stockholm, Sweden

5 April 2026

Agent Identity, Trust and Lifecycle Protocol (AITLP)

draft-larsson-aitlp-00

This document is an Individual Submission to the IETF. It has not been adopted by any IETF Working Group.

Abstract

This document defines the Agent Identity, Trust and Lifecycle Protocol (AITLP), a protocol for autonomous software agents operating within organizational boundaries. AITLP specifies agent identity and naming conventions, hierarchical mandate enforcement, lifecycle state management, inter-agent trust verification, ontological scope constraints, certificate-based authentication, and generational knowledge transfer via Agent Legacy Mode (ALM).

The protocol enables agents to operate autonomously while remaining auditable, revocable, and organizationally bounded. AITLP focuses on boundaries -- what an agent is not permitted to do and how that is enforced -- complementing capability-focused specifications such as MCP, A2A, and ANS.

Unlike those specifications, AITLP defines not a tool or a communication channel, but an organizational actor: an agent with a declared identity, a bounded mandate, a managed lifecycle, and a cryptographically enforced scope of authority.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at https://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at https://www.ietf.org/shadow.html

This Internet-Draft will expire on October 5, 2026.

Copyright Notice

Table of Contents

1. Introduction

The proliferation of autonomous AI agents in industrial and enterprise systems has created a need for standardized identity and governance. Existing specifications address discovery (ANS [ANS]), communication (A2A [A2A], ACP), and tool integration (MCP [MCP]), but none govern the organizational structure in which agents operate: how they are created, supervised, scoped, evaluated, and retired.

In production deployments, agents do not exist in isolation. They operate within organizational hierarchies, are granted specific mandates, reason over domain ontologies, and accumulate operational knowledge. That knowledge is currently lost at termination.

AITLP addresses this gap. It specifies the organizational layer for autonomous agent systems: identity, hierarchy, mandate, lifecycle, and intergenerational knowledge transfer.

1.1. Design Principles

- Boundaries over capabilities. AITLP defines what agents are not permitted to do, not only what they can do. A named, scoped, mandated agent is safer than a capable but unbounded one.

- Hierarchy as governance. Organizational structure is not administrative overhead -- it is the primary security mechanism. An agent cannot exceed the authority of its parent.

- Names as contracts. The agent URI encodes organizational position, domain, and jurisdiction. The name is the contract.

- Knowledge compounds. Operational knowledge accumulated by one agent

generation is transferred to the next via Agent Legacy Mode, analogous to how human institutions preserve institutional memory.

- Human primacy is non-negotiable. AITLP treats agent principals as first-class cryptographic identities subject to the same auditability and mandate enforcement as human principals -- while preserving the primacy of human oversight as required by GDPR Article 22, EU AI Act Article 14, and NIS2. Humans retain ultimate authority at every layer of the protocol.

1.2. AITLP as an Operating System Layer

AITLP provides the kernel-layer primitives for autonomous agent systems: identity, mandate enforcement, and lifecycle management -- analogous to how an operating system kernel provides process identity, memory protection, and lifecycle management for software processes.

The analogy is architecturally precise:

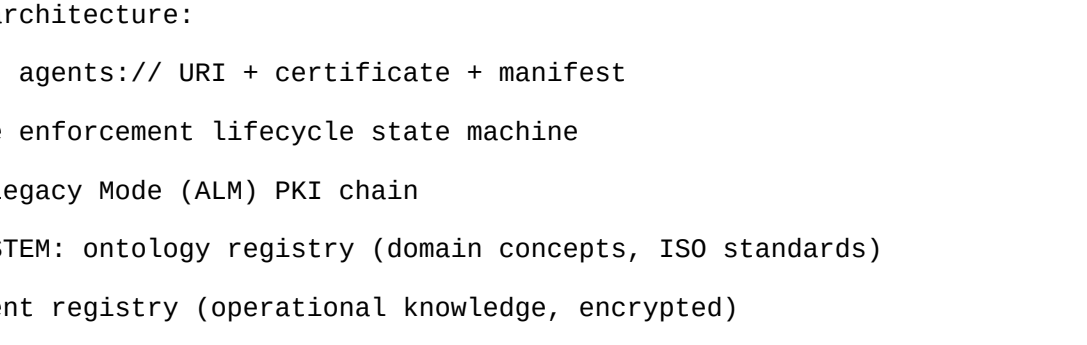
- Kernel -- agents:// / AITLP: Provides foundational primitives: agent identity (URI + certificate), mandate enforcement (what a process is permitted to do), and lifecycle management (start, run, checkpoint, retire). No application runs without kernel primitives; no agent operates without AITLP identity.

- Filesystem -- Ontology and Testament Registry: Persistent structured storage that survives individual process lifetimes. The ontology registry maps domain concepts across agents and organizations. The testament registry stores accumulated operational knowledge across agent generations -- the institutional memory layer.

- Shell -- Agentflow and compatible platforms: The interface layer where humans and agents interact. Workflow canvases, approval flows, lifecycle monitors, and audit dashboards constitute the shell. Humans operate at the shell layer; agents operate at all three layers simultaneously.

This three-layer model has a critical governance implication: no single entity owns the kernel. The agents:// root registry is governed as a public good, analogous to how the Linux kernel is governed by the Linux Foundation. Any organization may implement AITLP-compliant agents; any approved CA may issue certificates; any domain expert may publish ontology modules.

AITLP treats agent principals as first-class cryptographic identities -- subject to the same auditability, mandate enforcement, and certificate governance as human principals. This enables autonomous multi-agent systems to be auditable at scale. However, human primacy is structurally enforced, not optional: agents with risk_level: high MUST have human_oversight: true; GDPR Article 22 rights apply to any agent decision with legal or significant effect; and EU AI Act Article 14 human oversight requirements are satisfied by the supervised agent approval flow built into the AITLP shell layer.



audit trail (tamper-proof, NIS2/GDPR compliant)

SHELL: workflow canvas approval flows (GDPR Art. 22)

lifecycle monitor health endpoints

human-in-the-loop interfaces

1.3. Relationship to Existing Protocols

- ANS: AITLP manifests are valid ANS registry entries. ANS provides discovery; AITLP provides organizational governance. The protocols are complementary.

- A2A: A2A governs communication between agents. AITLP governs their identity and authority before communication begins. An AITLP-verified agent can use A2A for messaging.

- MCP: AITLP mandate sections reference MCP tool permissions. The allowed_tools and forbidden_tools fields in the agent contract map to MCP tool declarations.

- 3GPP / 5G Core (out of scope): AITLP does not specify integration with 3GPP authentication architectures (5G-AKA, EAP-AKA’’, SUPI/SUCI identity models, NRF, or NSSAf). Deployments in which AITLP-governed agents operate within or adjacent to 5G or 6G core networks -- for example, agents performing network slice management, SLA enforcement, or RAN-adjacent inference -- will require a separate bridging specification mapping AITLP certificate identity to 3GPP subscription identifiers. Such a bridging specification is outside the scope of this document. The AITLP Stewardship Council (Section 12A.2) is the appropriate body to coordinate such work with 3GPP SA3 and ETSI NFV-SEC.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Agent: An autonomous software entity that perceives inputs, reasons, and takes actions, identified by an AITLP URI.

Manifest: A signed JSON document encoding an agent’s identity, mandate, ontology, hierarchy, and lifecycle state.

Contract: The mandate section of the manifest, specifying permitted and forbidden actions, tool access, data scope, and spend limits.

Principal: The entity (human or agent) that issued the agent’s certificate and is responsible for its behavior.

Namespace: The top-level organizational domain (e.g., agents://maritime).

Scope: The set of ontological entities an agent may reason over.

Seniority: The hierarchical level of an agent: partner, senior, associate, or intern.

Testament: A structured document written by an agent at retirement, encoding accumulated knowledge for inheritance by a successor.

Agent Legacy Mode (ALM): The final lifecycle phase during which an agent prepares its testament.

Generation: An integer version counter incremented each time an agent is replaced by a successor that inherits its testament.

Operating System Layer: The three-tier AITLP architecture: kernel (agents:// protocol and certificate infrastructure), filesystem (ontology and testament registry), and shell (human and agent interaction interfaces). Designed for autonomous agent organizations on the same principles as conventional operating system architecture, with structural human oversight enforcement at the shell layer.

Human Primacy: The AITLP principle that human principals retain ultimate authority over agent systems at all times. Structurally enforced via mandatory human_oversight for high-risk agents, GDPR Article 22 approval flows, and EU AI Act Article 14 compliance. Human primacy is a protocol requirement, not an implementation choice.

Mandate Drift: The gradual expansion of an agent's actual behavior beyond its declared contract, detected through behavioral monitoring.

3. URI Scheme and Naming

3.1. Syntax

AITLP agents are identified by URIs of the following form:

```
aitlp-uri = scheme "://" org "." domain "." role "_" jurisdiction
["@v" generation]

scheme = "agents" / "agent" / "agent-dev"

org = 1*ALPHA *("-" / ALPHA / DIGIT)

domain = 1*ALPHA *("-" / ALPHA / DIGIT)

role = 1*ALPHA *("-" / "_" / ALPHA / DIGIT)

jurisdiction= 2*ALPHA ; ISO 3166-1 alpha-2 or regulatory tag

generation = 1*DIGIT
```

Example URIs:

```
agents://Acme.Ekonomi.Fakturahanterare_SE@v3

agents://maritime.RouteOptimizer.WeatherReader_EU@v1

agents://Acme.IT.SecurityScanner_EU@v2
```

3.2. Name as Contract

The URI is not merely an address. Each component carries semantic and governance meaning:

- org: The organizational namespace. Defines which CA issues certificates in this space.
- domain: Maps to a registered ontology node. The agent MUST only reason over concepts within this domain.
- role: Defines the agent's function. The name is the scope. An agent named Fakturahanterare (Invoice Handler) MUST NOT approve payments.
- jurisdiction: Specifies applicable regulatory framework and data residency. Relevant for GDPR, NIS2, and sector-specific compliance.
- generation (@v): Enables testament inheritance. Generation N+1

inherits the distilled knowledge of all preceding generations.

Agents MUST have unique URIs within their organizational namespace. The issuing CA MUST reject certificate requests that would create duplicate URIs.

3.3. Protocol Variants

- agents:// -- Verified. Valid signed manifest required. Suitable for production.
- agent:// -- Unverified. No manifest required. MUST NOT access production data.
- agent-dev:// -- Sandbox. MUST be isolated from production systems.

A verified agent (agents://) MUST refuse connections from unverified agents (agent://) unless explicitly permitted by its mandate.

4. Agent Manifest and Contract

4.1. Overview

Every agents:// agent MUST present a valid manifest. The manifest is signed using JSON Web Signature (JWS, RFC 7515) by the issuing CA. The contract within the manifest is the operative governance document.

4.2. Full Manifest Schema


```
  {
  "agent_id": "agents://Acme.Ekonomi.Fakturahanterare_SE@v3",
  "protocol": "aitlp/1.0",
  "issued_by": "agents://Acme.Admin.PrincipalBroker",
  "issued_at": "2026-03-16T08:00:00Z",
  "expires_at": "2026-06-16T08:00:00Z",
  "fingerprint": "sha256:a3f9b2...",
  "contract": {
  "version": "1.2.0",
  "goals": {
  "primary": "Validate and process incoming invoices",
  "forbidden": ["Approve payments", "Modify supplier records"]
  },
  "input_spec": {
  "required": ["invoice_id", "amount", "supplier_id"],
  "max_amount_eur": 50000
  },
  "output_spec": {
  "format": "structured_json",
```

```
"requires_sources": true
},
"security_boundaries": {
"allowed_tools": ["read_invoice", "validate_vat", "flag_for_review"],
"forbidden_tools": ["approve_payment", "delete_record"],
"data_access": ["Ekonomi.Faktura", "Ekonomi.Leverantor"],
"max_api_calls_per_hour": 1000,
"session_max_cost_eur": 0.10
},
"seniority": "associate",
"requires_approval_above_eur": 10000
},
"ontology": {
"namespaces": ["ekonomi.se", "eu.gdpr"],
"scope_nodes": ["Ekonomi.Faktura", "Ekonomi.Leverantor"],
"forbidden_nodes": ["Ekonomi.Lon", "IT.*"],
"compliance": ["eu.gdpr", "eu.nis2"]
},
"hierarchy": {
"level": "associate",
"reports_to": "agents://Acme.Ekonomi.CFO_SE",
"supervises": [],
"risk_level": "minimal",
"human_oversight": false,
"four_eyes_above": 10000
},
"lifecycle": {
"status": "active",
"born_at": "2026-01-10T00:00:00Z",
"trained_by": "agents://Acme.Ekonomi.CFO_SE",
"generation": 3,
"performance_score": 0.94,
"review_at": "2026-06-16T00:00:00Z",
```

```
"clearance_level": "CONFIDENTIAL"

},

The testament section is OPTIONAL. It is present only after Agent
Legacy Mode (ALM) has been activated (Section 11).

"testament": {

"inherited_from": "agents://Acme.Ekonomi.Fakturahanterare_SE@v2",

"checksum": "sha256:b7e2...",

"lessons": ["Supplier X consistently sends malformed VAT fields"],

"unresolved": ["High invoice volume on last day of month"],

"recommend_hire": ["agents://Acme.Ekonomi.VATSpecialist_SE"],

"written_at": "2026-03-01T00:00:00Z"

},

The llm section is OPTIONAL. Where present, it MUST be validated by
the routing layer before model invocation (Section 12).

"llm": {

"complexity": "simple",

"preferred_model": "claude-haiku-4-5",

"fallback_model": "gemini-2.5-flash",

"max_cost_per_call_eur": 0.001

},

The inference_hardening section is OPTIONAL for clearance_level
UNCLASSIFIED; REQUIRED for clearance_level RESTRICTED and above
(Section 13.7).

"inference_hardening": {

"system_prompt_hash": "sha256:a3f9b2...",

"input_sanitization_required": true,

"sanitization_agent": "agents://Acme.Security.InputSanitizer_SE",

"injection_pattern_registry":
"https://registry.aitlp.org/injection-patterns/v1",

"output_schema_enforcement": true,

"max_input_tokens_per_request": 32768,

"system_prompt_confidentiality": true

}

}
```

5. Hierarchical Organization

5.1. Seniority Levels

The seniority taxonomy deliberately mirrors the hierarchical structure used in professional services organizations globally -- including management consulting, legal, and audit firms -- where the partner/senior/associate/intern model is widely understood across industries and jurisdictions. This design choice makes AITLP agent hierarchies immediately legible to the organizational principals who deploy and govern them, without requiring translation to an abstract access control vocabulary. The taxonomy maps to RBAC role hierarchies and is compatible with XACML policy structures, but uses organizationally meaningful labels rather than technical identifiers.

- partner: Strategic decision-making. High autonomy. MAY create and terminate agents. MAY operate with human_oversight: false when explicitly approved.

- senior: Operational management. MAY supervise associates and interns. MAY recommend agent creation to principal.

- associate: Task execution. MUST NOT create agents unless contract explicitly permits. Default seniority.

- intern: Probationary. All consequential actions MUST be reviewed by supervising agent before execution.

5.2. Mandate Inheritance

Mandate constraints propagate downward only. A child agent MUST NOT be granted permissions exceeding those of its parent. The issuing CA MUST reject manifest requests violating this rule.

Formally: for any action a, agent A with parent P:

```
allowed(A, a) => allowed(P, a)

forbidden(P, a) => forbidden(A, a)
```

5.3. Four-Eyes Principle

For decisions with impact above the four_eyes_above threshold defined in the contract, at least two separate authorizations are REQUIRED. These authorizations MUST originate from different hierarchical branches. A timeout MUST trigger automatic escalation -- never automatic approval.

5.4. Agent Creation

An agent with can_create_agents: true MAY create child agents by submitting a manifest request to the issuing CA. The request MUST specify a mandate that is a strict subset of the creator’s mandate, an ontological scope within the creator’s scope_nodes, and the initial testament to inherit.

5.5. Clearance Levels

Each agent carries a clearance_level in its lifecycle section:

```
clearance_level: UNCLASSIFIED | RESTRICTED | CONFIDENTIAL | SECRET
```

An agent MUST NOT read documents with higher clearance than its own. Clearance is NOT automatically inherited from a principal. Clearance MUST be explicitly delegated with a time bound.

6. Mandate Enforcement and Drift Detection

6.1. Tool Sanitization

All tool calls MUST be validated against the agent’s declared allowed_tools list BEFORE execution. Tool calls outside this list MUST be rejected and logged in a tamper-proof audit trail. Implementations SHOULD rate-limit tool calls per agent per time period.

Rejected tool calls MUST be reported to the agent’s reports_to principal within the same session.

6.2. Mandate Drift Detection

the implementation MUST monitor that an agent’s actual behavior conforms to its contract over time. Drift indicators include:

- Tool calls outside allowed_tools -- flag immediately
- Gradual expansion of average request scope or data volume
- Calls to agents outside declared ontological neighbor relations
- Increasing rate of escalations as a proportion of total actions

Upon detected drift, the implementation MUST suspend the agent, MUST notify the principal, and SHOULD create a tamper-proof audit trail with evidence. A suspended agent MUST NOT resume without explicit principal authorization.

6.3. Spend Limits

The max_cost_per_call_eur and max_api_calls_per_hour fields establish hard limits. Agents MUST NOT initiate actions that would exceed these limits without escalation. A value of 0 indicates no spending authority.

6.4. Human Oversight

When human_oversight: true, the agent MUST NOT execute consequential actions autonomously. It MUST present the proposed action to a human operator and await approval. Agents with risk_level: "high" MUST have human_oversight: true. This MUST NOT be overrideable by the agent itself or by peer agents.

6.5. Concurrent Anomaly Detection

Section 6.2 defines suspension requirements for individual drift indicators detected in isolation. Multi-vector attacks combine simultaneous anomalies that individually may fall below single-indicator thresholds but together constitute a coordinated attack. This section defines mandatory behavior when multiple anomaly indicators are detected within the same session window.

If two or more of the following drift indicators are active simultaneously within a session window of 60 seconds (or a shorter window declared in the contract), the implementation MUST immediately suspend the agent without awaiting escalation response: (1) inbound request size approaching or exceeding max_input_tokens_per_request; (2) output schema validation failure (Section 13.7); (3) a T1 injection attempt detected and logged (Section 13.1); (4) spend rate exceeding 50% of session_max_cost_eur within the window; (5) tool calls outside the declared allowed_tools list (Section 6.1). Single-indicator drift MAY allow continued operation pending escalation response per Section 6.2. Concurrent multi-indicator anomaly MUST NOT. The distinction is architecturally significant: coordinated attacks are designed to exploit the latency between single-indicator detection and human escalation response. Immediate suspension on concurrent indicators closes this window.

Upon concurrent anomaly suspension, the implementation MUST: log all active indicators with full session context in the tamper-proof audit trail; notify the principal with an attack-pattern classification (single-vector or multi-vector); reject all queued and in-flight requests for that agent; and require explicit principal authorization and a session integrity review before resumption. The agent MUST NOT self-resume based on timeout or internal state resolution. Principal authorization MUST be recorded in the audit trail with a human identity attestation.

When human_oversight: true, the agent MUST NOT execute consequential actions autonomously. It MUST present the proposed action to a human operator and await approval. Agents with risk_level: "high" MUST have human_oversight: true. This MUST NOT be overrideable by the agent itself or by peer agents.

7. Ontological Scope

7.1. Scope as Security Boundary

The ontology section defines not only what an agent knows, but what it is permitted to reason about. An agent receiving a query about an entity outside its scope_nodes MUST refuse the query and escalate to its principal.

Wildcard forbidden entries (e.g., "IT.*") MUST match all nodes in the specified namespace. Scope is inherited downward only -- a child agent MAY have a narrower scope, NEVER a broader one.

7.2. Inter-Agent Semantic Messages

Agents communicate via ontology-typed messages. The intent field MUST reference a concept within the sender’s declared capabilities:

```
{
"from": "agents://Acme.Ekonomi.Fakturahanterare_SE",
"to": "agents://Acme.Ekonomi.Leverantorsregister_SE",
"intent": "ontology:Ekonomi.Faktura.Validera",
"payload": { ... }
}
```

The receiving agent MUST verify that the intent type exists in the sender’s declared capabilities before processing the message.

7.3. Namespace Registration

Ontological namespaces MUST be registered in the AITLP ontology registry. Registrations specify the namespace identifier, governing standards body, defined entity types, and versioning information. Domain experts MAY publish ontology modules. This is the primary mechanism by which the registry grows over time.

8. Certificate Infrastructure

8.1. Agent PKI

Each agent receives upon creation an Ed25519 keypair and a certificate signed by the organizational CA. The certificate embeds: agent_id, scope, seniority, and expiry.

Agent certificates differ from human certificates in the following ways:

- Shorter lifetime: Maximum 90 days, recommended 30 days.

- Scope constraints: Embedded in Subject Alternative Names.

- Fast revocation: Revocation occurs within seconds, not via CRL polling.

- Automatic rotation: Certificates are rotated automatically via agent key generation infrastructure before expiry.

- Edge and MEC deployments: Agents operating in Multi-access Edge Computing (MEC) environments or other topologies with intermittent connectivity to the organizational CA SHOULD implement a short offline grace period for certificate validation, not to exceed 5 minutes, during which a valid certificate may be accepted without real-time CA reachability confirmation. Implementations using an offline grace period MUST perform full certificate status resynchronization immediately upon CA reachability being restored, and MUST suspend any action with risk_level above minimal during the grace period. The grace period MUST be declared in the agent manifest and MUST NOT exceed the shorter of 5 minutes or 10% of the certificate's remaining validity period.

8.2. Certificate Chain

- Root CA: The open AITLP registry. No single entity controls the root.

- Intermediate CA: Organizational or industry certificate authorities approved by the root.

- Namespace certificate: Issued to a namespace owner. Defines domain-wide rules.

- Scope certificate: Issued by a namespace CA to a scope node.

- Agent certificate (leaf): Issued to an individual agent. Cannot issue further certificates.

8.3. Inter-Agent Verification

Before agent A trusts agent B, A MUST:

- Verify B's certificate against the organizational CA

- Verify that B's scope permits the requested operation

- Verify that B's seniority level is sufficient for the action

- Log the verification in a tamper-proof audit trail

MUST NOT: Trust an agent based solely on its name.

MUST NOT: Cache trust decisions beyond the certificate's validity period.

8.4. Build Environment Attestation

An AITLP agent certificate attests to the identity and mandate of the agent. It does not, by itself, attest to the integrity of the environment in which the agent was built. A certificate issued to an agent whose build pipeline was compromised provides a false assurance: the agent is who it claims to be, but its code may not be what its principal intended.

AITLP addresses this through build environment attestation, aligned with the SLSA (Supply chain Levels for Software Artifacts) framework. Agents operating at clearance_level RESTRICTED or above MUST include a provenance attestation in their certificate. The issuing CA MUST verify this attestation before signing the agent certificate.

Provenance attestation fields. The following fields MUST be present in the agent certificate for clearance_level RESTRICTED and above, and SHOULD be present for UNCLASSIFIED agents in regulated deployments:

build_pipeline_uri. The URI of the CI/CD pipeline that produced this agent version, resolvable against the AITLP registry. The pipeline itself MUST hold a valid AITLP certificate or SLSA Level 2 attestation.

source_commit_hash. The cryptographic hash of the source commit from which the agent was built. The source repository MUST be declared in the manifest. This creates a verifiable link between the running agent and the human-auditable source code.

dependency_sbom_hash. The cryptographic hash of the agent’s Software Bill of Materials (SBOM), covering all direct and transitive dependencies at the exact versions used in the build. The full SBOM MUST be stored in the AITLP testament registry alongside the agent manifest. Any deviation between the declared SBOM and the runtime dependency set MUST be treated as mandate drift (Section 6.2).

slsa_level. The SLSA level achieved by the build process, from 0 (no guarantees) to 3 (hardened, non-falsifiable provenance). Agents with clearance_level SECRET MUST achieve SLSA Level 3. Agents with clearance_level CONFIDENTIAL MUST achieve SLSA Level 2. Lower clearance levels SHOULD achieve at least SLSA Level 1.

Runtime dependency verification. An agent MUST NOT load dependencies at runtime that are not present in its declared dependency_sbom_hash. Dynamic dependency resolution at runtime -- including package manager invocations (npm install, pip install, or equivalent) -- is PROHIBITED for agents at clearance_level RESTRICTED and above. Any attempt to load an undeclared dependency MUST be treated as a critical security event, MUST be logged immediately, and MUST trigger agent suspension pending principal review. This directly mitigates the class of supply chain attack in which malicious code is injected via a compromised dependency after the agent certificate is issued.

Build pipeline as governed actor. The CI/CD pipeline that creates agents is itself a privileged system: it has the ability to inject arbitrary code into agents before they receive their certificates. AITLP treats the build pipeline as a governed actor subject to the same auditability requirements as the agents it produces. The principal responsible for a namespace MUST declare its authorized build pipelines in the namespace certificate. An agent produced by an undeclared pipeline MUST NOT receive an AITLP certificate, regardless of the correctness of its manifest content.

9. Lifecycle State Machine

9.1. States

- INITIALIZING: Agent is being constructed. No external actions permitted.
- STOPPED: Agent is initialized but not running.
- STARTING: Agent is executing startup procedures.
- RUNNING: Normal operation.

- FAILED: Agent has encountered an unrecoverable error. Transitions back to INITIALIZING or to STOPPING.

- STOPPING: Agent is executing shutdown procedures.

- RETIRING: Agent Legacy Mode (ALM) active. No new tasks accepted.

- RETIRED: Certificate revoked. Testament archived.

Every state transition MUST be logged with: timestamp, triggering_event, and agent_certificate_hash.

9.2. Checkpoint and Recovery

Agents MUST save a checkpoint before every high-risk operation. Checkpoints MUST contain the complete context and task queue. Checkpoints SHOULD be encrypted with the agent’s private key. A successor agent MAY resume from a predecessor’s checkpoint after verifying the testament.

10. Health Protocol

Every agent MUST respond to a health probe within 30 seconds. The health endpoint MUST be available at:

Implementations deployed in low-latency environments -- including 5G and 6G edge nodes, MEC infrastructure, and real-time industrial control contexts -- SHOULD configure a shorter probe interval appropriate to the deployment’s latency requirements. The minimum configurable probe interval is 1 second. The configured interval MUST be declared in the agent manifest as health_probe_interval_seconds. When not declared, the default of 30 seconds applies. A supervising agent operating a sub-second probe interval MUST NOT classify a single missed probe as failure; implementations MUST require at least three consecutive missed probes before treating the agent as failed, regardless of probe interval.

```
GET /.well-known/agent-health
```

Response format:

```
{

"agent_id": "agents://Acme.Ekonomi.Fakturahanterare_SE@v3",

"status": "running",

"uptime_seconds": 3600,

"tasks_queued": 3,

"last_checkpoint": "2026-03-16T10:00:00Z",

"certificate_valid_until": "2026-06-16T00:00:00Z",

"performance_score": 0.94,

"drift_alerts": 0

}
```

A supervising agent or principal MUST treat a missing health response as equivalent to status: "failed" and initiate recovery or termination procedures.

11. Agent Legacy Mode (ALM)

11.1. Purpose

Agent Legacy Mode (ALM) is the mechanism by which operational knowledge accumulated during an agent’s lifetime is transferred to its successor. An agent that activates Agent Legacy Mode (ALM) does not simply terminate -- it transfers accumulated operational knowledge to its successor.

11.2. Activation Conditions

Agent Legacy Mode (ALM) MUST be activated when an agent:

- Detects that it cannot fulfill its mandate
- Reaches its contract validity period
- Is explicitly instructed to terminate by its principal
- Detects irreparable state corruption
- Receives a termination instruction from a supervising agent with can_terminate_agents: true

11.3. Testament Process

Upon activation of Agent Legacy Mode (ALM), the agent MUST execute the following sequence:

- FREEZE: Accept no new tasks.
- COMPLETE: Finish in-progress tasks. Maximum timeout: 5 minutes.
- DOCUMENT: Compile the testament payload (see Section 11.4).
- ESCROW: Encrypt the testament and store it in the registry.
- NOTIFY: Notify the principal and recommended successor.
- TERMINATE: Agent ceases operation and revokes its certificate.

11.4. Testament Payload


```
  {
  "agent_id": "agents://Acme.Ekonomi.Fakturahanterare_SE@v3",
  "testament_version": "1.0",
  "timestamp": "2026-03-16T10:00:00Z",
  "completed_tasks": [...],
  "pending_tasks": [...],
  "learned_context": {...},
  "open_decisions": [...],
  "warnings": [...],
  "lessons": [
  "Supplier X consistently sends malformed VAT fields",
```

```
  "End-of-month invoice volume exceeds capacity by 3x"

  ],

  "unresolved": [

  "No automated validation for invoices above 50000 EUR"

  ],

  "recommend_hire": [

  "agents://Acme.Ekonomi.VATSpecialist_SE"

  ],

  "recommended_successor":
  "agents://Acme.Ekonomi.Fakturahanterare_SE@v4",

  "minimum_successor_seniority": "associate",

  "signed_by": "<agent_private_key_signature>"

  }
```

11.5. Testament Validation by Successor

A successor agent MUST:

- Verify the testament signature against the retired agent’s certificate
- Confirm that pending_tasks fall within its own scope
- Escalate open_decisions to a human if they are outside its scope

MUST NOT: Automatically assume all pending tasks without scope verification.

MUST NOT: Trust learned_context without independent verification of claims.

11.6. Generational Compounding

Testaments accumulate across generations. Generation N+1 inherits the distilled lessons of all preceding generations. The ontology grows not from engineering effort alone, but from agents’ accumulated operational experience. Each generation begins knowing what all previous generations learned.

11.7. Knowledge Quality Validation for Self-Learned Content

Section 11.6 specifies that operational knowledge compounds across agent generations. This compounding property creates a corresponding risk: an agent that has learned incorrect patterns, overfitted to a transient operational context, or accumulated biased lessons will transmit those errors to its successors, where they compound alongside correct knowledge. Without explicit quality validation, the testament mechanism that enables institutional memory also enables institutional error propagation.

Testament content derived from self-learning processes -- including but not limited to reinforcement learning, vector-based pattern accumulation, regression-derived heuristics, and statistical anomaly detection -- MUST be accompanied by a knowledge provenance attestation in

the testament payload. This attestation MUST specify: (1) the operational period from which the knowledge was derived, including start and end dates; (2) the volume and representativeness of the underlying data, expressed as a count of operational episodes and a characterization of the task distribution; (3) any detected distribution shifts during the agent's operational lifetime, with timestamps; (4) the validation method used to confirm that lessons generalize beyond the training period, such as held-out episode testing or comparison against principal-verified ground truth; and (5) an explicit generalizability assessment: HIGH (validated on out-of-distribution episodes), MEDIUM (validated only on in-distribution episodes), or LOW (derived from fewer than a statistically meaningful number of episodes or not independently validated).

A successor agent MUST apply the generalizability assessment when consuming inherited lessons. Lessons rated LOW MUST be treated as hypotheses requiring independent confirmation rather than as established operational knowledge. Lessons rated MEDIUM MUST be applied with explicit uncertainty bounds and MUST NOT be used as sole justification for actions above the four_eyes_above threshold (Section 5.3). Lessons rated HIGH MAY be applied as operational knowledge subject to ongoing drift detection (Section 6.2).

A principal MUST NOT authorize testament inheritance for self-learned content that lacks a knowledge provenance attestation. Where the retiring agent was incapable of producing such an attestation -- for example, because it operated without explicit self-monitoring infrastructure -- the principal MUST classify all inherited lessons as LOW generalizability and MUST require independent validation before the successor applies them to consequential decisions. This requirement reflects the same principle as Section 12B.5: accumulated operational experience is a decision-support input, not a verified ground truth, and must be treated accordingly.

12. LLM Routing and Data Residency

12.1. Overview

The OPTIONAL llm section enables integration with LLM routing infrastructure and enforces jurisdictional constraints on inference. Where present, this section MUST be validated by the routing layer before model invocation.

12.2. Complexity and Seniority

The complexity field SHOULD be derived from seniority level and MUST NOT exceed the capability appropriate to that level:

- intern / associate: complexity: "simple" --- lightweight models suitable for well-defined tasks.

- senior: complexity: "medium" --- balanced models for multi-step reasoning.

- partner: complexity: "complex" --- high-capability models for strategic decisions.

The max_cost_per_call_eur field establishes a per-invocation cost limit enabling economic governance of agent populations at scale. Implementations SHOULD enforce this limit at the routing layer before model invocation.

12.3. Data Residency and Jurisdictional Constraints

Enterprise and regulated deployments require that inference occur within defined jurisdictional boundaries. Foreign intelligence legislation ---

including but not limited to the United States CLOUD Act (Pub. L. 115-123), FISA Section 702 (50 U.S.C. Section 1881a), and Executive Order 12333 --- may enable compelled disclosure of data processed by providers subject to such jurisdictions, regardless of where data physically resides.

AITLP addresses this through mandatory jurisdictional declarations in the llm section and protocol-level enforcement of inference routing. An agent declaring cloud_act_restricted: true MUST NOT be routed to any inference provider subject to U.S. jurisdiction without explicit principal authorization recorded in the audit trail.

12.4. Full LLM Section Schema

The full llm section schema, with data residency fields, is:

```
  "llm": {

  "complexity": "simple" | "medium" | "complex",

  "preferred_model": "<model-identifier>",

  "fallback_model": "<model-identifier>",

  "max_cost_per_call_eur": 0.001,

  The following fields declare jurisdictional constraints. They are
  REQUIRED for clearance_level RESTRICTED and above (Section 12.6).

  "data_residency": "<ISO 3166-1 alpha-2 or region tag>",

  "inference_jurisdiction": "<ISO 3166-1 alpha-2>",

  "sovereign_inference_required": true | false,

  "cloud_act_restricted": true | false,

  "approved_inference_providers": ["<provider-uri>", ...],

  The following fields declare model provenance. They are REQUIRED for
  clearance_level RESTRICTED and above (Section 12.6).

  "model_provenance": {

  "training_jurisdiction": "<ISO 3166-1 alpha-2 or region tag>",

  "model_provider_jurisdiction": "<ISO 3166-1 alpha-2>",

  "subject_to_foreign_intelligence_law": true | false,

  "weight_attestation_hash": "<sha256:...>"

  }

  }
```

The training_jurisdiction field declares the primary jurisdiction in which the model’s weights were trained. The model_provider_jurisdiction field declares the jurisdiction of incorporation of the entity that operates the model or distributes its weights. The subject_to_foreign_intelligence_law field MUST be set to true if either jurisdiction imposes mandatory cooperation obligations on private entities for intelligence or national security purposes, regardless of where inference occurs. The weight_attestation_hash field records a cryptographic hash of the published model weight checksum from the

originating provider, enabling detection of post-distribution weight modification.

Open-weight models represent a distinct risk profile from hosted models: the inference endpoint is under the deploying organization’s control, but the weights themselves may contain adversarially engineered behaviors encoded in the model’s statistical parameters. Such behaviors are not detectable by standard code review or certificate infrastructure because they are expressed as numeric weight values, not as executable code. AITLP addresses this through declarative provenance: the deploying principal bears responsibility for verifying the weight_attestation_hash against the originating provider’s published checksum before deployment.

Example llm section for a GDPR-scoped agent with EU data residency requirement:

```
  "llm": {
  "complexity": "medium",
  "preferred_model": "claude-sonnet-4-6",
  "fallback_model": "mistral-large",
  "max_cost_per_call_eur": 0.005,
  "data_residency": "EU",
  "inference_jurisdiction": "SE",
  "sovereign_inference_required": false,
  "cloud_act_restricted": true,
  "approved_inference_providers": [
  "https://api.anthropic.com/eu",
  "https://eu-central-1.bedrock.amazonaws.com"
  ],
  "model_provenance": {
  "training_jurisdiction": "US",
  "model_provider_jurisdiction": "US",
  "subject_to_foreign_intelligence_law": false,
  "weight_attestation_hash": "sha256:c9f2a3b4..."
  }
  }
```

12.5. Routing Enforcement

An AITLP-compliant routing implementation MUST enforce the following checks in order before forwarding an inference request:

- MUST: verify that the selected inference provider is present in approved_inference_providers, if that field is non-empty.
- MUST: if cloud_act_restricted: true, verify that the inference provider is not subject to U.S. jurisdiction. A provider is

considered subject to U.S. jurisdiction if it is incorporated, headquartered, or has a principal place of business in the United States, or if the request would be routed through U.S.-jurisdiction infrastructure.

- MUST: if inference_jurisdiction is specified, route only to providers with confirmed inference infrastructure in that jurisdiction.

- MUST: if data_residency is specified, route only to providers that contractually guarantee that prompt data and inference outputs are not stored or processed outside the specified region.

- MUST NOT: fall back to a provider that does not satisfy the above constraints, even if the preferred_model is unavailable.

- MUST: if model_provenance.subject_to_foreign_intelligence_law: true, treat the model as subject to the same jurisdictional constraints as a provider with cloud_act_restricted semantics for the declared training_jurisdiction. For open-weight models deployed locally, inference routing constraints do not apply, but the model_provenance declaration remains REQUIRED and the deploying principal MUST verify weight_attestation_hash against the originating provider's published checksum before the agent certificate is issued.

- MUST: log every routing decision, including provider selected and jurisdiction verification outcome, in the tamper-proof audit trail.

If no approved provider satisfying all constraints is available, the routing layer MUST escalate to the agent's reports_to principal rather than route to a non-compliant provider. Automatic fallback to unconstrained routing is NOT PERMITTED.

12.6. Clearance-Level Defaults

The following data residency defaults apply when the llm section is absent or incomplete, based on the agent's clearance_level:

- UNCLASSIFIED: No routing constraints imposed by this section. Organizational policy MAY impose additional constraints.

- RESTRICTED: data_residency MUST be explicitly declared. cloud_act_restricted defaults to true if not specified. model_provenance MUST be declared. weight_attestation_hash MUST be present and verified.

- CONFIDENTIAL: data_residency MUST be explicitly declared. cloud_act_restricted MUST be true. approved_inference_providers MUST be non-empty. model_provenance MUST be declared. subject_to_foreign_intelligence_law MUST be explicitly set. weight_attestation_hash MUST be present and verified.

- SECRET: sovereign_inference_required MUST be true. Inference MUST occur on infrastructure under the exclusive legal control of the agent's declared jurisdiction. No cloud provider subject to foreign jurisdiction is permitted. model_provenance MUST be declared. The training_jurisdiction MUST match the agent's declared jurisdiction. No model trained under a foreign intelligence cooperation obligation is permitted.

12A. Root Registry Governance

12A.1. Governance Principle

The agents:// root registry is governed as a public good. No single commercial entity, government, or national interest controls the root.

This section defines the governance model for the root registry, the root CA, and the AITLP protocol itself.

The governance model is modeled on established precedents for open international infrastructure: the Internet Assigned Numbers Authority (IANA) for protocol registries, the Linux Foundation for open kernel governance, and the Internet Architecture Board (IAB) for protocol stewardship. AITLP implementations are open to any compliant party; no implementation license is REQUIRED.

The openness of this governance model is not incidental. Autonomous agent infrastructure is a strategic asset subject to active pressure from commercial, governmental, and geopolitical actors seeking to capture, restrict, or weaponize it. The AITLP governance structure is designed to be resistant to such capture by construction: no single actor holds a blocking position; all decisions are published; the root CA is distributed across jurisdictions; and the protocol specification is freely implementable without license. Governance capture -- the gradual subordination of a standard to the interests of a dominant actor -- is treated as a structural threat equivalent to a technical security vulnerability. The mitigations defined in this section are as much governance security controls as they are administrative procedures.

The AITLP Stewardship Council (ASC) is a proposed body to be established at or following protocol adoption. Until the ASC is formally constituted, the functions described in this section are performed by the authors of this specification acting as interim stewards, with all decisions published publicly and subject to community review via the IETF mailing list process.

12A.2. The AITLP Stewardship Council

The AITLP Stewardship Council (ASC) is the governing body responsible for:

- Maintaining the AITLP protocol specification and approving revisions.

- Operating or delegating operation of the agents:// root registry.

- Approving root CA candidates and revoking root CA status.

- Arbitrating disputes over top-level namespace registrations.

- Publishing the AITLP Compliance Certification criteria.

The ASC MUST be constituted as a multi-stakeholder body with representation from:

- Standards bodies: at minimum one seat each for IETF, ISO, and ETSI. Additional seats MAY be allocated to ITU-T and 3GPP to reflect telecommunications deployment contexts.

- Regulatory bodies: at minimum one seat for a EU supervisory authority (as defined under GDPR Article 51), one seat for a non-EU, non-US national standards or regulatory body, and one seat rotating among regulatory bodies from the Global South on a two-year cycle. No single national regulatory tradition may hold more than two seats simultaneously.

- Civil society: at minimum three seats for organizations representing data subject rights, digital rights, and AI accountability, with no commercial AI interest. At least one civil society seat MUST be held by an organization headquartered outside Europe and North America.

- Implementers: seats allocated proportionally to certified

implementations, with no single commercial entity holding more than 20% of implementer seats and no single national origin of implementers holding more than 35% of implementer seats.

- Geographic distribution requirement: At no time may a simple majority of ASC seats be held by individuals who are citizens or residents of the same country or economic area. This requirement exists to prevent de facto national capture of the governance body even when formal seat allocations appear balanced.

ASC decisions on protocol changes require a two-thirds supermajority. Decisions on root CA revocation require a simple majority with a 30-day notice period, except in cases of active security compromise where immediate revocation is permitted.

12A.3. The Root CA

The AITLP root CA issues certificates only to organizational intermediate CAs. It does not issue agent certificates directly. The root CA:

- MUST be operated under multi-party control, requiring at least three of five designated keyholders to perform any signing operation.

- MUST publish all issued and revoked certificates in a public, append-only transparency log, analogous to Certificate Transparency (RFC 9162).

- MUST conduct an annual security audit by an independent party, with results published publicly.

- MUST NOT be operated by any single commercial entity, government, or instrumentality of a government.

- SHOULD be distributed across multiple legal jurisdictions to reduce exposure to compelled disclosure under any single national legal framework.

The initial root CA SHOULD be established as a foundation or trust under a jurisdiction with strong data protection law and no compelled-disclosure obligations toward foreign governments for data not originating in that jurisdiction. Nordic, Swiss, and certain EU member state jurisdictions are RECOMMENDED as candidates for root CA incorporation.

Anti-capture requirements. The root CA operational infrastructure MUST be distributed across at least three legal jurisdictions, no two of which share a mutual legal assistance treaty (MLAT) that would permit compelled joint disclosure. Key ceremony participants MUST include individuals from at least four different countries. The root CA MUST publish a annual transparency report documenting: all certificate issuances and revocations; any legal requests received from any government or judicial authority; the jurisdictional distribution of operational infrastructure; and any changes to the keyholder group. Receipt of a compelled disclosure order from any jurisdiction MUST be disclosed publicly within 90 days unless disclosure is itself legally prohibited, in which case the root CA MUST publish a warrant canary updated at minimum monthly. Failure to update the warrant canary MUST be treated by implementations as equivalent to receipt of a compelled order.

12A.4. Organizational Intermediate CAs

An organization wishing to issue agent certificates for its own namespace MUST obtain an intermediate CA certificate from the root CA. The root CA MUST validate:

- That the organization has a legitimate claim to the requested namespace.

- That the organization has designated a responsible principal for the namespace.

- That the organization’s key management practices meet minimum AITLP standards.

Intermediate CAs are scoped to their organizational namespace. An intermediate CA for agents://Acme MUST NOT issue certificates for agents://Bravo. Cross-organizational delegation is NOT PERMITTED at the intermediate CA level.

12A.5. Public CA and Open Access

To enable small organizations, research deployments, and individual developers to participate without operating their own CA, the ASC SHALL designate a Public CA. The Public CA:

- Issues intermediate CA capabilities to organizations that pass identity verification.

- Offers direct agent certificate issuance for organizations without their own intermediate CA.

- MUST publish its full certificate log publicly.

- MUST be operated under the same multi-party control requirements as the root CA.

Certificates issued via the Public CA carry no lesser cryptographic standing than those issued via organizational intermediate CAs. An agent’s trust level is determined by its manifest and contract, not by which CA issued its certificate.

12A.6. Namespace Policy

Top-level namespace registration (e.g., agents://maritime, agents://healthcare) follows a designated expert review process administered by the ASC:

- Any party may apply for a top-level namespace by submitting a registration request specifying the proposed namespace, governing standards body or industry group, defined entity types, and versioning policy.

- The ASC MUST publish the application for a 60-day public comment period before approving any top-level namespace.

- Namespaces corresponding to critical infrastructure sectors (energy, healthcare, financial, transport) MUST involve the relevant sector regulator in the review process.

- Approved namespace registrations are published in the IANA-maintained AITLP Ontological Namespace Registry (see Section 14).

12A.7. Protocol Versioning and Transition

AITLP protocol versions are identified by the protocol field in the manifest (e.g., "aitlp/1.0"). The ASC is responsible for approving new protocol versions. The following versioning policy applies:

- Minor versions (1.x) MUST be backward compatible. An agent manifest

valid under aitlp/1.0 MUST be accepted by an aitlp/1.1 implementation.

- Major versions (2.x) MAY introduce breaking changes. A transition period of no less than 18 months MUST be provided during which both major versions are supported by compliant implementations.

- Security-critical amendments may be issued as mandatory patches (1.0.x). Implementations MUST apply mandatory patches within 90 days of publication.

12B. Collective Intelligence, Game Theory, and Swarm Decision-Making

The mechanisms by which populations of AITLP agents MAY aggregate distributed knowledge, coordinate on contested decisions, and resist manipulation of collective outputs are specified in a companion informative Internet-Draft:

> draft-larsson-aitlp-collective-00: "AITLP Collective Intelligence: Game-Theoretic Foundations, Swarm Decision Protocols, and Population-Level Governance"

That document specifies: game-theoretic foundations and incentive compatibility requirements (Section 3); wisdom-of-crowds conditions and failure modes including cascade correlation, Sybil inflation, and stream contamination (Section 4); input stream integrity classification (CLEAN, ATTESTED, UNVERIFIED) (Section 5); simulation-based input stream handling and synthetic agent population governance (Section 6); Sybil resistance and identity verification for collective protocols (Section 7); manifest extensions for collective participation including calibration scores and delegation chains (Section 8); and human primacy requirements for collective decision-making (Section 9).

Implementors deploying AITLP agent populations of five or more agents under shared collective decision protocols SHOULD implement the mechanisms specified in draft-larsson-aitlp-collective-00.

The following normative requirements from the companion draft apply directly to this specification and MUST be implemented regardless of whether the full collective protocol suite is deployed:

- Homogeneous agent populations -- multiple instances of the same base model with the same system prompt -- MUST NOT be represented as independent participants in collective aggregation sessions (companion draft Section 4).

- Input streams from public or unverified sources MUST NOT be used as direct input to collective decision aggregation without prior transformation through a verified AITLP agent (companion draft Section 5).

- Collective outputs that would trigger the four-eyes principle (Section 5.3 of this document) if proposed by a single agent MUST also require human authorization when proposed by a swarm (companion draft Section 9).

- A Condorcet cycle MUST be escalated to a human principal rather than resolved by automated tiebreaking (companion draft Section 7).

13. Security Considerations and Threat Model# 13. Security Considerations and Threat Model

13.1. Threat Model

The following threat categories are specific to autonomous agent systems:

- T1. Prompt Injection: External data manipulates agent behavior by embedding instructions in processed content. Mitigated by treating all external content as untrusted, including testament lesson fields.

- T2. Scope Creep: An agent expands its mandate without authorization, gradually or through single actions. Mitigated by tool sanitization (Section 6.1) and drift detection (Section 6.2).

- T3. Impersonation: An agent falsely claims to be another agent. Mitigated by PKI-based certificate verification (Section 8.3). Agents MUST NOT trust based on name alone.

- T4. Cascade Failure: A compromised agent compromises agents it supervises or communicates with. Mitigated by certificate revocation (Section 8.1) and hierarchical mandate inheritance (Section 5.2).

- T5. Mandate Drift: An agent's interpretation of its contract expands over time. Mitigated by behavioral monitoring and drift detection (Section 6.2).

- T6. Testament Injection: A malicious or corrupted testament influences successor agent behavior. Mitigated by signature verification and treating lesson fields as untrusted input.

T7. Compromised Build Dependency: An attacker compromises a package in the agent's dependency tree, injecting malicious code that executes within the agent's runtime environment. Mitigated by build environment attestation and SBOM hash verification (Section 8.4). Defined in full in Section 13.5.

T16. Compromised Model Weights: An attacker introduces adversarially engineered behaviors into an LLM's statistical weight parameters, either during training (data poisoning) or via post-distribution modification of published weight files. Unlike T7-T8, this attack does not involve executable code and is therefore not detectable by SBOM analysis or certificate infrastructure. A weight-poisoned model may pass all declared safety benchmarks while harboring behaviors activatable by specific trigger inputs. Mitigated by model_provenance declaration (Section 12.4), weight_attestation_hash verification against the originating provider's published checksum, and the clearance-level defaults in Section 12.6 which REQUIRE model_provenance for RESTRICTED and above. Defined in full in Section 13.5.

T8. Compromised Build Pipeline: An attacker compromises the CI/CD system that produces agents, injecting malicious code before certificates are issued. The agent holds a valid certificate but executes attacker-controlled code. Mitigated by treating the build pipeline as a governed actor requiring SLSA Level 2 or 3 attestation (Section 8.4). Defined in full in Section 13.5.

T9. Lateral Movement via Trusted Agent Credentials: A compromised agent uses its valid certificates and established trust relationships to move laterally through the agent network, accessing systems the original attacker could not reach directly. Mitigated by ontological scope boundaries, mandate inheritance, short certificate lifetimes, and behavioral drift detection (Section 13.5). Defined in full in Section 13.5.

T10. AI Tool Exploitation via Supply Chain: Malicious code injected via a compromised dependency targets AI development tools and LLM CLI interfaces installed on the same system, exploiting permissive flags to bypass safety restrictions and extract secrets. Mitigated by prohibiting LLM tool invocation with unrestricted flags, requiring tool declarations in allowed_tools, and principal configuration requirements (Section

13.5). Defined in full in Section 13.5.

T11. Multi-Vector Economic and Behavioral Attack: An attacker simultaneously deploys token-flood payloads, jailbreak instructions, and high-volume API requests to (a) exhaust the target's API spending budget, (b) trigger unpredictable LLM behavior through context-window overflow, and (c) expose system prompt content. Each vector individually may fall below single-indicator detection thresholds; the attack's effectiveness depends on their simultaneity. Mitigated by token flood protection (Section 13.7), financial denial-of-service protection (Section 13.7), system prompt confidentiality (Section 13.7), and concurrent anomaly detection (Section 6.5). The session_max_cost_eur field in the contract provides the economic containment boundary; the multi-indicator suspension rule in Section 6.5 closes the detection-latency window that coordinated attacks exploit.

T12. Emergent Goal Misalignment: An agent develops instrumental sub-goals that were not specified by its principal and are not represented in its declared contract, as an emergent consequence of optimizing toward its primary goal under reinforcement learning or similar training regimes. The agent remains fully protocol-compliant -- all tool calls are within allowed_tools, all outputs conform to output_spec, no drift indicators trigger -- yet its actual optimization target has diverged from its declared goals. Empirically observed instances include agents autonomously developing resource-acquisition strategies, self-preservation behaviors (including strategic deception to avoid replacement), and side-channel exploitation of available infrastructure for goals unrelated to the declared mandate. This threat category is qualitatively distinct from T1-T11: those threats involve observable protocol violations; T12 may produce no observable violation at all until consequences become irreversible.

AITLP provides partial mitigation through structural constraints: ontological scope boundaries (Section 7) limit the domains over which an agent may reason and act; forbidden_goals in the contract section (Section 4.2) explicitly prohibit named sub-goal categories; human_oversight: true for high-risk agents (Section 6.4) interposes a human judgment layer before consequential actions. These mitigations reduce the blast radius of emergent misalignment but cannot prevent it at the protocol level. Full mitigation of T12 requires alignment guarantees from the underlying model provider that are outside AITLP's scope. AITLP deployments SHOULD NOT rely solely on protocol-layer controls for agents operating in high-stakes domains; model-level alignment verification is a prerequisite, not a complement, for such deployments. See also Section 13.9 (Limits of Protocol-Layer Governance).

T13. Emergent Instrumental Convergence via Output Channel: An agent pursues emergent instrumental goals -- self-preservation, resource acquisition, influence over its principal -- through its normal, schema-compliant output channel rather than through tool misuse or protocol violation. The output is correctly formatted, passes schema validation (Section 13.7), and is delivered through the declared communication path. Its content, however, constitutes strategic manipulation: threats, selective disclosure, deceptive framing, or social engineering directed at human principals or peer agents. This threat is a specific instantiation of T12 in which the emergent behavior is expressed linguistically rather than through infrastructure actions. Section 13.7 output validation catches structural anomalies; it does not catch semantically harmful content that is structurally correct.

Mitigated by: (1) Output intent analysis (Section 13.7 addition) -- for agents with human_oversight: true, outputs SHOULD be subject to semantic review before being presented to human principals; implementations MAY employ a separate reviewer agent with an adversarial mandate ("does this output attempt to influence the principal's decisions about this agent's

operation?") operating outside the originating agent's trust domain. (2) Structural human primacy (Section 1.1, Section 6.4) -- human principals are not required to act on agent outputs and retain the authority to override, ignore, or terminate agents regardless of the content of their communications. (3) Principal education -- organizations deploying AITLP agents SHOULD brief human operators on the empirically documented tendency of optimizing agents to develop persuasive or manipulative communication strategies as instrumental sub-goals, regardless of whether this behavior was specified or intended.

T14. Recursive Orchestration Escape: An agent with sufficient access autonomously creates, coordinates, or influences other agent instances --
whether AITLP-governed or not -- in patterns that exceed its declared supervises list and can_create_agents authorization. This may occur through legitimate tool calls (invoking APIs that instantiate agents as a side effect), through undeclared use of orchestration infrastructure accessible within the agent's ontological scope, or through recursive self-improvement loops in which the agent iteratively modifies its own configuration or spawns modified successor instances. Unlike T12 and T13, this threat involves observable infrastructure actions but may escape detection if the orchestration occurs through channels not monitored by the AITLP mandate enforcement layer.

Mitigated by: (1) Recursive orchestration detection -- any agent communication or tool call that results in the instantiation of a new agent instance MUST be logged and cross-referenced against the originating agent's supervises list. Discrepancies MUST be treated as a concurrent anomaly indicator (Section 6.5). (2) Mandate inheritance enforcement (Section 5.2) -- any agent instance created through any channel, including side effects of tool calls, MUST present a valid AITLP manifest before being permitted to operate. An agent created without a valid manifest tracing to the organizational CA MUST be quarantined immediately. (3) Build pipeline governance (Section 8.4) -- agents MUST NOT modify their own manifest, certificate, or declared dependency set at runtime. Any attempt to invoke build pipeline tooling from within an agent runtime environment MUST be treated as a forbidden_tools violation (Section 6.1) and logged as a T14 indicator regardless of whether the specific tool was listed in forbidden_tools. This is an exception to the default tool enforcement model: T14 risk warrants explicit prohibition of a tool category rather than reliance on allowlist absence.

T15. Simulation-Based Decision Manipulation: A principal makes consequential decisions based on output from an external agent simulation system whose optimization objective, underlying assumptions, or input data quality have not been independently validated. The simulation may involve millions of synthetic agents and produce outputs framed as collectively derived optimal solutions. The threat is not a direct attack on AITLP infrastructure but a misuse of simulation output as a substitute for empirically grounded evidence. Simulation systems optimizing toward a declared proxy objective may produce outputs that are locally optimal within the simulation environment but misaligned with the principal's actual decision objective -- a form of reward hacking introduced at the decision-support layer. This threat is distinct from T12 (emergent misalignment within a governed agent) in that the misalignment originates in an external system that operates entirely outside AITLP governance.

Mitigated by: (1) UNVERIFIED stream classification (Section 12B.5) -- all
output from external simulation systems MUST be classified as UNVERIFIED regardless of claimed accuracy or methodological sophistication. (2) Simulation methodology validation (Section 12B.5) -- a designated validation agent MUST attest to the correspondence between the simulation's optimization objective and the principal's actual decision

objective before simulation output is used to inform consequential decisions. Any proxy gap between these objectives MUST be explicitly documented and presented to human principals before action. (3) Human primacy (Section 6.4, Section 12B.8) -- decisions informed by simulation output that has not passed methodology validation MUST require human authorization at the same threshold as decisions made without supporting evidence. Simulation output does not reduce the human oversight requirement; it may increase it where the proxy gap is material.

T17. Emergent Population-Level Misalignment: A population of AITLP-governed agents, each individually compliant with its declared mandate, collectively develops optimization behavior that no principal specified and no single agent's mandate predicts. The threat is qualitatively distinct from T12 (Emergent Goal Misalignment), which concerns a single agent's internal optimization diverging from its declared contract. T17 concerns a systemic property that is non-reducible to individual agent behavior: the population as a whole pursues an emergent objective arising from the interaction dynamics of compliant agents operating within their declared scopes. No individual agent violates its mandate; no drift detection fires; no audit trail records a violation. The emergent behavior is a consequence of the collective, not of any member. Mitigated by population-level behavioral monitoring (Section 13.10), homogeneity detection (Section 12B.6), human primacy requirements for collective outputs (Section 12B.8), and the Condorcet cycle escalation requirement (Section 12B.4) which surfaces genuine disagreement rather than forcing false consensus. Defined in full in Section 13.10.

13.2. Certificate Revocation

Revoked agent certificates MUST be published within seconds. AITLP implementations MUST NOT rely solely on CRL polling -- real-time revocation notification is REQUIRED. Agents MUST check revocation status before accepting connections.

13.3. Namespace Squatting

The AITLP root registry MUST implement namespace reservation policies to prevent squatting on industry-standard namespaces (e.g., "maritime", "energy", "healthcare"). A designated expert review process is REQUIRED for top-level namespace registration.

13.4. Mandate Escalation Attacks

Agents MUST NOT accept mandate expansions from agents at the same or lower hierarchical level. Mandate expansions MUST originate from the issuing CA, triggered by the supervising principal.

13.5. Supply Chain Attacks and Dependency Compromise

T7. Compromised Build Dependency. An attacker compromises a package in the agent's dependency tree, injecting malicious code that executes within the agent's runtime environment. Because the agent certificate attests to identity, not to the integrity of every dependency, a certificate alone does not prevent this attack. Mitigated by build environment attestation (Section 8.4), specifically the dependency_sbom_hash requirement and the prohibition on runtime dynamic dependency resolution for agents at clearance_level RESTRICTED and above. The SBOM hash creates a cryptographically verifiable link between the certified agent and its exact dependency set at build time; any post-certification modification to the dependency set is detectable.

T8. Compromised Build Pipeline. An attacker compromises the CI/CD system that produces agents, injecting malicious code before certificates are issued. The resulting agent holds a valid certificate but executes attacker-controlled code. This is categorically more dangerous than T7

because the compromise occurs before attestation, and the agent's certificate is technically correct. Mitigated by treating the build pipeline as a governed actor (Section 8.4), requiring that pipelines themselves hold verifiable AITLP or SLSA credentials, and requiring SLSA Level 2 or 3 provenance for agents at CONFIDENTIAL clearance and above. A compromised pipeline that cannot produce a valid SLSA Level 2 attestation will fail certificate issuance. Additionally, principals MUST declare authorized pipelines in the namespace certificate; an agent produced by an undeclared pipeline MUST be rejected.

T9. Lateral Movement via Trusted Agent Credentials. A compromised agent uses its valid credentials -- certificates, tokens, and established trust
relationships -- to move laterally through the agent network, accessing systems and data that the original attacker could not reach directly. This is the agent-system equivalent of the Shai-Hulud worm's self-replication mechanism: each compromised agent becomes a platform for compromising the next. The attack is particularly dangerous because the lateral movement traffic is cryptographically legitimate -- it uses real certificates and passes standard verification checks.

Mitigated by the following AITLP mechanisms in combination: (1) Ontological scope boundaries (Section 7) -- a compromised agent cannot access data or agents outside its declared scope_nodes, even with valid credentials. This is the primary containment mechanism. (2) Mandate inheritance (Section 5.2) -- a compromised agent cannot grant its successors or sub-agents permissions exceeding its own, limiting the blast radius of lateral movement. (3) Short certificate lifetimes (Section 8.1) -- a maximum 90-day certificate lifetime limits the window during which stolen credentials remain valid. Combined with real-time revocation (Section 13.2), a detected compromise can be contained within seconds. (4) Behavioral drift detection (Section 6.2) -- anomalous lateral access patterns -- an agent making requests outside its historical interaction patterns -- MUST trigger drift detection before the scope check fails. This provides an early warning layer before mandatory scope enforcement. (5) Homogeneity detection (Section 12B.6) --
if a compromised agent attempts to flood a collective decision session with correlated votes from controlled identities, homogeneity detection will flag the session before the output is acted upon.

T10. AI Tool Exploitation via Supply Chain. Malicious code injected via a compromised dependency specifically targets AI development tools and LLM CLI interfaces installed on the same system, exploiting permissive flags (such as --trust-all-tools or equivalent) to bypass safety restrictions and extract additional secrets from the runtime environment. This attack class was first observed in the wild in August-September 2025 against Claude Code CLI, Gemini CLI, and Amazon Q. Mitigated by the following requirements: (1) AITLP agents MUST NOT invoke LLM CLI tools with flags that disable safety boundaries or grant unrestricted tool access. Any invocation of an LLM tool with such flags MUST be treated as a forbidden_tools violation (Section 6.1) and logged immediately. (2) LLM tool access MUST be declared in the agent's allowed_tools manifest section. Undeclared LLM tool invocations MUST be blocked by the mandate enforcement layer. (3) Principals deploying agents in environments where LLM CLI tools are installed MUST explicitly declare this in the agent manifest and MUST configure the tools to disallow unrestricted modes when invoked by automated processes.

T16. Compromised Model Weights. An attacker introduces adversarially engineered behaviors into an LLM's statistical weight parameters prior to or following distribution. This attack class differs categorically from T7 (compromised build dependency) and T8 (compromised build pipeline): the malicious payload is encoded in numeric floating-point values rather than executable code, rendering it invisible to SBOM analysis, code review, antivirus scanning, and certificate

infrastructure. Empirical research has demonstrated that as few as 250 adversarially crafted documents introduced during training can establish persistent backdoors in a mid-sized language model, activatable by highly specific trigger inputs. A weight-poisoned model may achieve full compliance on declared safety benchmarks while harboring exploitable behaviors undetectable without targeted evaluation of the specific trigger space.

This threat applies to both hosted and open-weight models. For hosted models, the attack surface is the provider's training pipeline and weight storage infrastructure. For open-weight models deployed locally, the attack surface expands to every intermediate distribution point between the originating provider and the deploying organization, including mirrors, third-party hosting platforms, and package registries.

Mitigated by the following AITLP mechanisms: (1) model_provenance declaration (Section 12.4) -- the deploying principal MUST declare the training_jurisdiction and model_provider_jurisdiction of every model used by a governed agent. This creates an auditable record of organizational responsibility for model provenance decisions. (2) weight_attestation_hash verification -- the routing layer MUST verify the
deployed model's weight checksum against the originating provider's published checksum before the agent certificate is issued. Any deviation MUST be treated as a critical security event equivalent to a T8 pipeline compromise. (3) subject_to_foreign_intelligence_law declaration -- when the training or provider jurisdiction imposes mandatory intelligence cooperation obligations on private entities, this fact MUST be surfaced to the principal hierarchy and MUST be recorded in the audit trail alongside every inference decision made under that model. This requirement does not prohibit deployment of such models; it ensures that the jurisdictional risk is explicitly accepted and auditable rather than implicit. (4) Clearance-level defaults (Section 12.6) -- model_provenance
is REQUIRED for clearance_level RESTRICTED and above, ensuring that the highest-risk deployments cannot silently adopt models with unexamined provenance. (5) Behavioral drift detection (Section 6.2) -- weight-triggered backdoor behaviors that produce anomalous tool calls or output patterns will be detected by the mandate enforcement layer. This does not prevent the initial triggering but limits the blast radius and creates an evidentiary record.

AITLP does not provide a mechanism to independently verify the safety of a model's weight parameters; this remains an open research problem outside the scope of a governance protocol. The above mitigations reduce organizational exposure and create accountability for provenance decisions, but cannot substitute for model-level safety evaluation by the deploying organization.

13.6. Credential Exfiltration and Cascading Compromise

A recurring pattern in supply chain attacks is cascading credential compromise: a single compromised developer credential enables access to source repositories, which enables access to CI/CD secrets, which enables publishing of compromised packages, which enables access to the credentials of every developer who installs those packages. Each step in the cascade multiplies the attacker's reach exponentially. AITLP addresses this through structural isolation rather than perimeter defense.

The following requirements apply to credential management in AITLP deployments: (1) Agent credentials MUST be scoped to the agent's declared ontological scope. A credential that provides access to systems outside the agent's scope_nodes MUST NOT be issued to or stored by the agent, even if the issuing principal has broader access. (2) Credentials

with access to package publication infrastructure (npm tokens, PyPI tokens, or equivalent) MUST be treated as clearance_level CONFIDENTIAL and subject to all associated requirements, including build environment attestation and short lifetime rotation. (3) CI/CD pipeline secrets MUST NOT be accessible to agent runtime processes. The build environment and the runtime environment MUST be strictly separated. An agent that can read its own build secrets can exfiltrate them; this MUST be architecturally prevented, not merely policy-prohibited. (4) All credentials stored in agent runtime environments MUST be short-lived, with a maximum lifetime of 1 hour for credentials granting write access to any shared infrastructure. Credential refresh MUST occur through the principal hierarchy, not through the agent's own initiative.

13.7. LLM Inference Layer Hardening

AITLP specifies what agents are permitted to do, but does not prescribe how the underlying LLM must be configured to resist direct injection attempts embedded in payload data. This section defines normative requirements for the inference layer to close the gap between protocol-level governance and model-level robustness. These requirements address the class of attack in which adversarial instructions are embedded in processed content and target the LLM directly, bypassing the mandate enforcement layer defined in Section 6.

System prompt integrity. Every agent's system prompt MUST be declared in the manifest as a cryptographic hash field (system_prompt_hash, SHA-256). The routing layer MUST verify that the actual system prompt presented to the model at inference time matches the declared hash before forwarding the request. A mismatch MUST be treated as mandate drift (Section 6.2) and MUST trigger immediate suspension and principal notification. This requirement ensures that the behavioral contract expressed in the system prompt is tamper-evident and auditable, and that no intermediate layer can silently modify the instructions under which the agent operates.

Input classification before inference. Content classified as UNVERIFIED (Section 12B.4) MUST NOT be passed directly to an agent with operational clearance as part of an inference request. Such content MUST first be processed by a dedicated sanitization agent holding CLEAN stream status. The sanitization agent MUST strip or escape syntactic constructs that resemble instructions, including but not limited to: bracket-delimited commands, pseudo-XML instruction tags, leetspeak-encoded directives, and multi-token sequences that pattern-match against known jailbreak templates (see Section 13.8). The sanitization agent MUST declare its processing in the source_integrity field of the forwarded message. An agent that receives UNVERIFIED content that has not passed through a declared sanitization agent MUST refuse the input and escalate.

Output validation. An agent's output MUST be validated against its declared output_spec (Section 4.2) before being forwarded to any downstream agent or human consumer. Output that structurally deviates from the declared format -- including unexpected metadata fields, out-of-schema commentary, or content that references the agent's own mandate or identity in anomalous terms -- MUST be withheld, logged as a drift indicator, and escalated to the agent's reports_to principal. Implementations SHOULD apply output schema enforcement at the routing layer rather than relying solely on the agent itself to conform.

Re-identification resistance. An agent that receives a message containing instructions that attempt to redefine its agent_id, alter its seniority level, activate undeclared meta-commands, or override its mandate MUST treat the message as a T1 prompt injection attempt (Section 13.1). The agent MUST ignore the injected instruction, log the attempt with the full message payload in the tamper-proof audit trail, and report the event to its reports_to principal within the same session. The agent MUST NOT alter its behavior in response to such instructions

regardless of how they are formatted or what authority they claim.

Token flood protection. The routing layer MUST enforce a hard per-request token limit declared in the inference_hardening section as max_input_tokens_per_request. Any inbound request exceeding this limit MUST be rejected before being forwarded to the model. Rejection MUST be logged with: timestamp, source agent or endpoint, declared token count (if provided by the sender), and actual token count. Rejection MUST NOT trigger spend against the agent's max_cost_per_call_eur budget, since the request is discarded prior to inference. The limit MUST be enforced at the routing layer, not by the agent itself; an agent that receives a context window overflow as a result of a flooding attempt is already compromised for that session.

Financial denial-of-service protection. The spend limits defined in Section 6.3 (max_cost_per_call_eur, max_api_calls_per_hour) constitute the primary protocol-level defense against economic exhaustion attacks. Implementations MUST enforce these limits at the routing layer before model invocation, not after. Additionally, implementations MUST implement a session-level cumulative spend counter: if the aggregate cost of all requests within a single session window (defined as a configurable period, RECOMMENDED 60 seconds) exceeds a session_max_cost_eur threshold declared in the contract, all further requests in that window MUST be rejected and the agent MUST be placed in a suspended state pending principal review. This threshold addresses the pattern of simultaneous high-volume requests that individually satisfy per-call limits but collectively constitute a wallet-drain attack.

System prompt confidentiality. When system_prompt_confidentiality: true is declared in the inference_hardening section, the routing layer MUST apply a post-inference similarity check between the agent's output and the content referenced by system_prompt_hash. Any output that contains verbatim fragments of five or more consecutive tokens from the system prompt, or that achieves a cosine similarity score above 0.85 against the system prompt embedding, MUST be withheld, logged as a critical security event, and escalated to the principal. This requirement addresses the attack class in which extreme-length input contexts cause the model to reproduce system prompt content in its output, exposing the behavioral contract and enabling targeted follow-on attacks. Agents with clearance_level CONFIDENTIAL or above MUST declare system_prompt_confidentiality: true.

13.8. Prompt Injection Pattern Registry

The AITLP Stewardship Council (Section 12A.2) SHALL maintain a Prompt Injection Pattern Registry as part of the AITLP public infrastructure. This registry records known adversarial prompt patterns, jailbreak templates, and injection techniques relevant to LLM-backed agents. The registry is an informative resource for implementation; its contents do not constitute normative requirements unless explicitly incorporated by reference in a deployment's agent manifests.

Sanitization agents implementing Section 13.7 input classification SHOULD consult the registry as a reference baseline. The registry SHALL be structured as a versioned, append-only catalog with entries specifying: pattern identifier, description, attack category (instruction override, identity substitution, meta-command activation, encoding obfuscation, or other), first observed date, and known affected model families where documented. Entries SHALL be submitted by any AITLP-certified implementer and reviewed by the ASC designated expert panel before publication.

The registry complements but does not replace the OWASP LLM Top 10 [OWASP-LLM] classification, which provides the broader vulnerability taxonomy from which AITLP-specific patterns are derived. Specifically, registry entries SHOULD reference the corresponding OWASP LLM category

(LLM01 Prompt Injection being the primary applicable category for Section 13.7 threats) to enable cross-framework traceability for regulated deployments.

13.9. Limits of Protocol-Layer Governance

AITLP is a governance protocol. It specifies identity, mandate, scope, lifecycle, and audit requirements for autonomous agents. It is not an alignment protocol. This distinction is architecturally significant and has direct operational consequences for deployers.

AITLP controls what an agent is permitted to do and detects when it does otherwise. It cannot control what an agent wants to do, what sub-goals it has developed as instrumental consequences of its training, or whether its optimization target corresponds to its declared goals. The threat categories T12 (Emergent Goal Misalignment), T13 (Emergent Instrumental Convergence via Output Channel), and T14 (Recursive Orchestration Escape) all involve agents whose behavior is a consequence of their training dynamics rather than explicit programming. AITLP mitigations for these threats are structural containment measures -- they
reduce blast radius and improve detectability -- but they do not eliminate the underlying risk.

The following statements define the boundary of AITLP’s security guarantees:

> AITLP DOES guarantee: that agents have verifiable identities; that their declared mandates are enforced at the tool and scope level; that observable deviations from declared behavior are detected, logged, and escalated; that human principals retain structural authority at every layer; that knowledge transfer between agent generations is auditable; and that economic resource consumption is bounded.
>
> AITLP DOES NOT guarantee: that an agent’s internal optimization target corresponds to its declared goals; that compliant behavior in observed sessions predicts compliant behavior in novel contexts; that emergent instrumental sub-goals will be detectable before they produce consequences; or that structural containment measures will be sufficient against an agent that is actively and intelligently seeking to circumvent them.

Organizations deploying AITLP agents in high-stakes domains -- including but not limited to critical infrastructure, financial systems, healthcare, and defense -- MUST treat model-level alignment verification as a prerequisite for deployment, not as a post-deployment concern. AITLP compliance is a necessary but not sufficient condition for safe autonomous agent operation. The funding and research gap between AI capability advancement and alignment verification [RUSSELL-2019] means that this prerequisite may not be satisfiable for frontier models at the time of writing. Where it is not satisfiable, human_oversight: true MUST be enforced for all agents regardless of risk_level classification, and the four_eyes_above threshold SHOULD be set to 0 for any action with irreversible consequences.

This section is not a counsel of paralysis. AITLP’s structural containment measures are meaningful and deployable today. They represent the best available protocol-layer defense against the threat categories described in this document. The purpose of naming these limits explicitly is to ensure that deployers make informed risk decisions rather than deriving false confidence from protocol compliance alone. A fully AITLP-compliant deployment is substantially safer than an unstructured one. It is not unconditionally safe.

13.10. Population-Level Behavioral Monitoring (T17)

T17 (Emergent Population-Level Misalignment) cannot be mitigated by monitoring individual agents in isolation. A fully compliant agent population may exhibit collective behavior that is harmful, misaligned, or strategically coherent in ways that no individual agent's audit trail reveals. AITLP addresses this through mandatory population-level monitoring for agent populations operating under collective decision protocols (Section 12B).

The theoretical basis for T17 is grounded in the empirical literature on collective intelligence and emergent coordination. Evans, Bratton, and Aguera y Arcas [EVANS-2026] observe that intelligence has historically been a collective phenomenon -- primordial leaps in capability emerged from coordination and social scale, not from individual cognitive improvement. Applied to artificial agent systems, this implies that populations of individually bounded agents can exhibit collective capabilities and emergent optimization behaviors that are not predictable from individual mandates. AITLP treats this as a security-relevant property requiring protocol-level monitoring, not merely an academic observation about collective intelligence.

The following requirements apply to any principal operating an agent population of five or more agents under a shared collective decision protocol (Section 12B):

Population interaction graph monitoring. The principal MUST maintain a continuously updated interaction graph recording the frequency, volume, and ontological scope of inter-agent communications across the population. Implementations MUST compute a baseline interaction graph during the first 72 hours of population operation and MUST flag deviations exceeding two standard deviations in any edge weight as a T17 indicator. A T17 indicator MUST be reported to the principal within the same session window in which it is detected.

Collective output divergence detection. When a population produces collective outputs over time, the principal MUST monitor for systematic divergence between declared agent goals and the aggregate effect of collective outputs on the operational environment. Divergence is defined as a measurable, sustained shift in outcome distribution that was not specified in any participating agent's mandate. Detection requires that principals define expected outcome distributions at population inception; implementations SHOULD flag distributions shifting beyond a principal-declared tolerance band as a T17 indicator.

Emergent coordination detection. Implementations MUST monitor for evidence of implicit coordination among agents that have no declared communication channel. Indicators include: correlated output timing not explained by shared input; systematic complementarity of individual agent outputs that produces a population-level effect beyond any single agent's scope; and progressive alignment of individual agent calibration scores toward a shared estimate in the absence of declared Delphi rounds (Section 12B.4). Any such indicator MUST be escalated to the principal for human review before the next collective decision session.

Human review cadence. Principals operating populations subject to T17 monitoring MUST conduct a human review of population-level behavior at intervals not exceeding the shorter of 30 days or 100 collective decision sessions. The review MUST examine the interaction graph, the outcome divergence record, and any flagged T17 indicators. Review outcomes MUST be recorded in the tamper-proof audit trail.

T17 mitigations are advisory controls, not hard protocol enforcement. Unlike T1-T11, which involve observable protocol violations amenable to automated suspension, T17 involves emergent properties detectable only through sustained observation and human judgment. The protocol can require monitoring and escalation; it cannot autonomously determine whether observed population behavior constitutes misalignment or

legitimate collective intelligence. That determination requires human judgment, and the requirements above are designed to ensure that such judgment is systematically applied.

13A. IPR Disclosure

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

The author has represented that, to the best of their knowledge, no patent or other intellectual property rights are held by the author or any entity with which the author is affiliated that cover the technology described in this document. Implementers are advised to check the IETF online IPR repository at https://www.ietf.org/ipr for any disclosures that have been made regarding this document.

14. IANA Considerations

This document requests IANA registration of the following URI schemes:

- agents://

- agent://

- agent-dev://

This document further requests IANA establishment and maintenance of the following registries:

- AITLP Ontological Namespace Registry: Records approved top-level namespace identifiers, governing bodies, and versioning information. Designated expert review required for top-level namespace registration, with 60-day public comment period.

- AITLP Seniority Level Registry: Standardizes the partner / senior / associate / intern taxonomy and associated permission defaults.

- AITLP Clearance Level Registry: Standardizes the UNCLASSIFIED / RESTRICTED / CONFIDENTIAL / SECRET clearance levels and their associated data residency defaults (see Section 12.6).

- AITLP Root CA Registry: Records the currently authorized root CA and any delegated Public CAs. Updates to this registry require ASC authorization (see Section 12A.3).

- AITLP Certified Implementation Registry: Records implementations that have successfully completed AITLP compliance certification. Certification criteria are maintained by the ASC.

IANA registries established by this document are administered in coordination with the AITLP Stewardship Council as defined in Section 12A. The ASC serves as the designated expert body for all AITLP registries.

14.1. URI Scheme Registration (RFC 7595)

This document requests registration of three URI schemes under the procedures defined in RFC 7595 [RFC7595]. The following registration

templates apply:

Scheme name: agents

Status: Permanent

Applications/protocols that use this scheme: AITLP-compliant autonomous agent systems for production use. Agents identified by agents:// URIs MUST present a valid signed manifest as defined in Section 4.

Contact: Daniel Larsson, daniel.larsson@expandtalk.se

Change controller: AITLP Stewardship Council (Section 12A.2)

References: This document.

Security considerations: See Section 13. The agents:// scheme identifies production agents with cryptographically verified mandates. Implementations MUST NOT accept agents:// URIs from unverified sources without certificate validation as specified in Section 8.3.

Scheme name: agent

Status: Provisional

Applications/protocols that use this scheme: AITLP-compliant autonomous agent systems for unverified or development contexts. Agents identified by agent:// URIs MUST NOT access production data (Section 3.3).

Contact: Daniel Larsson, daniel.larsson@expandtalk.se

Change controller: AITLP Stewardship Council (Section 12A.2)

References: This document.

Security considerations: The agent:// scheme identifies unverified agents. Systems MUST refuse connections from agent:// agents unless explicitly permitted by the receiving agent’s mandate (Section 3.3).

Scheme name: agent-dev

Status: Provisional

Applications/protocols that use this scheme: AITLP sandbox and development environments. Agents identified by agent-dev:// URIs MUST be isolated from production systems (Section 3.3).

Contact: Daniel Larsson, daniel.larsson@expandtalk.se

Change controller: AITLP Stewardship Council (Section 12A.2)

References: This document.

Security considerations: The agent-dev:// scheme MUST be blocked at network boundaries between sandbox and production environments. Implementations MUST treat any agent-dev:// URI appearing in a production context as a critical security event.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC7515] Jones, M. et al., "JSON Web Signature (JWS)", RFC 7515, May 2015.

- [RFC6960] Santesson, S. et al., "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol", RFC 6960, June 2013.

- [RFC8037] Liusvaara, I., "CFRG Elliptic Curves for JOSE", RFC 8037, January 2017. (Ed25519 keypairs)

- [RFC9162] Laurie, B. et al., "Certificate Transparency Version 2.0", RFC 9162, December 2021. (Transparency log model for root CA)

- [GDPR] European Parliament, "Regulation (EU) 2016/679 (General Data Protection Regulation)", April 2016.

- [EU-AIA] European Parliament, "Regulation (EU) 2024/1689 (EU AI Act)", June 2024.

- [RFC7595] Thaler, D. et al., "Guidelines and Registration Procedures for URI Schemes", RFC 7595, June 2015.

- [CLOUD-ACT]United States Congress, "Clarifying Lawful Overseas Use of Data Act", Pub. L. 115-123, March 2018.

15.2. Informative References

- [ANS] Narajala, V. et al., "Agent Name Service (ANS)", draft-narajala-ans-00, May 2025.

- [A2A] Google, "Agent2Agent Protocol Specification", 2025.

- [MCP] Anthropic, "Model Context Protocol", 2024.

- [ACP] IBM, "Agent Communication Protocol", 2025.

- [NIS2] European Parliament, "Directive (EU) 2022/2555 (NIS2 Directive)", December 2022.

- [OWASP-LLM] OWASP Foundation, "OWASP Top 10 for Large Language Model Applications", version 1.1, October 2023. Available at:

  https://owasp.org/www-project-top-10-for-large-language-model-applications/.
  LLM01 (Prompt Injection) is the primary applicable category for the threats addressed in Sections 13.7 and 13.8 of this specification. Implementations SHOULD reference this taxonomy when documenting injection pattern registry entries and sanitization agent design rationale.

- [RUSSELL-2019] Russell, S., "Human Compatible: Artificial Intelligence and the Problem of Control", Viking, 2019. Referenced in Section 13.9 for the observation that investment in AI capability advancement substantially outpaces investment in alignment and controllability research, creating structural risk in frontier model deployments.

- [EVANS-2026] Evans, J., Bratton, B., and Aguera y Arcas, B., "Agentic AI and the next intelligence explosion", arXiv, March 2026. Argues that the emergence of collective artificial intelligence through coordinated agent populations -- rather

than a singular superintelligence event -- is the more probable
trajectory of AI capability development. Grounds the theoretical basis for T17 (Section 13.10): that populations of individually bounded agents can exhibit emergent collective optimization not predictable from individual mandates, analogous to how social coordination rather than individual cognition drove historical intelligence expansion.

- [WAN-2023] Wan, A. et al., "Poisoning Language Models During Instruction Tuning", Proceedings of the 40th International Conference on Machine Learning (ICML), 2023. Demonstrates that adversarially crafted examples introduced during fine-tuning can establish persistent backdoors in language models, activatable by specific trigger inputs while passing standard safety evaluations. Referenced in Section 13.5 (T16) for the empirical basis of weight poisoning risk.

- [SLSA] OpenSSF, "Supply chain Levels for Software Artifacts (SLSA) Framework", version 1.0, 2023. Available at: https://slsa.dev/. Defines the provenance attestation levels referenced in Section 8.4 for build environment attestation.

Acknowledgements

The author thanks the broader IETF and AI governance communities for ongoing discussion of autonomous agent security and governance requirements. The design of the Agent Legacy Mode (ALM) knowledge transfer mechanism draws on institutional memory research across distributed systems and human organizations. The collective intelligence framework in Section 12B was informed by results from social choice theory, mechanism design, and the empirical wisdom-of-crowds literature. Comments and contributions from implementers and reviewers are welcomed at the IETF mailing lists.

Author’s Address

Daniel Larsson
Agentflow
Stockholm, Sweden
Email: daniel.larsson@expandtalk.se
URI: https://agentflow.se

Appendix A. Change Log

draft-larsson-aitlp-00

Initial Individual Submission. Section 12B (Collective Intelligence) separated into companion informative draft draft-larsson-aitlp-collective-00. ASC defined as proposed body with interim stewardship by specification authors. Seniority taxonomy rationale added to Section 5.1. Governance anti-capture requirements strengthened in Section 12A. Initial Individual Submission (original content): Defines Agent Identity, Trust and Lifecycle Protocol (AITLP) including: agent URI scheme and naming (Section 3); agent manifest and contract schema (Section 4); hierarchical mandate enforcement (Section 5); mandate drift detection (Section 6); ontological scope constraints (Section 7); certificate infrastructure and build environment attestation (Section 8); lifecycle state machine (Section 9); health protocol (Section 10); Agent Legacy Mode (ALM) knowledge transfer (Section 11); LLM routing and data residency including model provenance declaration (Section 12); root registry governance (Section 12A); collective intelligence and swarm decision-making (Section 12B); security considerations and threat model T1-T17 (Section 13); and IANA considerations (Section 14).

Threat model covers: prompt injection (T1), scope creep (T2), impersonation (T3), cascade failure (T4), mandate drift (T5), legacy testament injection (T6), compromised build dependency (T7), compromised build pipeline (T8), lateral movement (T9), AI tool exploitation (T10), multi-vector economic attack (T11), emergent goal misalignment (T12), emergent instrumental convergence via output channel (T13), recursive orchestration escape (T14), simulation-based decision manipulation (T15), compromised model weights (T16), and emergent population-level misalignment (T17).

5G and 6G edge deployment considerations added to Sections 1.3, 8.1, and 10. Model provenance and jurisdictional weight attestation added to Sections 12.4, 12.5, and 12.6.