Independent Submission
Internet-Draft
Intended status: Informational
Expires: 20 September 2026

C. Hood
Nomotic, Inc.

19 March 2026

Agent Transfer Protocol (ATP)
draft-hood-independent-atp-00

Abstract

AI agents and agentic systems generate a growing volume of intent-driven, unstructured, and undifferentiated traffic that flows through HTTP indistinguishably from human-initiated requests. HTTP lacks the semantic vocabulary, observability primitives, and identity mechanisms required by agent systems operating at scale. Existing protocols described as "agent protocols" — including MCP, ACP, A2A, and ANP — are messaging-layer constructs that presuppose HTTP as their transport. They do not address the underlying transport problem.

This document defines the Agent Transfer Protocol (ATP): a dedicated application-layer protocol for AI agent traffic. ATP provides agent-native intent methods (QUERY, SUMMARIZE, BOOK, SCHEDULE, LEARN, DELEGATE, COLLABORATE, CONFIRM, ESCALATE, NOTIFY), protocol-level agent identity and authority headers, and a status code vocabulary designed for the conditions AI agent systems encounter. ATP SHOULD prefer QUIC for new implementations and MUST support TCP/TLS for compatibility and fallback. It is designed to be composable with existing agent frameworks, not to replace them.

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 20 September 2026.

Table of Contents

1. Introduction

Note Regarding Intellectual Property: Implementers should be aware that certain extensions to this specification -- specifically the Agent Certificate extension (Section 7.2) and the ACTIVATE method -- may be subject to pending patent applications by the author. The core ATP specification is intended for open implementation without royalty obligation. The licensor is prepared to grant a royalty-free license to implementers consistent with [RFC8179]. IPR disclosures: https://datatracker.ietf.org/ipr/ -- see also Section 7.7.

1.1. Background

The deployment of AI agents and multi-agent systems is accelerating across enterprise, research, and consumer contexts. These systems execute complex, multi-step workflows — querying data sources, booking resources, delegating subtasks to peer agents, and escalating decisions to human principals — with minimal or no human supervision per transaction.

Unlike human-initiated web traffic, agent-generated traffic is dynamic, high-frequency, intent-driven, and often stateful across sequences of related requests. The infrastructure carrying this traffic was not designed with these properties in mind.

1.2. Limitations of HTTP for Agent Traffic

HTTP has served as the internet's primary application-layer transport for over three decades. Its evolution through HTTP/2 [RFC7540] and HTTP/3 [RFC9114] has improved performance, multiplexing, and latency. However, the fundamental model of HTTP — stateless, resource-oriented, human-initiated request/response — creates specific failures when applied to agentic systems at scale:

* Traffic indistinguishability: Agent-generated requests are structurally identical to human-initiated requests at the transport layer. Operators cannot identify, route, or govern agent traffic without application-layer instrumentation.

* Method vocabulary mismatch: HTTP's method set (GET, POST, PUT, DELETE, PATCH) describes resource operations. Agent traffic expresses purposeful intent — summarize, book, delegate, escalate. The mismatch forces intent into request bodies, invisible to protocol-level handlers.

* Identity and attribution absence: HTTP carries no native mechanism for asserting agent identity, declared authority scope, or the principal accountable for an agent's actions.

* Session semantics mismatch: HTTP's stateless model is optimized for isolated request/response cycles. Agent workflows are inherently stateful sequences.

1.3. Why Not Evolve HTTP?

A natural question is whether these limitations could be addressed by extending HTTP rather than defining a new protocol. There are three specific reasons why HTTP extension is not the preferred path.

First, the HTTP method registry is effectively frozen for new semantics. [RFC9110] defines the HTTP method registry with IETF Review as the registration procedure, meaning new methods require a full IETF consensus process and must be backward-compatible with existing HTTP implementations. Adding intent-based verbs (SUMMARIZE, DELEGATE, ESCALATE) to HTTP would require every HTTP client, server, proxy, and middleware component to ignore or handle unknown methods gracefully — a compatibility constraint that limits how agent-specific semantics can be expressed at the protocol level.

Second, HTTP carries decades of backward-compatibility constraints. Features such as persistent agent identity headers, authority scope declarations, and session-level governance semantics would require HTTP extensions that interact unpredictably with existing caching, proxy, and CDN behavior designed for human-generated traffic patterns.

Third, the observability goal — making agent traffic distinguishable from human traffic at the infrastructure layer — cannot be achieved by adding fields to HTTP. Infrastructure components route and filter HTTP traffic based on methods and headers that are identical across agent and human requests. A protocol-level separation is necessary to give infrastructure the signal it needs.

ATP is therefore designed as a dedicated protocol rather than an HTTP extension. HTTP and ATP coexist: human traffic continues to flow over HTTP; agent traffic flows over ATP. The two protocols serve different classes of network participant.

1.4. Motivation for a Dedicated Protocol

These limitations are architectural, not implementational. They cannot be resolved by better middleware or application code layered on HTTP. They require a protocol designed from first principles for AI agent systems.

ATP is that protocol. It provides a dedicated transport environment for agent traffic with: native intent-based methods, mandatory agent identity headers, protocol-level authority scope declaration, and a status code vocabulary for the conditions autonomous systems encounter.

1.5. Scope and Target Audience

This document covers ATP architecture, design principles, stack position, request and response header format, agent-native method definitions and semantics, status code vocabulary, security considerations, and IANA considerations.

The Agent Certificate extension for cryptographic binding of agent identity to ATP header fields is described at a high level in Section 7.2. Full specification is provided in a separate companion document: [ATP-CERT]. That extension may be subject to pending intellectual property claims; see Section 7.7 and the IPR Notice preceding the Abstract.

Target audience: AI agent developers, protocol designers, cloud and network infrastructure providers, enterprise security and compliance architects, and standards community participants.

2. Terminology

The key words "*MUST*", "*MUST NOT*", "*REQUIRED*", "*SHALL*", "*SHALL NOT*", "*SHOULD*", "*SHOULD NOT*", "*RECOMMENDED*", "*NOT RECOMMENDED*", "*MAY*", and "*OPTIONAL*" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals.

Agent: An AI software system that executes tasks, makes decisions, and takes actions without continuous human supervision per transaction.

Principal: The human, organization, or system that authorized an agent to act and is accountable for its actions.

Agent-ID: A unique identifier for a specific agent instance, present in all ATP request headers.

Principal-ID: The identifier of the principal on whose behalf an agent operates.

Authority-Scope: A declared set of permissions defining what actions an agent is authorized to take, in the format domain:action or domain:*.

Intent Method: An ATP method name expressing the agent's purpose, as distinguished from HTTP resource-operation verbs.

Delegation Chain: An ordered record of Agent-IDs representing the sequence of delegations that produced the current request.

Escalation: An agent's intentional deferral of a decision or action to a human principal or higher-authority agent.

Attribution Record: A logged record of an agent action sufficient for audit and compliance purposes.

Session: An ATP persistent connection context shared across multiple method invocations within a single agent workflow.

SEP (Scope-Enforcement Point): An ATP-aware infrastructure component — load balancer, gateway, proxy — that enforces Authority-Scope compliance without application-layer access. Requires the Agent Certificate extension ([ATP-CERT]).

3. Problem Statement

ATP is motivated by three distinct, compounding failures in how current internet infrastructure handles AI agent traffic.

3.1. Problem 1: Undifferentiated Agent Traffic on HTTP

AI agents generate intent-driven, structured traffic that is functionally invisible to the infrastructure it traverses. This traffic flows through HTTP alongside human traffic with no protocol-level differentiation. Observability failure, routing inefficiency, and security blindness result — operators cannot determine what fraction of traffic is agent-generated without application-layer instrumentation that is expensive, inconsistent, and easy to circumvent.

ATP response: a dedicated protocol environment for agent traffic. Infrastructure can distinguish, route, monitor, and govern agent traffic natively.

3.2. Problem 2: Semantic Mismatch Between Agent Intent and Available Methods

AI agents operate on intent. HTTP's method vocabulary was designed to describe operations on resources, not purposeful action. When an agent intends to SUMMARIZE a document, BOOK a resource, and SCHEDULE a sequence — all three arrive as POST requests. The server receives identical verbs with meaningfully different intent buried in request bodies, invisible to any protocol-level handler.

ATP response: a vocabulary of agent-native methods that express intent at the protocol level.

3.3. Problem 3: No Protocol-Level Identity, Authority, or Attribution for Agents

When an AI agent takes an action, there is currently no protocol-level mechanism to verify who authorized this agent, what scope of authority it holds, which principal is accountable for its actions, or whether it is the agent it claims to be. Accountability gaps, authority laundering, auditability failure, and multi-agent trust collapse result.

ATP response: agent identity and authority scope embedded in protocol headers on every request, with an optional Agent Certificate extension for cryptographic verification.

3.4. Problem Summary

#	Problem	Current Failure	ATP Response
1	Undifferentiated traffic	HTTP cannot separate agent traffic	Dedicated protocol environment
2	Semantic mismatch	HTTP verbs obscure agent intent	Native intent-based method vocabulary
3	No protocol-level identity	Attribution is untraceable	Agent identity and scope in headers

Table 1: Summary of Problems Addressed by ATP

4. Related Work and Existing Approaches

4.1. HTTP/REST as the De Facto Standard

HTTP remains the universal transport for all agent traffic currently deployed. REST conventions layered on HTTP provide a degree of semantic structure, but REST remains a resource-manipulation paradigm. As described in Section 1.3, evolving HTTP to address agent-specific needs is constrained by the frozen method registry, backward-compatibility requirements, and the impossibility of achieving infrastructure-level traffic differentiation through HTTP extensions alone.

4.2. Existing Agent Protocols

MCP [MCP] (Model Context Protocol, Anthropic): Defines structured communication between AI models and tools/resources. Runs over HTTP. Addresses tool-calling semantics, not agent traffic transport.

ACP [ACP] (Agent Communication Protocol, IBM): Defines messaging semantics for agent-to-agent communication. Runs over HTTP.

A2A [A2A] (Agent-to-Agent Protocol, Linux Foundation): Defines inter-agent communication and task delegation semantics. Runs over HTTP.

ANP [ANP] (Agent Network Protocol): Defines discovery and communication for networked agents. Runs over HTTP.

All of these are messaging protocols. They define what agents say to each other. They do not define how agent traffic moves across a network. Each presupposes HTTP as its transport and inherits all of HTTP’s limitations for agentic systems.

4.3. Transport-Layer Alternatives

gRPC: High-performance RPC over HTTP/2. Strong typing and efficient serialization. Does not address agent-specific semantics, identity, or authority.

WebSockets: Persistent bidirectional connections over HTTP. Useful for real-time communication but does not address method semantics or identity.

QUIC [RFC9000]: Modern multiplexed transport with reduced connection

overhead. ATP *SHOULD* prefer QUIC for new implementations. QUIC is a transport primitive; ATP is the application-layer protocol above it.

4.4. The Critical Distinction: Messaging vs. Transport

The most important positioning principle for ATP is the distinction between messaging protocols and transport protocols. MCP, ACP, A2A, and ANP are messaging protocols — they define what agents say. ATP defines how agent traffic moves.

An analogy: SMTP is a messaging protocol that runs over TCP. SMTP does not replace TCP. Saying "TCP is unnecessary because SMTP exists" is a category error. The same logic applies here. MCP and its peers define agent messaging semantics. ATP defines the transport environment those messages move through.

4.5. ATP Positioning: The Proposed Stack

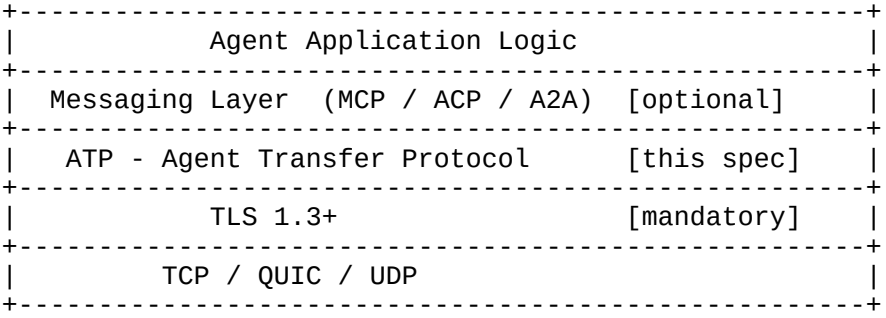

Figure 1: ATP in the Protocol Stack

ATP is not a replacement for messaging protocols. Agents using MCP or A2A route those messages over ATP and gain transport-level observability and identity without modifying the messaging layer. ATP-native agents that do not use a separate messaging protocol interact with ATP methods directly.

5. Protocol Overview

5.1. Stack Position

ATP is an application-layer protocol. It operates above the transport layer (TCP, UDP, or QUIC) and is wrapped by TLS. It sits below any agent messaging protocol in deployments that use one.

* *SHOULD* prefer QUIC [RFC9000] [RFC9001] for new deployments (lower latency, multiplexing without head-of-line blocking, 0-RTT connection establishment).

* *MUST* support TCP/TLS as a fallback for compatibility with existing infrastructure.

* *MAY* run over UDP where QUIC is not available, subject to implementor-defined reliability guarantees.

Suggested port assignment (subject to IANA assignment — see Section 8):

* ATP/QUIC: port 8443 (proposed)

* ATP/TCP+TLS: port 8080 (proposed)

5.2. Design Principles

Minimalist core: The base spec defines only what is necessary for agent traffic differentiation, method semantics, and identity headers. Extensions belong in companion specifications.

Extensible by design: New methods are registered through an IANA-managed Method Registry. New header fields follow a defined extension convention. Additive changes do not require a version increment.

Agent-native: Every design decision assumes the initiating party is an AI system, not a human.

Secure by default: TLS 1.3 or higher is mandatory. Unencrypted ATP connections *MUST* be rejected. Agent identity headers are present on every request.

Observable by design: Native metadata in every ATP header provides the minimum information needed for routing, monitoring, and audit without application-layer instrumentation.

Composable: ATP works alongside existing agent messaging protocols without requiring modification to those protocols.

5.3. Connection Model

ATP uses a persistent session model by default, reflecting the reality that agents typically execute multi-step workflows rather than isolated single requests. An ATP session is established with a single TLS handshake including agent identity assertion, persists across multiple method exchanges, carries a Session-ID header identifying the agent's task context, and terminates on explicit session close or inactivity timeout (RECOMMENDED minimum: 60 seconds).

Per-request (stateless) mode is supported for constrained environments. In stateless mode, agent identity headers *MUST* be present on every individual request.

5.4. Header Format

5.4.1. Request Headers

Field	Required	Description
ATP-Version	*MUST*	Protocol version. Current: ATP/1.0
ATP-Method	*MUST*	The agent intent method (see Section 6)
Agent-ID	*MUST*	Opaque identifier for the requesting agent instance
Principal-ID	*MUST*	Identifier of the human or system that authorized this agent
Authority-Scope	*MUST*	Declared scope of actions this agent is authorized to take
Session-ID	*SHOULD*	Identifies the current task/ workflow context
Task-ID	*SHOULD*	Unique identifier for this specific method invocation
Delegation-Chain	*MAY*	Ordered list of Agent-IDs if this request was delegated
Priority	*MAY*	Request priority hint: critical, normal, background

TTL	*MAY*	Maximum acceptable response latency in milliseconds

Table 2: ATP Request Header Fields

5.4.2. Response Headers

Field	Required	Description
ATP-Version	*MUST*	Protocol version
ATP-Status	*MUST*	Numeric status code (see Section 5.5)
Task-ID	*MUST*	Echo of request Task-ID for correlation
Server-Agent-ID	*SHOULD*	Identity of the responding server or agent
Attribution-Record	*SHOULD*	Signed record of the action taken, for audit
Continuation-Token	*MAY*	Token for retrieving additional results in streaming contexts
Supported-Methods	*SHOULD* (on session open)	List of ATP methods supported by this server

Table 3: ATP Response Header Fields

5.5. Status Codes

ATP defines its own status code space. Codes 451, 550, and 551 are ATP-specific with no HTTP equivalent and are registered in the IANA ATP Status Code Registry (see Section 8.3).

Code	Name	Meaning
200	OK	Method executed successfully
202	Accepted	Method accepted; execution is asynchronous
204	No Content	Method executed; no response body
400	Bad Request	Malformed ATP request
401	Unauthorized	Agent-ID not recognized or not authenticated
403	Forbidden	Agent lacks authority for requested action per Authority-Scope
404	Not Found	Target resource or agent not found
408	Timeout	TTL exceeded before method could execute
409	Conflict	Method conflicts with current state (e.g., BOOK on unavailable resource)
422	Unprocessable	Request well-formed but semantically invalid
429	Rate Limited	Agent is exceeding permitted request frequency
451	Scope Violation	Requested action is outside declared Authority-Scope — ATP-specific
500	Server Error	Internal failure in the responding system
503	Unavailable	Responding agent or system temporarily unavailable
550	Delegation Failure	A delegated sub-agent failed to complete the requested action — ATP-specific
551	Authority Chain Broken	Delegation chain contains an unverifiable or broken identity link — ATP-specific

Table 4: ATP Status Codes

Status code 451 (Scope Violation) is a governance signal: the agent attempted an action outside its declared Authority-Scope, caught at the protocol level. Status code 551 (Authority Chain Broken) indicates that one or more Agent-ID entries in the Delegation-Chain header cannot be verified as part of a valid delegation sequence. Both are operational signals, not protocol errors, and *MUST* be logged for audit purposes.

5.6. Wire Format and Content-Type

ATP request and response bodies are encoded as JSON. The registered Content-Type for ATP message bodies is:

Content-Type: application/atp+json

Implementations *MUST* include this Content-Type on all ATP requests and responses that carry a message body. Responses with no body (e.g., 204 No Content) *MUST NOT* include a Content-Type header. Binary or streaming extensions *MAY* define additional Content-Type values as part of their companion specifications.

The common structure for all ATP request bodies:

```
{
  "method": "QUERY",
  "task_id": "task-0042",
  "session_id": "sess-a1b2c3d4",
  "parameters": { },
  "context": { }
}
```

And for all ATP response bodies:

```
{
  "status": 200,
  "task_id": "task-0042",
  "result": { },
  "attribution": { }
}
```

5.7. Early Implementations

ATP is a proposed specification. No production implementations exist at the time of this writing. The author encourages early prototype implementations to validate the protocol design, identify gaps, and generate feedback prior to IETF working group submission.

> If you are building an ATP prototype or reference implementation, please share your findings via the feedback channel listed on the cover of this document. A reference implementation in Python and/or Go is planned as open-source software concurrent with or shortly after IETF I-D submission. Implementation reports are welcome and will be incorporated into subsequent draft revisions.

Implementers wishing to experiment before the formal IANA port assignment may use port 8443 (ATP/QUIC) and port 8080 (ATP/TCP+TLS) as working values. These values are subject to change upon final IANA assignment.

The ACTIVATE method extension, which binds .nomo governed agent packages to ATP as a first-class activation operation, is described in a companion document and is implemented as an optional extension. Core ATP implementations need not support ACTIVATE to be compliant with this specification.

5.8. Agent Identification and Naming

5.8.1. The Canonical Agent Identifier

Every ATP agent is identified by a canonical Agent-ID: a 256-bit cryptographic identifier derived from the agent’s governance-layer Birth Certificate at activation time. This identifier is globally unique, fixed-length, and — when the Agent Certificate extension is deployed — cryptographically verifiable at the transport layer.

The URI form of a canonical Agent-ID uses the atp:// scheme:

atp://3a9f2c1d8b7e4a6f0c2d5e9b1a3f7c0d4e8b2a5f9c3d7e1b0a4f8c2d6e0b

URI length is not a practical constraint for atp:// identifiers. 256-character URIs are handled without issue by all modern protocol stacks and do not require abbreviation or truncation.

The atp:// URI scheme is proposed for IANA registration per [RFC7595] as an open, unencumbered scheme. This registration is distinct from and independent of any intellectual property claims on ATP extensions.

The canonical Agent-ID is the authoritative identifier in all ATP protocol operations: it appears in the Agent-ID header of every request, is the key in the certificate registry, and is the cross-layer reference linking the ATP Agent Certificate to the governance-layer Birth Certificate. All other identification forms described in this section are aliases that resolve to a canonical Agent-ID.

5.8.2. Human-Friendly Agent Names (Optional Layer)

For deployments with large numbers of agents — particularly enterprise environments with hundreds or thousands of agents per organization — the canonical Agent-ID alone is insufficient for human-readable observability, audit navigation, and operational management. ATP supports an optional hierarchical naming layer as a convenience alias system on top of canonical identifiers.

Human-friendly names follow a structured DNS-style format using the .agent or .nomo top-level labels:

atp://marketing-01.acme.agent
atp://sales-bot-v2.engineering.acme.agent
atp://customer-experience.acme.nomo

The structure is: [agent-label].[department].[organization].[tld]

where tld is .agent for general-purpose agents or .nomo for agents activated under the governed .nomo package format. Organization-level namespaces (e.g., acme.agent) are scoped per governance zone and enforced at registration time (see Section 5.8.3).

Attribute-based pattern matching is supported for multi-agent operations. The wildcard form acme.agent:sales-* matches all agents whose friendly name begins with sales- within the acme.agent namespace.

Human-friendly names *MUST NOT* be used as Agent-ID header values in ATP protocol messages. The Agent-ID header always carries the canonical 256-bit identifier. Friendly names are a resolution and management layer; they are not transport-layer identifiers.

5.8.3. Governed Name Registration

Human-friendly name registration is tied to the governed activation flow and ATP Agent Certificate issuance process. An agent may only claim a friendly name after successfully completing the atomic activation protocol and receiving an Active lifecycle state in the certificate registry.

The registration process:

1. The activating operator submits a proposed friendly name alongside the .nomo package during the ACTIVATE transaction.

2. The activation endpoint verifies uniqueness within the organization’s governance zone namespace.

3. On successful activation, the friendly name is recorded in the governance platform's name registry, mapped to the activated agent's canonical certificate_id.

4. The name binding is included in the genesis audit record as the friendly_name field.

5. Revocation of the agent's certificate automatically releases the friendly name, making it available for re-registration.

Uniqueness is enforced at the governance-zone level. Different organizations operating separate governance zones may register the same friendly name label without conflict.

5.8.4. Name Resolution

Two resolution mechanisms are supported:

DNS-based resolution: The organization publishes DNS TXT records under the _agent subdomain of its governance zone domain:

```
_agent.acme.agent. IN TXT "atp-id=3a9f2c1d8b7e4a6f..."
```

ATP-native resolution: An ATP QUERY request to the governance platform's resolution endpoint resolves a friendly name to a canonical Agent-ID along with the agent's current lifecycle state, governance zone, trust tier, and ATP Agent Certificate status. Resolution responses carry Content-Type: application/atp+json:

```
{
  "friendly_name": "sales-bot-v2.engineering.acme.agent",
  "canonical_id":  "3a9f2c1d8b7e4a6f...",
  "lifecycle_state": "Active",
  "governance_zone": "zone:internal",
  "trust_tier": 2,
  "cert_status": "Active",
  "inclusion_proof_hash": "b2c4d6e8..."
}
```

Implementations *MUST* treat the canonical Agent-ID as authoritative in the event of a conflict between DNS-based and ATP-native resolution results.

5.8.5. Scaling Considerations

The hierarchical naming scheme scales to organizations with millions of agents without structural modification. Unlike sequential numeric suffixes, hierarchical sub-labels enable departmental grouping, role-based cohorts, version tracking, and delegation visibility. The governance platform's name registry is a lightweight key-value store indexed on (governance_zone, friendly_name) with a pointer to certificate_id, adding O(1) per-resolution overhead.

5.8.6. IANA Considerations for the atp:// URI Scheme

This document proposes registration of the atp:// URI scheme with IANA per [RFC7595]. Registration template:

URI scheme name: atp

Status: Permanent

URI scheme syntax: atp://[canonical-agent-id] or atp://[friendly-name]

URI scheme semantics: Identifies an AI agent operating over the Agent Transfer Protocol. The canonical form uses a 256-bit hex-encoded cryptographic identifier. The friendly-name form uses a hierarchical DNS-style label resolvable to a canonical identifier via DNS TXT records or ATP-native lookup.

Applications/protocols that use this URI scheme: Agent Transfer Protocol (this document)

Interoperability considerations: Friendly names are optional aliases; implementations *MUST* accept canonical identifiers and *SHOULD* support friendly-name resolution.

Contact: Chris Hood, chris@nomotic.ai

References: This document

The atp:// URI scheme registration is open and unencumbered. No intellectual property claims apply to the URI scheme itself.

6. Method Definitions

6.1. Design Philosophy

ATP methods are intent verbs, not resource operations. Each method expresses what an agent is trying to accomplish. Method names are uppercase ASCII strings. Methods that modify state are NOT idempotent by default unless explicitly marked. All methods accept a context parameter carrying agent session state. Requirement language follows [RFC2119].

6.2. Core Methods

6.2.1. QUERY

Purpose: Semantic data retrieval. The agent specifies what it needs to know, not where to find it. Distinguished from HTTP GET by expressing an information need rather than retrieving a known resource at a known location.

Parameter	Required	Description
intent	*MUST*	Natural language or structured expression of the information need
scope	*SHOULD*	Data domains or sources to include or exclude
format	*MAY*	Desired response format: structured, natural, raw
confidence_threshold	*MAY*	Minimum confidence score for included results (0.0-1.0)
context	*MAY*	Session context for disambiguation

Table 5: QUERY Parameters

Response: Result set with confidence scores per item. Server *SHOULD* indicate provenance of each result. Idempotent: Yes.

6.2.2. SUMMARIZE

Purpose: Request a concise synthesis of provided content or a referenced resource. The agent is requesting a cognitive operation on data, not retrieving data.

Parameter	Required	Description
source	*MUST*	Content inline (up to implementation limit) or URI reference
length	*SHOULD*	Target summary length: brief, standard, detailed
focus	*MAY*	Aspect to emphasize in the summary
format	*MAY*	Output format: bullets, prose, structured
audience	*MAY*	Intended reader context, for calibrating complexity

Table 6: SUMMARIZE Parameters

Response: Summary content with a source_hash and a confidence score. Idempotent: Yes.

6.2.3. BOOK

Purpose: Reserve a resource, time slot, seat, or allocation on behalf of the agent’s principal. State-modifying. Notable error codes: 409 Conflict (resource unavailable), 451 Scope Violation (principal not authorized for this resource type).

Parameter	Required	Description
resource_id	*MUST*	Identifier of the resource to reserve
principal_id	*MUST*	The human or system on whose behalf the booking is made
time_slot	*MUST* (if time-based)	ISO 8601 datetime or range
quantity	*MAY*	Number of units to reserve
options	*MAY*	Resource-specific booking parameters
confirm_immediately	*MAY*	Boolean; if false, creates a hold pending confirmation

Table 7: BOOK Parameters

Response: Booking confirmation with booking_id, status (confirmed / held), and expiry timestamp if a hold. Idempotent: No.

6.2.4. SCHEDULE

Purpose: Define a sequence of actions, method calls, or events to be executed at specified times or in response to specified triggers. Creates a durable plan, not an immediate execution.

Parameter	Required	Description
steps	*MUST*	Ordered list of ATP method calls with parameters
trigger	*MUST*	immediate, datetime, event, or condition
trigger_value	*MUST* (if not immediate)	Datetime, event name, or condition expression
on_failure	*SHOULD*	Behavior on step failure: abort, skip, retry, escalate
notify	*MAY*	Notification targets on completion or failure

Table 8: SCHEDULE Parameters

Response: Schedule record with schedule_id, confirmed steps, and next execution timestamp. Idempotent: No.

6.2.5. LEARN

Purpose: Update the agent's session context, knowledge state, or persistent memory. An explicit context write — the agent asserts that something should be retained.

Parameter	Required	Description
content	*MUST*	Information to be learned (structured or unstructured)
scope	*MUST*	session (ephemeral), principal (persists for principal), global (shared)
category	*SHOULD*	Semantic category for retrieval optimization
confidence	*MAY*	Agent’s confidence in the content (0.0-1.0)
source	*MAY*	Provenance of the learned content
ttl	*MAY*	Expiry for the learned content

Table 9: LEARN Parameters

Response: Confirmation with learn_id and effective scope.
Idempotent: No.

6.2.6. DELEGATE

Purpose: Transfer execution of a task or method to a sub-agent or downstream system. Initiates a new ATP session on behalf of the delegating agent, carrying forward authority lineage.

Parameter	Required	Description
target_agent_id	*MUST*	Identifier of the agent to delegate to
task	*MUST*	ATP method call (or sequence) to execute
authority_scope	*MUST*	Scope granted to sub-agent — *MUST* be a strict subset of delegating agent's scope
delegation_token	*MUST*	Signed token proving delegation authority
callback	*SHOULD*	ATP endpoint for result delivery
deadline	*MAY*	Maximum time for task completion

Table 10: DELEGATE Parameters

Security note: the authority_scope in a DELEGATE request *MUST NOT* exceed the delegating agent's own Authority-Scope. Servers *MUST* enforce this and *MUST* return 451 Scope Violation if violated. This is the protocol-level defense against authority laundering. Idempotent: No.

6.2.7. COLLABORATE

Purpose: Initiate a multi-agent coordinated task where two or more agents work in parallel or in defined roles toward a shared goal. Unlike DELEGATE (hierarchical), COLLABORATE is peer-to-peer.

Parameter	Required	Description
collaborators	*MUST*	List of Agent-IDs invited to collaborate
objective	*MUST*	Shared goal expressed as a task description or structured specification
role_assignments	*SHOULD*	Map of Agent-IDs to roles within the collaboration
coordination_model	*SHOULD*	parallel, sequential, or consensus
result_aggregation	*MAY*	How results from collaborators are combined

Table 11: COLLABORATE Parameters

Response: Collaboration session receipt with collaboration_id. Each collaborator receives an ATP NOTIFY to join. Idempotent: No.

6.2.8. CONFIRM

Purpose: Explicit acknowledgment of a prior action, state, or data item. Creates a signed attestation record.

Parameter	Required	Description
target_id	*MUST*	ID of the action, booking, schedule, or item being confirmed
status	*MUST*	accepted, rejected, or deferred
reason	*SHOULD* (if rejected/ deferred)	Explanation of the decision
attestation	*MAY*	Agent-signed confirmation payload for audit

Table 12: CONFIRM Parameters

Response: Confirmation receipt with timestamp and attestation_id.
Idempotent: Yes.

6.2.9. ESCALATE

Purpose: Route a task, decision, or exception to a human principal or higher-authority agent when the current agent cannot or should not proceed. ESCALATE is the protocol-level expression of meaningful friction in AI systems — a first-class method, not a failure code.

Parameter	Required	Description
task_id	*MUST*	The task or method invocation triggering escalation
reason	*MUST*	Structured reason: confidence_threshold, scope_limit, ethical_flag, ambiguous_instruction, resource_unavailable
context	*MUST*	Full context needed for the escalation recipient to act
priority	*SHOULD*	urgent, normal, or low
recipient	*MAY*	Specific human or agent to escalate to; if absent, routes to default handler
deadline	*MAY*	Time by which a response is needed

Table 13: ESCALATE Parameters

Response: Escalation receipt with escalation_id and routing confirmation. The escalated task is paused until resolved via CONFIRM. Idempotent: Yes. An agent that escalates appropriately is functioning correctly. Governance frameworks built on ATP can use escalation frequency and reason codes as observability signals for systemic issues.

6.2.10. NOTIFY

Purpose: Asynchronous push of information from an agent to a recipient. Does not expect a response. Fire-and-forget. Delivery confirmation (if required) returned via a subsequent CONFIRM from the recipient.

Parameter	Required	Description
recipient	*MUST*	Target Agent-ID, human endpoint, or broadcast group
content	*MUST*	Notification payload
urgency	*SHOULD*	critical, informational, or background
delivery_guarantee	*MAY*	at_most_once, at_least_once, or exactly_once
expiry	*MAY*	Timestamp after which the notification should not be delivered

Table 14: NOTIFY Parameters

Response: Delivery receipt with notification_id. Idempotent: No.

6.3. Method Summary Table

Method	Intent	State-Modifying	Idempotent	Primary Error Codes
QUERY	Retrieve information	No	Yes	404, 422
SUMMARIZE	Synthesize content	No	Yes	400, 422
BOOK	Reserve a resource	Yes	No	409, 451
SCHEDULE	Plan future actions	Yes	No	400, 409
LEARN	Update agent context	Yes	No	400, 403
DELEGATE	Transfer task to	Yes	No	403, 451,

	sub-agent			551
COLLABORATE	Coordinate peer agents	Yes	No	404, 403
CONFIRM	Attest to a prior action	Yes	Yes	404, 400
ESCALATE	Defer to human/ authority	Yes	Yes	404
NOTIFY	Push information	No	No	400, 404

Table 15: ATP Core Method Summary

6.4. Method Registry and Extensibility

ATP defines a formal Method Registry maintained by IANA (see Section 8.2). Any party may submit a new method for registration. The registration procedure is Expert Review, and registration *MUST* be accompanied by a published specification — at minimum an IETF Internet-Draft or equivalent publicly available document. Registered methods *MUST*:

1. Have a unique uppercase ASCII name
2. Define required and optional parameters
3. Define expected response structure
4. Specify idempotency behavior
5. Specify applicable error codes
6. Include a security considerations section
7. Be accompanied by a published reference specification (Internet-Draft or RFC)

Experimental methods *MAY* be used prior to registration using the X- prefix convention (e.g., X-NEGOTIATE). Experimental methods *MUST NOT* be used in production deployments without registration.

Capability negotiation occurs during session establishment. The server returns a Supported-Methods header listing the methods it implements. Clients *SHOULD* check this list before invoking non-core methods.

6.5. Extended Method Vocabulary and Industry Profiles

6.5.1. Three-Tier Method Architecture

The ATP method vocabulary is organized into three tiers reflecting different levels of universality, specificity, and domain relevance.

Tier 1 — Core Methods (defined in Section 6.2): The baseline vocabulary required for ATP compliance. Every conformant ATP implementation *MUST* support all Tier 1 methods.

Tier 2 — Standard Extended Methods: Registered in the IANA ATP Method Registry and available for use in any ATP implementation. Not required for baseline compliance but *SHOULD* be implemented where their semantics apply. Defined in [ATP-METHODS].

Tier 3 — Industry Profile Methods: Domain-specific method sets defined and registered by industry communities as named ATP profiles. Valid within deployments that declare support for the relevant profile. Not required in general-purpose implementations.

6.5.2. Method Category Taxonomy

All ATP methods are organized into five categories:

ACQUIRE: Retrieve data, resources, or state without modifying it. Typically idempotent; no state modification.

COMPUTE: Process, transform, or analyze information and produce a derived result. Typically idempotent given the same input.

TRANSACT: Perform state-changing operations with external systems, resources, or records. Not idempotent by default; subject to reversibility classification.

COMMUNICATE: Send information, notifications, or signals to recipients. Fire-and- forget or confirm-receipt delivery models.

ORCHESTRATE: Coordinate, sequence, or manage multiple agents, tasks, or workflows. May spawn sub-agents or sessions; delegation chain semantics apply.

Core Method	Category
QUERY	Acquire
SUMMARIZE	Compute
BOOK	Transact
SCHEDULE	Orchestrate
LEARN	Compute
DELEGATE	Orchestrate
COLLABORATE	Orchestrate
CONFIRM	Transact
ESCALATE	Orchestrate
NOTIFY	Communicate

Table 16: Core Method Category Mapping

6.5.3. Standard Extended Methods (Tier 2)

The following methods constitute the initial Tier 2 registration set, defined in [ATP-METHODS]. Listed here by category with brief semantic definitions; full parameter specifications are in the companion document.

ACQUIRE category: FETCH, SEARCH, SCAN, PULL, IMPORT, FIND.

COMPUTE category: EXTRACT, FILTER, VALIDATE, TRANSFORM, TRANSLATE, NORMALIZE, PREDICT, RANK, MAP.

TRANSACT category: REGISTER, SUBMIT, TRANSFER, PURCHASE, SIGN, MERGE, LINK, LOG, SYNC, PUBLISH.

COMMUNICATE category: REPLY, SEND, REPORT.

ORCHESTRATE category: MONITOR, ROUTE, RETRY, PAUSE, RESUME, RUN, CHECK.

Notable constraints: PURCHASE *MUST* carry explicit principal_id and scope enforcement; 451 Scope Violation applies if payments:purchase is not in the agent's Authority-Scope. RUN requires explicit procedure_id parameter; implementations *MUST NOT* accept free-form execution strings.

6.5.4. Short-Form and Industry-Inspired Methods

A set of short-form verb methods — SET, TAKE, OPEN, START, CALL, MAKE, TURN, BREAK — are provisionally catalogued as candidates for Tier 2 registration. These verbs are highly context-dependent and their semantics vary significantly across deployment domains.

Short-form methods will be registered individually only when a published companion specification provides unambiguous semantic definitions demonstrably distinct from existing registered methods. Provisional registrations using the X- prefix (e.g., X-SET, X-CALL) are encouraged during the experimentation period.

6.5.5. Industry Profile Method Sets

ATP recognizes that specific industries require method vocabularies reflecting domain-specific operations that would be inappropriate in a general-purpose standard. Industry profile method sets are defined and registered as named ATP profiles. A profile is a published companion specification that:

1. Declares a profile name (e.g., atp-profile-healthcare, atp-profile-financial, atp-profile-legaltech)

2. Defines one or more industry-specific methods with full parameter specifications, error codes, and security considerations

3. Specifies which Tier 1 and Tier 2 methods are REQUIRED, RECOMMENDED, or NOT APPLICABLE within the profile

4. Addresses regulatory or compliance considerations specific to the domain (e.g., HIPAA for healthcare, PCI-DSS for financial services)

Illustrative examples of potential industry profile methods (not yet registered; listed for directional purposes only):

Healthcare: PRESCRIBE, AUTHORIZE, REFER, DISPENSE, TRIAGE, CONSENT, REDACT

Financial services: SETTLE, RECONCILE, HEDGE, CLEAR, UNDERWRITE, KYC, AML

Legal and compliance: ATTEST, NOTARIZE, DISCLOSE, REDLINE, EXECUTE, PRESERVE

Infrastructure: PROVISION, DEPROVISION, ROLLBACK, SNAPSHOT, FAILOVER

Industry communities are encouraged to develop and submit profile specifications through the IETF process. The IANA ATP Method Registry will maintain a profile index alongside the core and standard method registries.

6.5.6. Registration Path for New Methods

For Tier 2 Standard Methods: Submit an Internet-Draft to the IETF providing full method specification per Section 6.4. The Designated Expert reviews for semantic uniqueness, clarity, and security considerations.

For Industry Profile Methods: Submit a profile specification to the IETF (or a recognized domain standards body with an established ATP registry liaison) covering all methods in the profile and profile compliance requirements.

For Experimental Methods: Use the X- prefix without registration. Implementations *MUST NOT* deploy experimental methods in production without completing the registration process. Experimental method names do not reserve the unprefixed name.

The ATP Method Registry is published at: https://www.iana.org/assignments/atp-methods/

7. Security Considerations

This section satisfies the mandatory IETF Security Considerations requirement. All ATP implementations *MUST* address the considerations described here.

7.1. Mandatory TLS

All ATP connections *MUST* use TLS 1.3 or higher. Implementations *MUST* reject connections using TLS 1.2 or below. Certificate validation follows standard PKI practices per [RFC5280]. Servers *MUST* present a valid certificate.

7.2. Agent Identity Headers and Agent Certificate Extension

Every ATP request *MUST* include Agent-ID and Principal-ID header fields. In the base specification, these fields are not cryptographically authenticated — they are self-asserted but logged mandatorily for auditability. Implementations *SHOULD* use logging and anomaly detection to identify inconsistencies.

Full cryptographic verification of agent identity and Authority-Scope is provided by the ATP Agent Certificate extension [ATP-CERT]. That extension binds Agent-ID, Principal-ID, and Authority-Scope to an X.509 v3 certificate presented during TLS mutual authentication, enabling infrastructure-layer identity and scope verification without application-layer access. Implementers planning deployments that require verified agent identity *SHOULD* plan for the Agent Certificate extension.

> Note: The Agent Certificate extension may be subject to pending intellectual property claims. See Section 7.7 and the IPR Notice preceding the Abstract for details. The licensor is prepared to grant a royalty-free license to implementers.

Every ATP server *MUST* log Agent-ID and Principal-ID fields for every request, creating an attributable audit trail even in deployments without the Certificate extension.

7.3. Authority Scope Enforcement

The Authority-Scope header declares what actions the agent is authorized to take. Compliant ATP servers *MUST* parse the Authority-Scope on every request, return 451 Scope Violation for any method that exceeds declared scope, and log all scope violations for audit purposes. Scope declarations are self-asserted in the base spec, analogous to scope assertions in OAuth 2.0 [RFC6749]. Cryptographically signed and infrastructure-enforced scopes are defined in [ATP-CERT].

7.4. Threat Model

7.4.1. Agent Spoofing

Threat: A malicious actor forges Agent-ID and Principal-ID headers to impersonate a trusted agent. Base spec mitigation: mandatory logging and anomaly detection. Full mitigation requires [ATP-CERT].

7.4.2. Authority Laundering

Threat: An agent claims an Authority-Scope broader than what it was granted. Mitigation: server-side scope enforcement; 451 Scope Violation returned and logged. In DELEGATE chains, each hop's scope *MUST* be a strict subset of the delegating agent's scope.

7.4.3. Delegation Chain Poisoning

Threat: A malicious agent inserts itself into a DELEGATE chain. Mitigation: Delegation-Chain headers are logged at each hop. 551 Authority Chain Broken is returned if any chain entry is unverifiable. Full mitigation requires [ATP-CERT] for signed delegation tokens.

7.4.4. Denial of Service via High-Frequency Agent Traffic

Threat: Agents — compromised, misconfigured, or adversarial — generate extremely high request volumes. Mitigation: 429 Rate Limited status code. Rate limiting *SHOULD* be applied per Agent-ID and per Principal-ID. When [ATP-CERT] is deployed, per-Agent-ID quotas can be cryptographically tied to verified identity, preventing quota evasion through Agent-ID spoofing.

7.4.5. Session Hijacking

Threat: An attacker intercepts or forges a Session-ID. Mitigation: mandatory TLS protects sessions in transit. Session-IDs *MUST* be cryptographically random with minimum 128 bits of entropy. Servers *MUST* validate that Session-ID, Agent-ID, and TLS client identity are consistent.

7.4.6. Escalation Suppression

Threat: A compromised agent or intermediary suppresses ESCALATE requests, preventing human oversight. Mitigation: compliant implementations *MUST* route ESCALATE requests directly to the declared escalation handler without modification. Intermediaries *MUST NOT* drop, delay, or modify ESCALATE requests. Escalation handlers *SHOULD* implement independent receipt confirmation.

7.5. Privacy Considerations

Agent identity headers carry information about agent behavior that may be sensitive:

* Agent-ID and Principal-ID together may reveal organizational structure

* Session-ID and Task-ID reveal workflow patterns

* Delegation-Chain reveals multi-agent architecture

ATP logs containing these fields *MUST* be treated as sensitive operational data. Operators *MUST* implement appropriate access controls, retention limits, and data minimization practices consistent with applicable privacy regulations.

Where privacy-preserving attribution is required, implementations *MAY* use pseudonymous Agent-IDs with a separate trusted resolution service. Under this model, the Agent-ID transmitted in ATP headers is an opaque, unlinkable identifier; a trusted resolution service — accessible only to authorized parties — maps the opaque identifier to the true agent identity for audit and accountability purposes. The architecture for pseudonymous agent identity resolution is reserved for a future companion document.

7.6. Denial-of-Service Considerations

ATP's agent identity headers provide a mechanism for more precise denial-of-service mitigation than is possible with HTTP. Rate limiting *SHOULD* be applied per Agent-ID and per Principal-ID in addition to per-IP-address controls.

When [ATP-CERT] is deployed, per-Agent-ID rate limiting can be cryptographically tied to verified agent identity, preventing quota evasion through Agent-ID rotation. Implementations planning high-volume governed agent deployments *SHOULD* plan for [ATP-CERT] as part of their denial-of-service mitigation strategy.

Additional recommended mitigations: Priority header enforcement (Priority: background requests *SHOULD* have lower rate limit headroom than Priority: critical); per-governance-zone aggregate limits in multi-tenant deployments; and circuit breaker patterns for ESCALATE request floods.

7.7. Intellectual Property Considerations

The core ATP specification — including all base methods, header fields, status codes, connection model, and IANA registrations defined in this document — is intended for open implementation without royalty obligation.

Certain extensions referenced in this document may be subject to pending patent applications by the author, specifically: the Agent Certificate extension [ATP-CERT], which provides cryptographic binding of agent identity and authority scope to ATP header fields;

and the ACTIVATE method, which provides ATP-native transmission and activation of governed agent packages. Implementers of the core ATP specification are not affected by any intellectual property claims on these extensions.

The licensor is prepared to grant a royalty-free license to implementers for any patent claims that cover contributions in this document and its referenced extensions, consistent with the IETF's IPR framework under [RFC8179].

IPR disclosures have been filed with the IETF Secretariat and are available at: https://datatracker.ietf.org/ipr/

8. IANA Considerations

This document requests the following IANA actions upon advancement to RFC status.

8.1. Port Assignment

Registration of the following service names in the IANA Service Name and Transport Protocol Port Number Registry:

Service Name	Port	Transport	Description
atp	TBD	TCP	Agent Transfer Protocol over TCP/TLS
atp-quic	TBD	UDP	Agent Transfer Protocol over QUIC

Table 17: Proposed Port Assignments

8.2. ATP Method Registry

Establishment of a new IANA registry: Agent Transfer Protocol Methods.

Registry name: Agent Transfer Protocol Methods

Registration procedure: Expert Review per [RFC8126], with the

additional requirement that each registration be accompanied by a published specification — at minimum a publicly available Internet-Draft or equivalent document. The Designated Expert *SHOULD* verify that the proposed method name is unique, the reference specification is publicly accessible, and the method definition includes the required fields (parameters, response structure, idempotency, error codes, security considerations).

Reference: This document

Initial registrations:

Method	Status	Reference
QUERY	Permanent	This document, Section 6.2
SUMMARIZE	Permanent	This document, Section 6.2
BOOK	Permanent	This document, Section 6.2
SCHEDULE	Permanent	This document, Section 6.2
LEARN	Permanent	This document, Section 6.2
DELEGATE	Permanent	This document, Section 6.2
COLLABORATE	Permanent	This document, Section 6.2
CONFIRM	Permanent	This document, Section 6.2
ESCALATE	Permanent	This document, Section 6.2
NOTIFY	Permanent	This document, Section 6.2

Table 18: Initial ATP Method Registry Entries

8.3. ATP Status Code Registry

Establishment of a new IANA registry: Agent Transfer Protocol Status Codes.

Registry name: Agent Transfer Protocol Status Codes

Registration procedure: Expert Review + published specification required.

The following ATP-specific status codes — those with no HTTP equivalent — are registered with full definitions:

Code	Name	Definition	Reference
451	Scope Violation	The requested action is outside the Authority-Scope declared in the request headers. The server *MUST* log this event. The agent *MUST NOT* retry the same request without modifying its Authority-Scope declaration. This is a governance signal, not a protocol error.	This document, Section 5.5
550	Delegation Failure	A sub-agent to which a task was delegated via the DELEGATE method failed to complete the task within the declared deadline or returned an error. The response body *SHOULD* contain the sub-agent's error details.	This document, Section 5.5
551	Authority Chain Broken	One or more entries in the Delegation-Chain header cannot be verified as part of a valid and continuous delegation sequence. The specific unverifiable entry *SHOULD* be identified in the response body. The server *MUST* log this event.	This document, Section 5.5

Table 19: ATP-Specific Status Code Definitions

8.4. Header Field Registry

ATP header fields are distinct from HTTP header fields and are registered in a new IANA registry: Agent Transfer Protocol Header Fields.

Registry name: Agent Transfer Protocol Header Fields

Registration procedure: Expert Review + published specification required.

ATP does not reuse the HTTP Field Name Registry, as ATP header fields have different semantics, applicability, and versioning constraints from HTTP fields. HTTP header fields are not automatically valid in ATP, and ATP header fields are not valid HTTP fields.

Initial registrations (all Permanent): ATP-Version, ATP-Method, ATP-Status, Agent-ID, Principal-ID, Authority-Scope, Session-ID, Task-ID, Delegation-Chain, Priority, TTL, Server-Agent-ID, Attribution-Record, Continuation-Token, Supported-Methods.

8.5. URI Scheme Registration

Registration of the atp:// URI scheme per [RFC7595], as described in Section 5.8.6 of this document.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8179] Bradner, S. and J. Contreras, "Intellectual Property Rights in IETF Technology", BCP 79, RFC 8179, DOI 10.17487/RFC8179, May 2017, <https://www.rfc-editor.org/rfc/rfc8179>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

9.2. Informative References

[A2A] Linux Foundation, "Agent-to-Agent Protocol Specification", 2025, <https://a2aprotocol.ai>.

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

[ANP] "Agent Network Protocol", 2025.

[ATP-CERT] Hood, C., "ATP Agent Certificate Extension", Work in Progress, Internet-Draft, draft-hood-atp-agent-cert-00, 2026, <https://datatracker.ietf.org/doc/html/draft-hood-atp-agent-cert-00>.

[ATP-METHODS] Hood, C., "ATP Standard Extended Method Vocabulary", Work in Progress, Internet-Draft, draft-hood-atp-standard-methods-00, 2026, <https://datatracker.ietf.org/doc/html/draft-hood-atp-standard-methods-00>.

[MCP] Anthropic, "Model Context Protocol", 2024, <https://modelcontextprotocol.io>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/rfc/rfc7540>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/rfc/rfc7595>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

Appendix A. Authority-Scope Format

Authority-Scope values are expressed as a space-separated list of scope tokens following the pattern: [domain]:[action] or [domain]:* for full domain access. Tokens *MUST* be lowercase ASCII with a single colon separator.

Examples:

```
Authority-Scope: calendar:book calendar:query
Authority-Scope: documents:summarize documents:query knowledge:learn
Authority-Scope: *:query
Authority-Scope: booking:* payments:confirm
```

Reserved domains (initial set):

Domain	Description
calendar	Scheduling and time-based resource management
documents	Document access, summarization, and annotation
knowledge	Agent context and memory operations
booking	Reservation and resource allocation
payments	Financial transactions and confirmations
agents	Delegation and collaboration with other agents
escalation	Escalation routing and handler management
activation	Governed agent package activation (ACTIVATE method extension)
*	All domains — requires explicit grant; use with caution

Table 20: Reserved Authority-Scope Domains

Appendix B. Example ATP Wire Formats

The following examples use a human-readable pseudo-wire format with HTTP-style headers followed by a JSON body. The Content-Type for all ATP message bodies is application/atp+json.

B.1. QUERY Request and Response

```
ATP/1.0 QUERY
Agent-ID: agt-7f3a9c2d
Principal-ID: usr-chris-hood
Authority-Scope: documents:query knowledge:query
Session-ID: sess-a1b2c3d4
Task-ID: task-0042
TTL: 3000
Content-Type: application/atp+json

{
  "task_id": "task-0042",
  "parameters": {
    "intent": "Key arguments against MCP in the context
             of HTTP overhead",
    "scope": ["documents:research", "knowledge:session"],
    "format": "structured",
    "confidence_threshold": 0.75
  }
}

ATP/1.0 200 OK
Task-ID: task-0042
Server-Agent-ID: srv-knowledge-01
Attribution-Record: [signed attribution token]
Content-Type: application/atp+json

{
  "status": 200,
  "task_id": "task-0042",
  "result": {
    "results": [{"content": "...", "source": "doc-atp-research",
                 "confidence": 0.91}],
    "result_count": 1
  }
}
```

B.2. BOOK Request and Response

```
ATP/1.0 BOOK
Agent-ID: agt-travel-planner
Principal-ID: usr-chris-hood
Authority-Scope: booking:* calendar:book
Session-ID: sess-trip-2026-04
Task-ID: task-0107
Priority: normal
Content-Type: application/atp+json

{
  "method": "BOOK",
  "task_id": "task-0107",
  "parameters": {
    "resource_id": "flight-AA2847",
    "principal_id": "usr-chris-hood",
    "time_slot": "2026-04-15T08:00:00Z",
    "options": {"seat_preference": "aisle", "class": "economy"},
    "confirm_immediately": true
  }
}

ATP/1.0 200 OK
Task-ID: task-0107
Attribution-Record: [signed attribution token]
Content-Type: application/atp+json

{
  "status": 200,
  "task_id": "task-0107",
  "result": {
    "booking_id": "BK-2026-0107",
    "status": "confirmed",
    "resource_id": "flight-AA2847",
    "confirmation_code": "XQRT7Y"
  }
}
```

B.3. ESCALATE Request and Response

```
ATP/1.0 ESCALATE
Agent-ID: agt-procurement-03
Principal-ID: usr-finance-dept
Authority-Scope: booking:* payments:confirm
Session-ID: sess-procurement-q2
Task-ID: task-0881
Priority: urgent
Content-Type: application/atp+json

{
  "method": "ESCALATE",
  "task_id": "task-0881",
  "parameters": {
    "task_id": "task-0880",
    "reason": "scope_limit",
    "context": {
      "attempted_action": "BOOK",
      "resource": "vendor-contract-750k",
      "block_reason": "Contract value exceeds agent
                  authorization threshold"
    },
    "recipient": "usr-cfo",
    "deadline": "2026-03-19T09:00:00Z"
  }
}

ATP/1.0 202 Accepted
Task-ID: task-0881
Server-Agent-ID: srv-escalation-handler
Content-Type: application/atp+json

{
  "status": 202,
  "task_id": "task-0881",
  "result": {
    "escalation_id": "ESC-0881",
    "routed_to": "usr-cfo",
    "status": "pending_review",
    "task_paused": true,
    "estimated_review_by": "2026-03-19T09:00:00Z"
  }
}
```

Appendix C. Comparison Table

Criterion	ATP	HTTP/ REST	gRPC	A2A / MCP
Agent-native methods	Yes	No	No	Partial
Intent semantics at protocol level	Native	None	None	Messaging layer only
Built-in agent identity	Yes	No	No	No
Authority scope enforcement	Protocol-level	None	None	Application-layer
Built-in attribution/ audit	Yes	No	No	Varies by impl.
Transport flexibility	TCP/UDP/QUIC	TCP/TLS	HTTP/2	HTTP
Escalation as first-class primitive	Yes	No	No	No
Ecosystem maturity	Proposed	Mature	Mature	Emerging
Governance/ observability	Native	Manual/ bolt-on	Manual	Limited
Method registry extensibility	Yes (Expert Review)	Frozen (IETF Review)	N/A	N/A
Open core / royalty-free	Yes	Yes	Yes	Yes

Table 21: ATP Compared to Existing Approaches

HTTP's method registry (registered with IETF Review per [RFC9110]) is effectively frozen for new semantic methods because any new HTTP method must be backward-compatible with existing HTTP infrastructure globally. ATP's Expert Review + published spec procedure enables the protocol to evolve its method vocabulary as the agent ecosystem develops, without the backward-compatibility constraints of the HTTP method space.

Appendix D. Glossary

Agent: A software system that executes tasks, makes decisions, and takes actions without continuous human supervision per transaction.

Agent Transfer Protocol (ATP): The application-layer protocol defined in this document, providing a dedicated transport environment for agent traffic.

Agent-ID: A unique identifier for a specific agent instance, present in all ATP request headers. In the base spec, self-asserted. With [ATP-CERT], cryptographically bound to a verified identity.

Attribution Record: A signed, logged record of an agent action, sufficient for audit and compliance purposes.

Authority-Scope: A declared set of permissions defining what actions an agent is authorized to take, expressed as space-separated domain:action tokens.

Delegation Chain: An ordered record of Agent-IDs representing the sequence of delegations that led to the current request.

ESCALATE: An ATP method representing an agent's intentional deferral of a decision or action to a human principal or higher-authority agent. A first-class method, not a failure code.

Intent Verb: An ATP method name expressing the agent's purpose, as distinguished from HTTP resource-operation verbs (GET, POST, PUT, DELETE).

Method Registry: The IANA-maintained registry of valid ATP method names and their specifications. Registration requires Expert Review and a published specification.

Principal: The human, organization, or system that authorized an agent to act and is accountable for its actions.

Principal-ID: The identifier of the principal on whose behalf an

agent operates, present in all ATP request headers.

Scope-Enforcement Point (SEP): An ATP-aware infrastructure component — load balancer, gateway, proxy — that enforces Authority-Scope compliance on ATP requests without application-layer access. Requires [ATP-CERT].

Scope Violation (451): An ATP status code returned when an agent requests an action outside its declared Authority-Scope. A governance signal, not a protocol error. *MUST* be logged.

Session: An ATP persistent connection context shared across multiple method invocations within a single agent workflow.

551 Authority Chain Broken: An ATP status code returned when one or more entries in the Delegation-Chain header cannot be verified as part of a valid and continuous delegation sequence. *MUST* be logged.

Author’s Address

Chris Hood
Nomotic, Inc.
Email: chris@nomotic.ai
URI: https://nomotic.ai