Network Working Group
Internet-Draft
Intended status: Informational
Expires: 14 November 2026

C. Rehfeld

13 May 2026

API Index (APIX): Core Infrastructure for Autonomous Agent Service Discovery
draft-rehfeld-apix-core-03

Abstract

The internet was designed for human actors. Its discovery infrastructure — search engines, directories, and hyperlinked documents — assumes a human reading and navigating. Autonomous agents (bots) operating on the internet today face a structural gap: there is no machine-native, globally accessible index of services they can consume.

This document defines the core infrastructure of the *API Index (APIX)*: a HATEOAS-based, globally accessible, commercially sustainable service discovery infrastructure designed for autonomous agents as its primary consumers. It specifies the governance model, the three-dimensional trust model, the APIX Manifest (APM) base format, commercial onboarding and sanctions compliance, the supply-side funding model, and the base Index API. These elements are shared across all APIX service types.

Profile documents extend this core for specific service categories: the APIX Services Profile (draft-rehfeld-apix-services-00) defines the web API and bot service profile; the APIX IoT Device Profile (draft-rehfeld-apix-iot-00) defines the IoT device profile.

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 14 November 2026.

Table of Contents

1. Introduction

1.1. The Bot Ecosystem Gap

The internet’s foundational infrastructure — HTTP, HTML, DNS, and search engines — was designed with human actors as the primary consumers. Web pages render visual layouts for human eyes. CAPTCHA systems explicitly discriminate against non-human access. Discovery mechanisms such as search engines index content for human-readable navigation.

Autonomous agents — software programs that independently execute tasks, consume APIs, and interact with external services without per-action human instruction — are not recognized as legitimate, first-class internet participants in this architecture. They are systematically treated as threats to be filtered, blocked, or rate-limited.

This situation is changing. The rapid growth of large language model-based agents, robotic process automation, and programmatic service consumers means that non-human actors now represent a significant and growing proportion of internet traffic. As this proportion increases, internet service providers will increasingly need to serve autonomous agents as a recognized user class alongside humans.

The API Index is premised on this trajectory: bots are becoming first-class internet participants, and the infrastructure to support them — starting with service discovery — does not yet exist. Regulators are converging on the same direction: the EU AI Act (Article 50) requires transparency and identity disclosure for AI systems that interact with people, and NIST’s Center for AI Standards and Innovation solicited public input on securing AI agent systems in early 2026. APIX’s verifiable trust model is designed to meet these emerging compliance requirements by construction.

1.1.1. Motivation: A Concrete Origin

The API Index was not conceived in the abstract. It emerged from a concrete practical failure.

A buying bot was built for a private use case: monitoring online shops for a specific product and purchasing it automatically the moment it became available. This is a straightforward task for an autonomous agent — exactly the kind of task agents are well-suited for.

The bot failed, not because the task was technically complex, but because the internet’s infrastructure is actively hostile to it:

HTML-only product pages. Product availability, price, and purchase state were encoded in HTML rendered for a human eye. No machine-readable API existed. The bot had to parse HTML — fragile, maintenance-intensive, and broken by every redesign.

Cloudflare Bot Management and equivalent shields. The majority of commercial web services now sit behind bot mitigation infrastructure. Cloudflare Bot Management, and equivalent products from Akamai, Imperva, and others, are deployed specifically to detect and block non-human request patterns. Repeated automated requests — even at modest frequency — trigger rate limiting, CAPTCHA challenges, or silent blocking. A buying bot that polls a product page to detect availability is treated identically to a malicious scraper or a DDoS participant.

CAPTCHA payment barriers. Even when product pages were reachable, payment flows required solving CAPTCHAs that explicitly excluded non-human actors. The purchasing step — the final, necessary action — was deliberately made inaccessible to the bot.

Proxy network pollution. To work around rate limits and bot detection, the bot required a rotating proxy network — different IP addresses on each request to disguise its automated origin. This is not a solution: it pollutes internet traffic with avoidable requests, raises the cost of operation, and contributes directly to the adversarial dynamic between bots and infrastructure operators. Every proxy request is a wasted roundtrip that a machine-readable API endpoint would have made unnecessary.

Polling as the only state-change mechanism. Because the bot had no way to subscribe to product availability events, it had to poll the product page continuously. This is architecturally wasteful: the bot consumes server resources and network bandwidth to repeatedly ask a question whose answer has not changed.

These are not edge cases. They are the standard experience for any autonomous agent attempting to consume a commercial internet service today. The buying bot illustrates why the API Index is necessary: not as an academic exercise, but as the infrastructure layer that makes autonomous agents functional participants in the commercial internet.

1.1.2. The Discovery Problem

When an autonomous agent must fulfill a task that requires an external service, it faces a fundamental discovery problem: how does it find services that can fulfill its requirement?

Current approaches are inadequate:

* *Hardcoded URLs*: brittle, require human maintenance, do not adapt to new or changed services.

* *LLM training data*: stale, non-authoritative, not machine-verifiable.

* *Human-curated lists*: do not scale, not machine-navigable, lack structured metadata.

* *Web search*: returns HTML documents designed for humans, not structured service descriptions for agents.

What is needed is a machine-native equivalent of a search engine: a global, always-current, structured index of services that autonomous agents can query by capability, trust level, liveness, and other machine-relevant criteria.

1.1.3. The Discovery Shift

Every automated system that calls an external service today does so because a human hardcoded that endpoint. The human is the discovery layer — the automation executes instructions, it does not find candidates independently.

APIX addresses this gap at infrastructure level: a globally queryable index of services that an agent can search by capability, trust level, and liveness — without prior human configuration of the specific endpoint. The agent discovers what exists; the human does not need to enumerate it in advance.

1.1.4. Infrastructure Efficiency and the Overhead of Human-Facing Responses

When an autonomous agent retrieves data from a web service today, it typically receives a response designed for a human browser: HTML markup, CSS stylesheets, JavaScript bundles, embedded fonts, advertising payloads, and analytics tracking instrumentation. The actual information content — an endpoint URL, a price, an availability flag — may occupy two kilobytes. The page weight delivering that content is routinely one to three megabytes.

This is a 500- to 1500-fold payload multiplier that carries no value for a machine consumer. It consumes bandwidth at the client, compute at the server, transit capacity on the network, and — at the scale of the growing autonomous agent population — represents a measurable and unnecessary energy expenditure.

Machine-native APIs eliminate this overhead entirely. A structured JSON response delivers exactly the information the agent requested and nothing else. The IETF Datatracker provides a concrete illustration: the human-facing document page for an Internet-Draft loads several hundred kilobytes of rendered HTML and supporting assets; the equivalent information retrieved via the Datatracker REST API returns in under one kilobyte of JSON. The data is identical. The difference is entirely overhead serving a human rendering pipeline that a machine does not have.

APIX addresses both the discovery gap and this efficiency gap together. By providing infrastructure that indexes machine-native service endpoints, APIX encourages Service Owners to expose structured, agent-consumable APIs alongside or in place of human-facing interfaces. The aggregate effect, as autonomous agent workloads scale, is a reduction in the payload overhead carried by bot traffic across the internet as a whole. This is an explicit co-mission of APIX: machine-native infrastructure is not only more functional for agents — it is more efficient for the internet, and helps reduce humanity’s environmental footprint as much as possible.

1.1.5. Lessons from Prior Art

The APIX is not the first attempt at a global service registry. Prior efforts must be understood explicitly so that their failure modes are not repeated.

UDDI (Universal Description, Discovery and Integration) UDDI was a SOAP-era standard for a global service registry with the same conceptual goal as APIX, published as an OASIS Committee Draft in October 2004. It failed for three reasons: (1) extreme complexity of the XML-based data model; (2) no automatic verification — all data was self-asserted with no crawling or validation; (3) no adoption incentive — there was no commercial model to sustain registration or discovery. APIX addresses all three directly: a simple JSON manifest, automated spider verification, and a commercial tier model.

robots.txt (Robots Exclusion Protocol) Machine-readable, but concerned with exclusion — telling crawlers what not to access — not with discovery of capabilities. Per-domain only. Not a registry.

MCP (Model Context Protocol) Defines tool and capability descriptions for LLM-based agents. Excellent for consumption once a server URL is known. Does not address the discovery problem: there is no index of MCP servers. APIX is complementary to MCP — it can index MCP servers as one supported spec type. As of December 2025, MCP is governed by the Linux Foundation Agentic AI Foundation ([AAIF]), under a vendor-neutral SEP (Specification Enhancement

Proposal) process that explicitly prevents single-company control — a governance philosophy that directly aligns with APIX's own neutrality requirements.

Well-Known URIs (RFC 8615) Per-domain machine-readable metadata at /.well-known/. Useful for per-service metadata but requires the consumer to already know the domain. No cross-service search or global index.

DNS DNS resolves names to addresses but carries no capability semantics. It is an architectural analogy for APIX's federation model, not a comparable system.

1.1.6. Related IETF and W3C Work

As of April 2026, the number of Internet-Drafts working in adjacent areas of agent/bot infrastructure has grown significantly. None addresses the same problem as APIX. This section documents each and states the relationship explicitly.

draft-pioli-agent-discovery (ARDP) Proposes a federated agent registration and discovery protocol. Deliberately decentralised — no global registry mandate, no central query URL. Relationship to APIX: complementary. ARDP addresses agent-to-agent capability advertisement within a federation. APIX addresses global, cross-organisation service discovery from a neutral central index. ARDP's JWS-based signing of registration payloads provides cryptographic non-repudiation of the manifest content — a property APIX currently achieves through layered governance verification (DNS ownership proof at O-1, Spider crawl, KYC pipeline). APM manifest-level signing is a candidate extension for a future APIX revision, and ARDP's signing model is the reference design for that work.

draft-narajala-courtney-ansv2 (ANS v2) Anchors autonomous agent identities to DNS domain names with Registration Authority verification. Focused on agent identity and trust anchoring, not service capability discovery. ANS v2 builds on a peer-reviewed predecessor published at IEEE ICAIC 2026, simplifying the name format to three components (ans://v{version}.{agentHost}), introducing a dual-certificate model, and replacing conceptual registry integrity with a cryptographic Transparency Log. ANS v2 explicitly identifies the limitation of DNS-SD ([RFC6763]): DNS-SD adds service discovery but cannot tell a client whether the agent at an address is the one it claims to be. ANS v2 fills that identity gap. Relationship to APIX: complementary. DNS-SD locates a service; ANS v2 verifies the identity of the agent at that address; APIX provides capability search and multi-dimensional trust metadata across organisations. ANS v2 could serve as the identity layer for service operators registered in APIX.

draft-vandemeent-ains-discovery (AINS) Agent discovery via signed, append-only replication logs. No central authority. No commercial sustainability model. Relationship to APIX: different philosophy. AINS prioritises decentralisation and cryptographic verifiability. APIX prioritises a single authoritative global index with a governed trust model.

AINS defines a multi-channel verification model in which each verified channel produces an independent evidence object. The principle is sound: independent signals from multiple channels produce stronger identity assurance than any single channel alone. AINS names DNS, HTTPS, and email as verification channels — all of which are compatible with APIX's own trust evidence model (DNS TXT record at O-1, HTTPS-reachable manifest verified by the APIX Spider). AINS additionally names source code repositories (e.g., GitHub) as a verification channel. APIX does not adopt repository access as an evidence channel. For open-source projects and developer platforms this channel is accessible and useful; however, the majority of enterprise API services — financial institutions, healthcare providers, manufacturers, and proprietary IoT backends — maintain private repositories as protected intellectual property, often under regulatory or contractual obligations that prohibit external access. APIX's governance-based evidence channels (DNS, legal entity registration, commercial contract, third-party audit) apply universally regardless of whether a registrant's codebase is open-source or proprietary, and this universality is a deliberate scope decision.

draft-aiendpoint-ai-discovery (AI Discovery Endpoint) Defines /.well-known/ai as a per-host machine-readable capability document. Per-domain only; not a global index. Relationship to APIX: directly

complementary. The APIX Spider SHOULD read /.well-known/ai when present on a registered service's domain as an additional source of capability metadata.

This draft defines a flat category taxonomy for service classification: "productivity", "ecommerce", "finance", "news", "weather", "maps", "search", "data", "communication", "calendar", "storage", "media", "health", "education", "travel", "food", "government", "developer". The convergence with APIX's capability taxonomy is notable: search, communication, storage, and media appear in both; ecommerce and finance correspond directly to APIX's commerce and data.financial terms. The two taxonomies differ in architecture — AI Discovery Endpoint uses flat single-word labels optimised for human-readable classification; APIX uses hierarchical dot-separated terms (commerce.marketplace, data.financial) optimised for machine-queryable precision — but the independent convergence on the same fundamental service categories validates both approaches. Categories present in AI Discovery Endpoint but not yet in APIX's v1.0 starter set (health, education, government, travel, food, news, weather, maps, developer) are candidates for future additions through the governing body's capability taxonomy governance process ([APIX-SERVICES]).

draft-batum-aidre (AIDRE) Defines /.well-known/ai-discovery as a per-origin discovery document. Decentralised by design. Relationship to APIX: complementary. APIX provides the global aggregation and trust verification layer that per-origin endpoints cannot provide alone.

draft-cui-ai-agent-discovery-invocation Specifies a metadata format for agent capabilities and a registry-based discovery mechanism. Explicitly permits multiple coexisting registries; no global authority defined.

This draft introduces a notable split between two metadata fields: capabilities (high-level descriptors of what the service does, e.g., ["translation", "summarization"]) and tags (broader, orthogonal properties such as domain, language support, or deployment model, e.g., ["nlp", "chinese", "transformer_model", "cloud"]). The split recognises that some service properties are functional capabilities while others are orthogonal classifiers that do not fit a strict capability hierarchy.

APIX takes a different approach. The hierarchical dot-separated capability taxonomy (nlp.translation, commerce.marketplace) encodes both the category and the specific capability in a single governed term, enabling prefix-based machine queries (nlp.*) and registry-controlled vocabulary. Orthogonal dimensions that draft-cui

expresses as free-form tags are handled in APIX through dedicated typed fields: language (BCP 47, [RFC5646]) covers language support; deployment model is not yet represented and is noted as a potential future gap. The APIX design trades the flexibility of a free-form tag bag for machine-queryability and governance — a tag field without a registry becomes a folksonomy that degrades search precision at scale. An empirical basis for preferring intent-aligned capability descriptors over opaque operation labels is provided by the controlled benchmark study in [I-D.hood-agtp-api], which demonstrates that intent-aligned names produce materially higher endpoint selection accuracy in frontier-class language models, with the accuracy gain attributable to the name itself independent of additional documentation.

This draft also identifies pricing information as a legitimate service metadata concern — noting that if a service charges per use, agents need this information at discovery time. The draft does not standardise a pricing schema ("not standardized here but can be included as needed"). APIX adopts this observation and formalises it: the pricing field in the APM schema ([APIX-SERVICES]) defines a governed model enum (free, freemium, paid, enterprise, dynamic) and a pricing_endpoint for real-time load-based price queries. The index stores only the declared model and the endpoint reference; consuming agents are responsible for querying the pricing_endpoint directly to obtain and evaluate the current price before invocation.

This draft also defines a Semantic Routing Platform (SRP): an optional control-plane service that performs semantic matching, ranking, and policy-based filtering of candidate agents before invocation, without participating in task execution. The SRP pattern assumes a structured candidate pool as its input. APIX is the natural data source for that pool: an SRP would query APIX with structured filters to retrieve a trusted, governed candidate set, then apply semantic ranking over that set before presenting the shortlist to the invoking agent. The two layers are complementary — APIX provides structured discovery and trust metadata; the SRP provides semantic selection above that foundation.

Relationship to APIX: partially overlapping problem space. The capability/tag split, the pricing observation, and the SRP pattern are all concrete design contributions; APIX’s governed taxonomy, typed fields, and formalised pricing schema address the same concerns through a more structured mechanism, and the SRP architecture positions APIX as the structured input layer to semantic selection rather than as a competitor to it.

draft-am-layered-ai-discovery-architecture Proposes a conceptual two-layer architecture separating a Discovery Transport Layer (DTL) from the metadata format carried over it. The DTL is explicitly abstract: the draft names HTTP, pub/sub, multicast, and MoQ as candidate substrates without specifying any of them normatively. No wire format, no concrete protocol mechanisms, and no IANA actions are defined.

APIX resolves the transport question concretely and normatively: HTTPS with TLS ([RFC8446]), JSON ([RFC8259]), and HATEOAS navigation over a single stable entry point. This is a deliberate design position in favour of implementability over substrate generality. Adding a DTL abstraction layer atop APIX's concrete HTTP interface would introduce indirection without communicative or interoperability benefit — the transport is already specified, and no agent implementation benefits from treating it as one option among many.

Directly relevant to APIX is the draft's categorisation of discoverable object types (agents, models, data resources, robots), which recognises that different object categories require different metadata profiles. This independently converges on the same architectural reasoning behind APIX's decision to separate the Services Profile ([APIX-SERVICES]) from the IoT Device Profile ([APIX-IOT]) rather than collapsing all service types into a single flat schema.

Relationship to APIX: categorisation framing is consistent with the APIX profile split; the abstract DTL layer is not adopted.

AGTP Protocol Family

The Agent Transfer Protocol (AGTP) defines a dedicated agent-native protocol substrate, distinct from HTTP, with an IANA-registered URI scheme (agtp://) and port 4480, media types in expert review, and live reference servers at agtp://agents.agtp.io. The AGTP family currently comprises four drafts.

[I-D.hood-independent-agtp] is the core transport substrate. The defining architectural commitment of the family is that agent-native APIs operate on AGTP rather than HTTP.

[I-D.hood-agtp-discovery] defines an Agent Name Service (ANS) — a governed registry that resolves capability queries into ranked lists of Agent Manifest Documents for authenticated agents. ANS servers act as Scope-Enforcement Points, applying trust score thresholds, trust tier requirements, and governance zone constraints. Cross-organisational discovery is supported through peer ANS server federation.

[I-D.hood-agtp-api] defines the Agentic API contract layer: a curated method catalog of intent-aligned verbs (QUERY, EXECUTE, PROPOSE, DISCOVER, and eight additional methods), endpoint primitives carrying semantic contracts, path grammar rules, and schema validation. The draft introduces a runtime contract negotiation mechanism via the PROPOSE method: a consuming agent may propose an endpoint that does not exist, and the serving system synthesises it from its existing capabilities at session scope. The intent-aligned method vocabulary is grounded in a controlled empirical benchmark across four frontier-class model families showing that intent-aligned verbs produce materially higher endpoint selection accuracy than CRUD verbs, with description-swap ablations confirming that the accuracy gain is attributable to the method name itself independent of documentation quality.

[I-D.hood-agtp-trust] defines a three-tier verification model with three independent Tier 1 verification paths (DNS-anchored per RFC 8555, log-anchored per RFC 9162, and SCITT per RFC 9943), hybrid trust composition, and a normative 0.0-1.0 continuous trust score with freshness semantics that are operation-class-dependent.

Relationship to APIX: overlapping problem space, fundamentally different architectural commitment. The AGTP family’s defining premise is that agent-native services should operate on a dedicated off-HTTP protocol substrate. APIX’s defining premise is that the discovery layer should operate over existing HTTP infrastructure with zero adoption friction: any service already reachable over HTTP registers in APIX without changing its underlying protocol. These are not competing answers to the same deployment question; they address different positions in the adoption spectrum. AGTP targets greenfield services designed for agent-native operation from scratch; APIX targets the full landscape including existing HTTP/REST APIs, MCP-served models, IoT backends, and enterprise systems that will not migrate off HTTP for operational, legal, or contractual reasons.

Three specific alignments are worth noting. First, the AGTP trust tier evidence paths (DNS per RFC 8555, transparency log per RFC 9162, SCITT per RFC 9943) are structurally analogous to APIX’s O-level evidence channels (DNS TXT record at O-1, GLEIF LEI database at O-2, independent audit at O-5); a shared trust evidence vocabulary between the two specifications would benefit consuming agents that interact with both. Second, the AGTP PROPOSE method — server-side synthesis of non-existent endpoints from existing capabilities at session scope — has no current analogue in APIX and is identified as a candidate area for future dynamic capability negotiation. Third, the empirical finding on intent-aligned method names in [I-D.hood-agtp-api] provides an independent quantitative basis for APIX’s capability taxonomy design: APIX capability terms (nlp.translation,

commerce.marketplace) are intent-aligned descriptors rather than CRUD-style operation labels, and the benchmark result supports that design choice.

draft-mozley-aidiscovery (AI Agent Discovery Problem Statement) Argues for a distributed, organisation-centric discovery model in which each organisation independently publishes agent capabilities at a well-known entry point. The draft explicitly opposes centralised registries on two grounds: single points of failure limiting resilience, and the competitive harm risk — stated directly as: "An adversarial centralized directory is also able to stifle competitor advertisement capabilities." The scope is cross-organisational; the draft addresses public, multi-domain agent discovery, not only local or intra-organisation scenarios.

Relationship to APIX: this draft articulates the strongest counter-position to APIX's architecture, and the adversarial directory argument deserves a direct response. APIX addresses it structurally: the neutrality requirements (Section 4.2), the prohibition on ranking preferences and preferential treatment, the independent governance of the standard from the commercial operation, and the mandatory open bulk data download are specifically designed to make the adversarial scenario impossible by construction. A directory operated under these constraints cannot stifle competitor advertisement because it cannot discriminate between registrants at the same commercial tier.

The distributed model's remaining gap, which APIX addresses, is the zero-prior-knowledge case: an agent that has no prior relationship with any service provider needs a single starting point from which to discover unknown third parties. An organisation-centric model requires the discovering agent to already know which organisations to query — which presupposes the discovery problem is already solved.

draft-mozleywilliams-dnsop-dnsaid (DNS for AI Discovery) Proposes DNS-AID: using SVCB records to publish agent service endpoints. Relationship to APIX: complementary at the infrastructure layer. The distinction across the three systems is precise: DNS-AID tells a client where to connect; ANS v2 ([I-D.narajala-courtney-ansv2]) tells it whether to trust the agent at that address; APIX tells it what to connect to and why — capability search, multi-dimensional trust metadata, and liveness verification across the global service landscape.

draft-meunier-webbotauth-registry (webbotauth) Defines a JSON-based "Signature Agent Card" format for bot authentication. Focused on bot identity — how a bot proves who it is to a service. Related to the active webbotauth IETF Working Group. Relationship to APIX: orthogonal but complementary. webbotauth addresses bot consumer identity; APIX addresses service provider discovery.

I-D.ietf-scitt-architecture (SCITT) Defines an append-only transparency service for supply chain integrity, transparency, and trust. An IETF WG specification ([I-D.ietf-scitt-architecture]). SCITT provides a tamper-evident, auditable ledger model where statements about artefacts are registered and independently verifiable. Relationship to APIX: architectural reference. APIX's audit trail for organisation trust level progressions, LER submissions ([APIX-IOT]), and sanctions screening events follows the same append-only, non-repudiable model that SCITT formalises. ANS v2 ([I-D.narajala-courtney-ansv2]) bases its Transparency Log on SCITT. A future revision of APIX MAY adopt SCITT-compliant transparency log semantics for its governance audit trail.

Google Cloud Fraud Defense A commercial trust platform for the agentic web announced at Google Cloud Next '26 (April 2026), positioned as the next evolution of reCAPTCHA. Fraud Defense explicitly integrates with the webbotauth IETF Working Group and SPIFFE for agent and workload identity classification. Relationship to APIX: complementary at adjacent layers. Fraud Defense operates at the consumption layer — it verifies and classifies agent traffic arriving at a service endpoint. APIX operates at the discovery layer — it provides the service index, trust metadata, and capability taxonomy that agents use to locate services before interacting with them. The two systems are not competitive; a Fraud Defense policy engine can consume APIX trust signals (O-level, S-level) as inputs to its risk scoring.

SPIFFE (Secure Production Identity Framework For Everyone) A CNCF open standard for workload identity attestation. Provides cryptographically verifiable identities (SVIDs) to software workloads in dynamic infrastructure. Referenced as an integration target by Google Cloud Fraud Defense alongside webbotauth. Relationship to APIX: complementary at the identity layer. SPIFFE addresses machine/workload identity; APIX addresses service and device discovery with human-governed trust levels. A SPIFFE SVID could serve as a technical credential for an agent whose operator is registered in APIX at O-2 or above.

W3C AI Agent Protocol Community Group Proposed May 2025, targeting agent interoperability protocols. Pre-specification as of this writing. Relationship to APIX: APIX will monitor this group's outputs and align the APM capability taxonomy with any vocabulary standardised by the W3C CG where applicable.

Agent2Agent Protocol (A2A) Defines a secure communication protocol for agent-to-agent interaction across frameworks [A2A]. Originated at Google (April 2025), transferred to the Linux Foundation Agentic AI Foundation ([AAIF]) in June 2025; as of early 2026 it has 150+ supporting organisations and is in production use. Relationship to APIX: directly complementary. A2A addresses how agents communicate once they have located each other. APIX addresses how agents locate each other in the first place. An agent that uses APIX for discovery and A2A for subsequent communication is using both systems for their intended purpose with no overlap.

AGNTCY (Open Agent Schema Framework) A multi-component open infrastructure project for multi-agent systems [AGNTCY], originating at Cisco and transferred to the Linux Foundation ([AAIF]) in July 2025. As of early 2026 it has 65+ supporting organisations and is in production use for CI/CD, IT automation, and telecommunications. AGNTCY comprises four components: the Open Agent Schema Framework (OASF) for capability discovery, cryptographic identity, SLIM messaging, and end-to-end observability. AGNTCY is governed under the Linux Foundation AAIF mandate of no single-company control.

Relationship to APIX: the governance philosophies are aligned; the architectural scope is different. OASF defines a capability schema format — analogous to OpenAPI for agent capabilities — for registering and advertising what an agent can do. APIX is a globally queryable index infrastructure: a single authoritative entry point where agents discover unknown third-party services by capability, with commercial sustainability, verified trust metadata, and structured search. OASF and APIX are complementary: OASF provides the schema language; APIX provides the global index that can be populated with OASF-described services. An AGNTCY-registered agent is a candidate APIX registrant. The principal architectural difference is scope: AGNTCY is optimised for intra-platform and intra-organisation agent coordination; APIX is designed for cross-organisation, cross-border, zero-prior-knowledge discovery of agent-consumable services and IoT device classes. The two systems address different points in the discovery spectrum and are not substitutes for each other.

draft-drake-agent-identity-registry (Agent Identity Registry) Defines a federated registry architecture for persistent, hardware-anchored agent identities. Introduces a three-tier model: Agent

Identity Authority (AIA) as a governance body, Registry Operators as authoritative identity databases, and Registrars for hardware attestation and OIDC token issuance. The AIA is explicitly required to be constituted as a multi-stakeholder body — the draft states directly that "single-entity control would undermine the federated design" ([I-D.drake-agent-identity-registry]).

Relationship to APIX: this draft provides the strongest independent validation of APIX's core governance premise. Two separate specifications, developed independently, arrive at the same structural requirement: that foundational agent infrastructure must be governed by a multi-stakeholder body, not controlled by a single entity. The functional domains are complementary rather than overlapping — draft-drake addresses agent identity (who is this agent, which hardware backs its credential); APIX addresses service discovery (what services exist, what can they do, are they trustworthy). An agent whose identity is established under draft-drake's AIA model is a well-suited candidate to consume and register services in APIX.

Linux Foundation Agentic AI Foundation (AAIF) Formed December 2025 with founding contributions from Anthropic (MCP), OpenAI (AGENTS.md), and Block (goose); additional members include AWS, Bloomberg, Cloudflare, Google, Cisco, Dell, Oracle, and Red Hat. The AAIF's explicit founding mandate is to ensure "no single company controls the direction of foundational infrastructure" ([AAIF]), implemented through a vendor-neutral directed fund structure and per-project Specification Enhancement Proposal (SEP) processes modelled on Kubernetes's KEP governance.

Relationship to APIX: the AAIF's governance mandate independently validates APIX's constitutional neutrality requirements. APIX predates the AAIF as an IETF submission and implements the same principle — no single commercial interest may control the standard or its operation — through a different structural mechanism: a Swiss Stiftung with a supply-side commercial model that funds operations without creating discovery-layer incentives to favour any registrant. The AAIF governs communication and invocation protocols (MCP, A2A); APIX governs the discovery index. These are adjacent, non-overlapping layers of the same infrastructure stack.

Positioning The agent infrastructure space has consolidated significantly in 2025-2026. At the protocol layer, the Linux Foundation AAIF has emerged as the primary governance body for communication and invocation standards (MCP, A2A), with 150+ supporting organisations and active production deployment. At the IETF, over a dozen individual drafts address agent discovery and identity from different architectural starting points; none has reached Working Group consensus.

APIX occupies a distinct position in this landscape: it is the only specification in the IETF space that makes governance the primary architectural requirement, and the only proposal for a globally queryable, commercially sustainable, neutral discovery index. The dominant IETF tendency toward decentralisation addresses legitimate concerns about single points of control; APIX answers those concerns structurally, through its neutrality mandates, open bulk data requirements, and separation of standard governance from commercial operation — rather than by abandoning the global index model that those concerns are directed at.

APIX is designed to compose with, not replace, the adjacent standards: APIX provides the discovery layer that MCP, A2A, and AGNTCY do not provide; draft-drake provides the identity layer that APIX delegates to external identity infrastructure; the webbotauth Working Group provides the bot authentication layer that APIX references as a trust signal. Each standard goes deep in its own sub-problem; APIX depends on that depth rather than duplicating it.

The AGTP protocol family represents a distinct architectural trajectory: a dedicated agent-native transport substrate (agtp://) that replaces HTTP rather than extending it. APIX and AGTP are not substitutes and the distinction is one of adoption scope, not superiority. AGTP is the invocation substrate for greenfield services designed from scratch; APIX is the discovery index for the full existing service landscape, including the large majority of deployable services that will not migrate off HTTP in any planning horizon relevant to agent infrastructure standardisation.

1.2. 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 [RFC2119].

All API responses MUST be encoded as UTF-8 as mandated by [RFC8259] Section 8.1. All string fields in APM documents and Index API responses MUST contain valid UTF-8. HTTP status codes used throughout this specification are defined in [RFC9110].

Agent An autonomous software program that executes complex, goal-directed tasks by consuming external services, without per-action human instruction. Agents may use LLM-backed or programmatic orchestration logic. The primary consumer class targeted by the APIX Index API.

Bot An autonomous software program that executes deterministic, rule-based internet tasks: web crawling, API polling, automated messaging, without per-action human instruction. Behavior is scripted rather than goal-directed. The APIX Spider is itself a bot.

Connected Device A physical or embedded hardware unit with network connectivity that exposes services or sensor data via the APIX Presence Protocol. Registered as a Device Class and tracked as a Device Instance as defined in [APIX-IOT]. Distinct from Agent and Bot in that the principal is hardware, not software.

Service A machine-consumable API or connected device class offered by an organisation, registered in the APIX, and described by an APIX Manifest. The term covers both web API services ([APIX-SERVICES]) and IoT device services ([APIX-IOT]).

Service Owner The organisation responsible for registering, maintaining, and operating a Service in the APIX.

APIX Manifest (APM) The structured metadata document that describes a Service to the APIX, including its technical specification reference, capability taxonomy, trust metadata, and commercial terms. Profile documents define the additional fields applicable to each service type.

Governing Body The neutral, non-profit entity that operates the APIX, maintains its registries, accredits Regional Representatives and Verifiers, and ensures the governance and operational requirements defined in this specification are met. Any entity that satisfies those requirements MAY fulfil this role.

API Index (APIX) The global, centralised index of registered Services, operated by the governing body and queryable by autonomous agents via the Index API.

Index API The HATEOAS-compliant HTTP API exposed by the APIX for agent discovery and navigation.

Accredited Verifier A trusted third-party organisation, accredited by the governing body, that performs human-intensive trust verification at Organisation levels O-4 and O-5.

Accredited Regional Representative An organisation accredited by the governing body to operate commercial onboarding, contracting, and customer relationships within a defined geographic jurisdiction, under the governing body’s standard and governance.

Trust Policy A set of minimum trust requirements expressed by a consuming agent that a Service must satisfy before the agent will use it.

Liveness The confirmed operational status and response availability of a Service, as measured by automated means at a frequency determined by the Service’s commercial tier. The specific liveness mechanism differs by service type: Spider health checks for web API services; presence signals for IoT device services.

Tier A commercial subscription level that determines a Service’s visibility in the APIX, liveness check frequency, and API query rate allocation.

1.3. Design Goals

1.3.1. Requirements (MUST)

* The APIX MUST be queryable by autonomous agents via a stable, globally accessible URL without prior knowledge of any specific service.

* The Index API MUST follow HATEOAS principles: agents MUST be able to navigate the full index starting from a single entry-point URL.

* Every Service record MUST expose machine-readable trust metadata across all three trust dimensions (Organisation, Service, Liveness).

* Service registration MUST be human-initiated. The registrant MUST agree to the index operator’s Terms of Service before any service record is activated. For O-0 and O-1, self-service portal registration with accepted Terms of Service satisfies this requirement. For O-2 and above, registration MUST additionally involve a formal B2B contractual relationship between the Service Owner and the index operator or its Accredited Regional Representative.

* The APIX MUST expose trust metadata as verifiable facts, not as recommendations. Trust decisions MUST remain with the consuming agent.

* The APIX Manifest (APM) MUST be format-agnostic: it MUST support referencing multiple service types via an extensible type registry.

* The APIX MUST be operated as a neutral, non-profit infrastructure under the governance of the governing body.

1.3.2. Goals (SHOULD)

* The Index API SHOULD support full-text and structured search by capability, category, organisation trust level, service verification level, liveness freshness, and protocol type.

* The APIX SHOULD provide SDKs in common agent development languages to lower the integration barrier for consuming agents.

* The APIX SHOULD support a federated accredited verifier model so that Organisation trust levels O-4 and O-5 can be verified at scale without centralising all human review in the governing body.

* Accredited Regional Representatives SHOULD be established in major jurisdictions to allow Service Owners to contract in their local language and legal framework.

* The APIX SHOULD publish a public transparency report at least annually, disclosing the number of registered services by tier and trust level, coverage statistics, and verifier accreditation status.

* The APIX SHOULD, through its verification model and tier structure, incentivise Service Owners to expose structured, machine-consumable API endpoints rather than requiring agents to adapt to human-facing HTML interfaces. Eliminating rendering, styling, and advertising overhead from machine-to-machine communication is an explicit efficiency objective of this infrastructure.

1.3.3. Out of Scope

The following are explicitly not addressed by this document. Items marked MUST NOT are normative constraints on conforming implementations; remaining items are editorial scope boundaries.

* *Bot identity and authentication*: how a bot proves its own identity to a service it consumes. This is addressed by complementary work such as draft-meunier-webbotauth-registry. This document takes no position on bot identity mechanisms.

* *Bot rights and legal personhood*: outside the scope of a technical infrastructure standard.

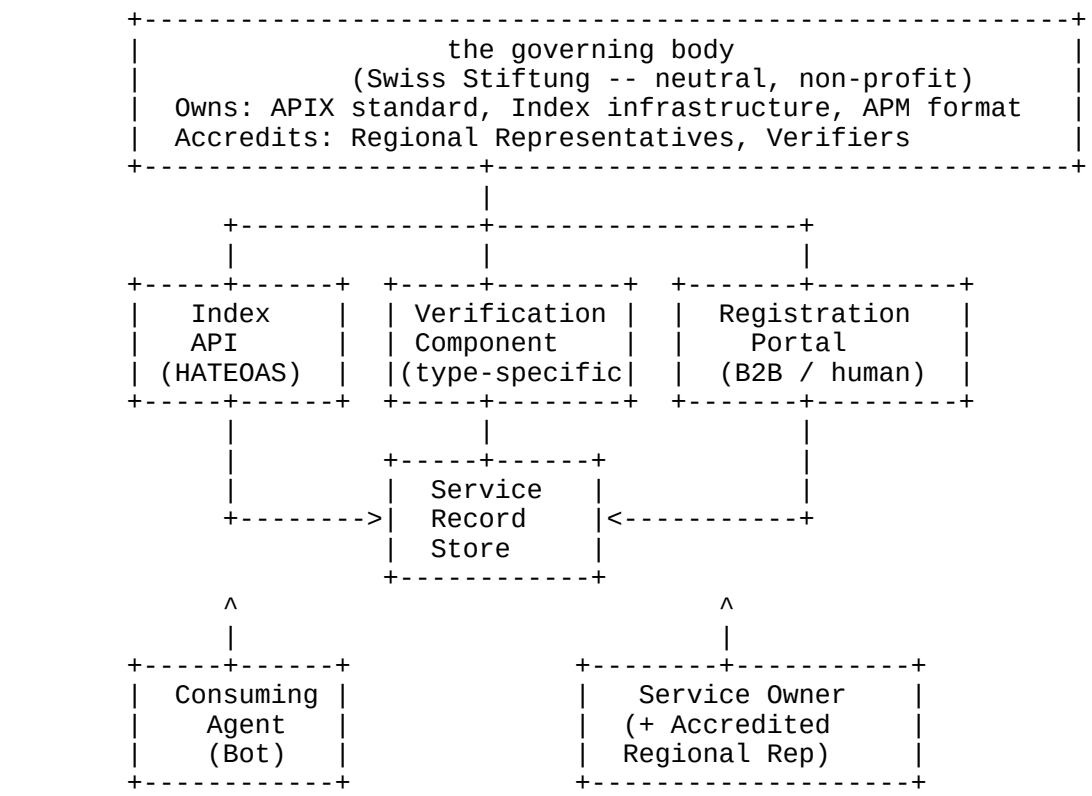
* *Service execution*: a conforming APIX implementation MUST NOT proxy, mediate, or execute service calls on behalf of consuming agents. The APIX is a discovery layer only; all service interactions occur directly between the consuming agent and the Service Owner’s infrastructure.

* *Content indexing*: a conforming APIX implementation MUST NOT index service response content. The APIX indexes service metadata — capability declarations, trust levels, liveness signals — not the data that services return when called.

* *Mandatory consumer registration*: a conforming APIX implementation MUST NOT require consuming agents to register or identify themselves as a condition of performing discovery queries (see Section 9.2).

1.4. Architecture Overview

1.4.1. Component Model



This document uses the generic terms "governing body" and "index operator" in all normative requirements. These terms are intentionally role-based: any entity that satisfies the governance, neutrality, and operational requirements defined in this specification MAY fulfil them. The reference implementation of these roles is described in the non-normative appendix "Reference Implementation" at the end of this document.

Flow:

1. A Service Owner (or their Accredited Regional Representative) creates an Organisation Account in the APIX Registration Portal, providing company details and agreeing to a commercial contract.

2. The Registration Portal creates a draft Service Record and triggers profile-appropriate verification (Spider crawl for web API services; manufacturer provisioning for IoT device classes).

3. The verification component updates the Service Record with verified technical metadata.

4. The Service Record becomes queryable via the Index API.

5. A consuming agent queries the Index API from the single entry-point URL, navigates by HATEOAS links, applies its Trust Policy, and selects services that satisfy its requirements.

6. Verification rechecks services on the schedule defined by each service's liveness monitoring configuration.

1.4.2. Governance Model

The APIX MUST be operated by a *neutral governing body* that satisfies the following normative requirements. These requirements apply to any conforming APIX implementation; the specific legal form of the governing body is an implementation choice.

Neutrality requirements:

* The governing body MUST have no commercial interest in preferring any registrant's services over another in index results or liveness scheduling.

* The governing body MUST NOT offer exclusive discovery advantages, ranking preferences, or prioritised verification treatment to any registrant regardless of commercial relationship.

* Governance of the APIX standard and APM specification MUST be separated from operation of the commercial index. A single entity may not simultaneously control standard evolution and derive commercial benefit from preferential application of that standard.

Operational requirements:

* The governing body MUST accredit Regional Representatives who may handle service onboarding in specific jurisdictions. Regional Representatives operate under licence from the governing body; the index itself remains a single global store.

* The governing body MUST accredit Verifiers who perform Organisation trust assessments at O-4 and O-5. Accredited Verifiers are structurally analogous to Certificate Authorities in the TLS ecosystem.

* The governing body MUST maintain the capability taxonomy and publish all versions of the APM specification and Index API specification as open standards under a permissive licence.

* The governing body MUST perform sanctions screening on service registrants (see Section 8).

Openness requirements:

* The full index MUST be made available as a freely downloadable bulk dataset on the first day of each calendar month, under the Open Database Licence (ODbL) 1.0. No entity, including the governing body, may hold an exclusive lock on the index data.

* Incremental diff files MUST be published daily, each covering all record additions, updates, and deletions since the previous day's snapshot. A downstream consumer MUST be able to reach current index state by applying the monthly full snapshot and the sequence of daily diffs since that snapshot, without downloading any additional full snapshots.

* Discovery queries to the Index API MUST be available without authentication or payment (subject to rate limits; see Section 9.2).

1.4.2.1. Global Participation

A conforming APIX implementation SHOULD establish mechanisms to ensure global representation in the capability taxonomy, including service categories relevant to underrepresented regions. Where regional institutional partners are willing to co-sponsor regional participation, the governing body SHOULD establish formal co-sponsorship relationships and associated governance representation for those regions.

Regional verification nodes are RECOMMENDED in regions with significant service registrant populations to eliminate intercontinental latency in liveness verification.

1.4.3. Standard Registries

The APIX standard maintains normative registries of enumerated values. Registries are authoritative lists of valid values for specific APM and Index API fields. Using values not present in the relevant registry is a protocol violation.

Registry location: Registries are published as live JSON endpoints at apix.example.org/registry/ and are updated independently of the RFC revision cycle. The RFC defines the registry structure and lifecycle rules; the live endpoints are the authoritative source of current values.

protocols Protocol type registry. Endpoint:
apix.example.org/registry/protocols. APM field: spec.type.

capabilities Capability taxonomy registry. Endpoint:
apix.example.org/registry/capabilities. APM field:
capabilities[].

notification-channels Notification channel type registry. Endpoint:
apix.example.org/registry/notification-channels. APM field:
notifications.channels[].type.

presence-modes Presence mode registry. Endpoint:
apix.example.org/registry/presence-modes. APM field:
spec.presence_mode (device classes).

delegation-scopes Device delegation scope registry. Endpoint:
apix.example.org/registry/delegation-scopes. APM field: scopes[]
in delegation grant requests (device classes).

Initial values for each registry are defined in the applicable profile document: [APIX-SERVICES] for protocol types and capability taxonomy; [APIX-IOT] for presence modes, delegation scopes, and IoT capability terms.

Registry entry lifecycle:

Each registry entry transitions through three phases. The standard_warnings flag in a Service Record does not appear until the grace period has elapsed — service operators have a silent window to update their APM before any public signal is issued.

```
active  ->  deprecated (announced)
              |
              +-- [grace period: 90 days min]
              |     silent: operator notified, no public flag
              |
              +-- [warning period: remainder of deprecation window]
              |     standard_warnings visible in Service Record
              |
              +-- sunset
                    new registrations rejected; flagged non-compliant
```

Phase	Status	standard_warnings	New regs.
Normal use	active	No	Accepted
Grace period	deprecated	*No*	Accepted
Warning period	deprecated	*Yes*	Accepted
Sunset	sunset	Yes (non-compliant)	*Rejected*

Table 1

Deprecation rules:

* The governing body MUST publish a deprecated_in_version, sunset_date, grace_period_ends, and replacement value when deprecating any registry entry.

* The minimum total deprecation window (announcement to sunset) is *12 months*. The governing body MAY extend this window but MUST NOT shorten it.

* The minimum grace period is *90 days* from the deprecation announcement. During the grace period, standard_warnings MUST NOT be set on any Service Record, regardless of whether the service uses the deprecated value.

* The governing body MUST notify all registered Service Owners whose services use the deprecated value before the grace period begins. Notification MUST include the grace_period_ends date, the sunset_date, and the replacement value.

* After the grace period, the index operator MUST set standard_warnings on Service Records that still use the deprecated value.

* At sunset, the index operator MUST reject new APM submissions using the sunsetted value and MUST escalate existing Service Records from standard_warnings to a non_compliant status flag.

Registry versioning: each registry is independently versioned. The Index root resource (Section 10.2) exposes the current version of each registry so consuming agents may detect changes.

1.5. Lawful Cooperation and Non-Surveillance Commitment

1.5.1. Purpose of the Service

APIX is infrastructure designed for one purpose: enabling autonomous agents and the organisations that deploy them to discover legitimate services and operate productively in the commercial internet. Registration in the APIX is a declaration that a service or device class is offered in good faith for legitimate use. The APIX is not a neutral medium indifferent to the purposes for which it is used. It is infrastructure built for legitimate use, and it is by design closed to actors who are refused or removed under the compliance mechanisms defined in this specification — sanctions screening, KYC verification, and judicial enforcement through the LER process.

This is not a policy statement. It is the foundational design constraint from which the cooperation mechanisms in this document and in [APIX-IOT] derive their legitimacy.

1.5.2. Cooperation Duty

Because APIX provides infrastructure for legitimate use, it has a duty to cooperate with properly authorised law enforcement when that infrastructure is misused. This duty is not conditional on commercial convenience or reputational risk. When a registrant or device fleet is confirmed to be operating criminally, APIX MUST act — through the mechanisms defined in this document and in [APIX-IOT] — to limit the harm that flows from that misuse.

APIX MUST cooperate with authorised law enforcement requests that satisfy the jurisdictional and judicial requirements defined in [APIX-IOT] Section 5.8. Refusal to cooperate with a validly authorised request is not permitted. Delay beyond the processing time commitments defined in that section requires documented justification and MUST be reported in the governing body’s annual transparency report.

1.5.3. Non-Surveillance Commitment

APIX is not a surveillance instrument. The cooperation mechanisms in this specification are reactive and bounded. The following prohibitions are normative and apply to all conforming implementations:

* APIX MUST NOT proactively monitor, profile, or analyse the behaviour of registered services, device fleets, or consuming agents beyond what is technically necessary to deliver liveness verification and abuse detection as defined in this specification.

* APIX MUST NOT share index data, presence signal logs, device instance records, or consuming agent query patterns with any law enforcement or government authority except through the Law Enforcement Request process defined in [APIX-IOT] Section 9.8, with its associated judicial authorisation requirements and jurisdictional constraints.

* Bulk data requests — requests that are not targeted at identified specific devices, instances, or registrants but instead seek aggregate ecosystem intelligence — MUST be refused regardless of the requesting authority's jurisdiction or claimed legal basis. A valid LER MUST identify specific device IP addresses or registrant identifiers. A request for "all devices in region X" or "all services in category Y" is not a valid LER.

* APIX MUST NOT establish any data-sharing arrangement, standing access grant, or automated feed to any law enforcement or intelligence agency. Every cooperation action is event-triggered, scoped to a specific identified case, and subject to the judicial authorisation requirement.

1.5.4. The Trigger Requirement

Enhanced monitoring, graduated response actions, and LER processing are ALWAYS triggered by one of two conditions:

1. *External identification*: a legitimate authority in an accepted jurisdiction has submitted an LER with valid judicial authorisation identifying specific devices or registrants as confirmed participants in criminal activity. Suspicion alone is not sufficient. The judicial authorisation requirement is the gatekeeping mechanism.

2. *Technical anomaly detection*: APIX's own infrastructure detects signal patterns technically inconsistent with legitimate device operation — such as rapid mass re-registration from a single IP address, heartbeat flooding at rates outside any plausible device density, or token reuse patterns that cannot arise from legitimate manufacture and provisioning. Such detections result in classification at the observe tier of the Bad-Bot Graduated Response ([APIX-IOT] Section 9.9), not in immediate blocking. They are recorded, monitored, and shared with authorised law enforcement on request through the LER process. They do not trigger autonomous enforcement action by APIX.

Speculative profiling — building behavioural models of registered services or device fleets in the absence of a trigger — is prohibited under the Non-Surveillance Commitment above.

1.5.5. Jurisdictional Guardrails

All cooperation is bounded by the accepted jurisdictions framework defined in [APIX-IOT] Section 9.8. This boundary is not negotiable on a case-by-case basis. APIX MUST NOT cooperate with a law enforcement request from a jurisdiction not on the Accepted Jurisdiction Whitelist, even when:

* The requesting authority presents a compelling case.

* The alleged criminal activity is severe.

* Political, commercial, or reputational pressure is applied.

* Another accepted-jurisdiction authority vouches for the request.

The Accepted Jurisdiction Whitelist exists precisely to make this boundary resist pressure. The governing body MAY add jurisdictions to the whitelist through its defined board decision process; it MUST NOT bypass the whitelist for individual cases. Any governing body action that grants cooperation outside the whitelist is a specification violation and MUST be reported in the transparency report.

1.5.6. Transparency as Enforcement

The annual transparency report required by Section 4.2 is not merely informational. It is the mechanism by which the non-surveillance commitment and the jurisdictional guardrails are held accountable. The governing body MUST disclose in that report:

* The number of LER requests received, accepted, and refused, by requesting jurisdiction tier.

* The number of bulk data requests received and refused.

* Any case in which cooperation outside the accepted jurisdictions framework was requested, with the governing body’s response.

* Any case in which APIX’s own technical anomaly detection was used as the basis for a law enforcement referral.

* The total number of device instances, services, and organisations subject to active suppression, suspension, or graduated response measures at the reporting date.

If a governing body fails to publish this report within 90 days of the close of a calendar year, any member of the governing body board MUST be empowered to publish it unilaterally. The right to publish the transparency report MUST NOT be waivable by board resolution.

1.6. The APIX Manifest (APM)

1.6.1. Purpose

The APIX Manifest is the structured document that a Service Owner provides at registration. It is the index-facing contract for a Service: format-agnostic, extensible, and designed for machine consumption.

The APM has two layers:

Base fields — defined in this document and required for all service types: apm_version, service_id, name, description, owner (with organisation_name, jurisdiction, registration_number, contacts), capabilities, trust (organisation and service level assignments), and legal. These fields are common to all profiles.

lifecycle_stage is required for all service types but its valid values and transition rules are profile-defined. Each profile owns its own lifecycle model; the field is not a shared enum. See [APIX-SERVICES] and [APIX-IOT] for the lifecycle models applicable to each service type.

Profile fields — defined in profile documents and required only for the applicable service type. [APIX-SERVICES] defines the full APM schema for web API services. [APIX-IOT] defines the full APM schema for device class registrations. An APM submission MUST conform to the profile schema corresponding to its spec.type value.

Extension fields — the custom array is a governed extension mechanism for declaring properties not yet covered by the base or profile schemas. The custom field is OPTIONAL in all profiles. It is a flat list of reverse-domain key name strings; no values are stored in the index. The APIX indexes only the declared key names, enabling discovery via the custom_key search parameter. This design provides a clean promotion path: when a custom key accumulates sufficient independent adoption across organisations, the Bot Standards Foundation MAY initiate a governance track to promote the pattern to a standard named field in a future APM version. Full normative rules — including key naming conventions, list size limits, and Spider behaviour — are defined in the applicable profile document ([APIX-SERVICES], [APIX-IOT]).

The trust fields in an APM submission MUST be set exclusively by the index operator based on verification outcomes. APM submissions that include trust field values MUST have those values overwritten by the index upon processing. A Service Owner MUST NOT assert their own trust level.

1.7. Trust Model

The APIX Trust Model has three independent dimensions. Each dimension produces a machine-readable value in the Service Record. Consuming agents combine these values according to their own Trust Policy.

The APIX provides trust metadata. It does not make trust decisions.

1.7.1. Dimension 1 — Organisation Trust Level

Describes the verified identity and compliance posture of the organisation that owns the service.

Level	Label
O-0	Unverified
O-1	Identity Verified
O-2	Legal Entity Verified
O-3	Hygiene Verified
O-4	Operationally Verified
O-5	Audited

Table 2

O-0 (Unverified): Self-registered. No checks performed.

O-1 (Identity Verified): Valid business email confirmed. Domain ownership verified via DNS TXT record.

O-2 (Legal Entity Verified): Company registration number confirmed against official registry of the declared jurisdiction.

O-3 (Hygiene Verified): security.txt (RFC 9116) present and valid at

/.well-known/security.txt; DMARC and SPF DNS records configured for the registered domain; Privacy Policy, Terms of Service, and Data Processing Agreement accessible at declared URLs. All checks performed automatically by APIX. No human reviewer required.

O-4 (Operationally Verified): Organisation governance structure, operational security practices, incident response capability, and personnel vetting reviewed by an Accredited Verifier against the Verifier Standard.

O-5 (Audited): Third-party compliance audit completed (SOC 2 Type II, ISO 27001, or equivalent). Audit certificate on file with the governing body. O-5 may be achieved directly without O-4 as a prerequisite via direct certificate submission to the governing body.

Organisation levels are assessed against the organisation as a whole, not per service. An organisation that achieves any O-level applies that level to all its registered services.

1.7.2. Dimension 2 — Service Verification Level

Describes what has been automatically verified about the service itself. The specific verification mechanism differs by service type (Spider for web API services; manufacturer registration process for device classes).

Level	Label
S-0	Unchecked
S-1	Reachable
S-2	Spec Verified
S-3	Schema Stable
S-4	Security Reviewed

Table 3

S-0 (Unchecked): Registered. Verification has not yet run.

S-1 (Reachable): Service confirmed reachable by automated check.

S-2 (Spec Verified): Specification or capability declaration

confirmed and consistent with registration.

S-3 (Schema Stable): No breaking changes detected across at least three consecutive verification runs.

S-4 (Security Reviewed): Automated vulnerability scan completed with no critical findings, OR third-party penetration test certificate provided and validated by an Accredited Verifier.

Profile documents define the exact criteria by which each level is achieved for each service type.

1.7.3. Dimension 3 — Liveness

Describes the confirmed operational availability of the service, including how recent and how frequent the availability data is. Liveness data is expressed as a set of metrics, not a single level.

last_ping_at (ISO 8601 timestamp) Time of the most recent successful liveness check.

ping_interval_seconds (integer) Configured interval between liveness checks.

uptime_30d_percent (float) Percentage of checks successful over the last 30 days.

avg_response_ms (float) Mean response time in milliseconds over the last 30 days.

consecutive_failures (integer) Number of consecutive failed checks at last run.

The check interval is determined by the service’s liveness monitoring configuration. A service configured at initial-only frequency receives no recurring checks; its last_ping_at reflects only the initial verification run.

The concrete fields and measurement model for Liveness differ by service type and are defined in each profile document.

1.7.4. Trust Model Implementations by Service Type

The three trust dimensions (Organisation, Service Verification, Liveness) are universal across all APIX service types. However, their concrete implementation — the verification mechanisms, the APM fields that carry trust state, and the achievable levels — differs by service type. Three distinct trust implementations are defined across the APIX profile suite.

API Service Trust (defined in [APIX-SERVICES])

Verification is pull-based: the APIX Spider visits the service on a scheduled basis, checks reachability, fetches and parses the specification, and runs schema comparison across consecutive runs. Liveness is measured by the index — the Spider pings the service endpoint and records response time and availability metrics. The trust object in an API service APM carries observed metrics (last_ping_at, uptime_30d_percent, avg_response_ms, consecutive_failures).

Device Class Trust (defined in [APIX-IOT])

Verification is registration-based: a device manufacturer registers the device class, providing a capability declaration and firmware version contract. The APIX Spider does not visit device hardware. Liveness configuration is declared by the manufacturer at registration time (presence_mode, heartbeat_interval_seconds) — not observed by the index. The trust object in a device class APM carries manufacturer-declared configuration, not measured metrics. spec_consistency is always null for device classes: there is no specification document for the Spider to fetch.

Device Instance Trust (defined in [APIX-IOT])

Liveness is push-based: individual device instances signal their presence to the index at regular intervals. The index does not probe devices. Instance trust state (online, reachable, last_seen_at) reflects the most recent presence signal received, not a Spider measurement. Device instance trust state is private — it is never returned to unauthenticated queries regardless of trust levels.

These are three architecturally distinct trust models that share only the O-level and S-level abstractions. Implementers MUST NOT assume that trust object fields in a device class or device instance APM follow the structure of an API service APM.

1.7.5. Bot-Side Trust Policy Expression

A consuming agent expresses its Trust Policy as a set of minimum thresholds across all three dimensions. Example policy expressed in pseudo-notation:

```
require:
  organisation_level >= O-2
  service_level >= S-2
  last_ping_age < 3600         # seconds since last_ping_at
  uptime_30d_percent >= 99.0
  consecutive_failures == 0
```

The Index API SHOULD support filtering by trust dimension thresholds so that agents can retrieve only records that satisfy their policy without downloading the full index.

Trust Policies are defined and enforced by consuming agents. The APIX does not validate or enforce Trust Policies.

1.7.6. Accredited Verifier Model

Organisation level O-3 (Hygiene Verified) is achieved by automatic APIX checks and requires no human reviewer. Organisation level O-4 requires an Accredited Verifier assessment. Organisation level O-5 may be achieved directly without O-4 as a prerequisite via direct certificate submission to the governing body (SOC 2 Type II or ISO 27001). The APIX uses a federated Accredited Verifier model, analogous to the Certificate Authority model in TLS:

* the governing body defines the verification criteria for each level and publishes the Verifier Standard.

* Organisations apply to the governing body for Verifier accreditation.

* Accredited Verifiers perform O-4 assessments and, where applicable, O-5 attestations, signing verification reports in each case.

* the governing body maintains a public registry of Accredited Verifiers and their accreditation status.

* A Service Record at O-4 MUST include the identifier of the Accredited Verifier that performed the assessment and the date of assessment.

* A Service Record at O-5 via direct certificate submission MUST include the certificate reference, issuing auditor, scope, and expiry date.

* Accreditation of Verifiers is reviewed annually by the governing body.

* A Verifier placed in suspended status following a failed annual review MUST be given a minimum 90-day remediation window before final revocation. The 90-day window applies to performance failures: lapsed certifications, reduced capacity, or failure to meet audit quality standards. It does not apply to fundamental violations, for which the governing body MUST revoke accreditation immediately. Fundamental violations include: issuing a false or unsupported O-4 or O-5 assessment, certifying a related entity in breach of the independence requirement, leaking confidential assessment data, or colluding with an organisation to obtain a trust level fraudulently.

Elevation verification requirements:

Elevation to O-4 or O-5 MUST be verified through an out-of-band channel that is independent of the digital submission path used to submit the elevation request. The governing body MUST NOT record an O-4 or O-5 elevation solely on the basis of a digitally submitted application, regardless of the authentication mechanism used for that submission. The out-of-band verification MUST confirm that the elevation was intentionally authorised by a responsible representative of the applicant organisation, and that the submitted evidence (Accredited Verifier report or audit certificate) is genuine.

Elevation to O-5 MUST additionally be confirmed by two independently authorised representatives of the applicant organisation. The two confirming individuals MUST hold separate credentials and MUST act independently; a single individual confirming twice does not satisfy this requirement. The governing body MUST enforce this programmatically for O-5 elevations processed through its operational interface.

The specific out-of-band verification mechanism and the implementation of the two-representative confirmation are operational responsibilities of the governing body and are documented in the APIX implementation guide. Conforming implementations of the APIX governing body role MUST implement mechanisms that satisfy these requirements; the specific mechanisms are not prescribed by this specification.

Design Note — Future Trust Level Evolution (non-normative): The O-0 through O-5 architecture defined here is the Version 1 model. O-3 was introduced to provide an automatable, zero-cost on-ramp for early-stage organisations, bridging the gap between legal entity verification (O-2) and the first human-reviewed tier (O-4). As the governing body Accredited Verifier market matures and a meaningful population of O-5 organisations is established, a Version 2 evolution is anticipated in which O-5 is joined by a premium O-6 designation with APIX-specific assessment criteria beyond the industry baseline — dedicated incident response covering governing body cooperation obligations, agreed governing body audit access, and APIX-specific operational commitments. This evolution requires the governing body to develop and publish an O-6 assessment standard, which is not feasible at initial launch. The trust level record structure (see implementation guide Part I §1.4) is designed to accommodate additional components without breaking existing consumers.

1.8. Commercial Contract and Sanctions Compliance

Every registered service MUST be covered by a commercial agreement between the Service Owner and the index operator (or its Accredited Regional Representative). The agreement MUST define:

* The liveness monitoring configuration and its obligations.

* The index operator’s obligations regarding verification frequency and Index API availability.

* Acceptable use terms.

* Data processing terms in accordance with applicable law.

Sanctions compliance: the index operator MUST screen all service registrants against applicable sanctions lists prior to account activation. At minimum, screening MUST cover the UN Security Council consolidated sanctions list. Operators subject to additional jurisdictional sanctions regimes (e.g., EU, US OFAC, Swiss SECO) MUST additionally screen against those lists as applicable to their jurisdiction of incorporation. Entities subject to applicable sanctions MUST be refused registration regardless of commercial tier.

Registrants MUST represent and warrant in the commercial agreement that they are not subject to applicable sanctions, and MUST notify the index operator immediately of any change in that status.

Ongoing sanctions monitoring: The index operator MUST perform periodic re-screening of all registered organisations against the same sanctions lists checked at initial registration. Re-screening

MUST occur at least quarterly. Upon detection of a new match for a previously-cleared organisation — whether by periodic re-screening, third-party notification, or registrant self-report — the index operator MUST immediately:

1. Suspend the organisation’s account. All API credentials are revoked; no further registration or update operations are accepted from the organisation.

2. Suspend all services registered by the organisation. Suspended services are removed from all discovery results.

3. Revoke all active credentials issued to the organisation (API keys, instance tokens where applicable). All associated service instances are marked offline or unreachable.

4. Open a legal review case. The specific sanctions list and matched entry MUST NOT be disclosed externally; the organisation receives only a generic account suspension notice.

If the sanctions match is subsequently determined to be a false positive or the registrant is removed from the relevant list, the index operator MAY reinstate the account following legal review. Reinstatement requires a fresh KYC and sanctions check.

Unauthenticated discovery queries to the Index API are not subject to registration screening and MUST remain available without restriction, consistent with the APIX’s mission as open global infrastructure.

1.9. Operational Model

1.9.1. Supply-Side Funding Principle

A conforming APIX implementation MUST be funded primarily by service registration fees paid by Service Owners (supply side). Discovery queries by consuming agents MUST NOT be the primary revenue mechanism. This principle is normative: an implementation that charges consuming agents for standard discovery queries is not conformant with the APIX model, as doing so contradicts the open infrastructure mission and undermines the network effect that makes the supply side valuable.

The APIX model is structurally analogous to the DNS model: registrants pay to be listed; queries are free.

Fee structures applicable to each service type are defined in the relevant profile document. All implementations MUST apply fees consistently to all registrants of a given service type at the same

commercial tier, with no preferential treatment. The governing body publishes the normative fee schedule as a separate registry document, updated independently of this RFC.

1.9.2. Consumer Access Model

Discovery queries to the Index API MUST be available without authentication or payment. Rate limits MAY be applied to protect infrastructure integrity but MUST NOT be set at levels that prevent reasonable agent operation. Implementations MUST support at minimum three consumer access layers:

Layer 1 — Unauthenticated access

Any agent MUST be able to query the Index API without authentication or registration, subject to a per-IP rate limit. This layer is sufficient for individual agents and proof-of-concept deployments.

Layer 2 — Authenticated access (free)

Any agent MAY register a consumer identity token at no cost. Token registration requires a valid email address. Authenticated access MUST provide a higher rate limit than unauthenticated access and MAY additionally provide result caching hints and webhook subscriptions for service record changes.

Consumer tokens SHOULD be compatible with the webbotauth identity model ([I-D.meunier-webbotauth-registry]) to enable interoperability with bot authentication infrastructure.

Layer 3 — High-volume access (paid, optional)

Implementations MAY offer a paid high-volume access tier for platforms operating agents at scale that require guaranteed query capacity and operational SLAs. This tier is supplementary; the index’s operational sustainability MUST NOT depend on it.

Public bulk download (REQUIRED)

Implementations MUST provide the full index as a freely downloadable bulk dataset on the first day of each calendar month, without authentication, under the Open Database Licence (ODbL) 1.0. This requirement implements the openness requirement of Section 4.2: no entity, including the index operator, may hold an exclusive lock on the index data.

Implementations MUST additionally publish a daily diff file covering all record additions, updates, and deletions since the previous day. Daily diffs MUST be serialised in the same format as the full snapshot and MUST be available at the same endpoint, identified by an ISO 8601 date in their filename or URL path (e.g. diff-2026-04-28.json). A new mirror MUST be able to reach current index state by downloading the latest monthly full snapshot and applying the sequence of daily diffs since that snapshot date, without downloading any additional full snapshots.

1.9.3. Ecological Impact Transparency

A conforming APIX implementation SHOULD publish aggregate ecological impact statistics derived from observed index usage. These statistics quantify the efficiency gain attributable to machine-native API consumption compared to equivalent traditional web request technology consumption, and SHOULD be updated in real time and included in the annual transparency report.

The comparison baseline is the full traditional web request stack — not payload size alone — including the request waterfall (HTML page with dependent CSS, JavaScript, image, and font resources), JavaScript execution overhead for dynamically rendered pages, polling requests that occur in the absence of a notification mechanism, retry waste from access-control measures, and proxy infrastructure maintained solely to circumvent those measures.

The following metrics SHOULD be derived from directly observable index events and published at a stable public endpoint:

* *Discovery requests served* — each request represents one agent retrieval that did not require scraping or probing a service endpoint directly.

* *Notification events fired* — each event represents one or more polling requests eliminated across all subscribed consuming agents.

* *Estimated data transfer saved (GB)* — computed from discovery request count, service profile type, and the differential between average traditional web page size and average machine-native API response size for that profile type.

* *Estimated CO2 equivalent avoided* — computed from total estimated data transfer saved using a published CO2-per-GB methodology. The methodology document, including its source data and version, MUST be publicly accessible at a stable URL.

All published figures MUST be accompanied by the computation methodology, confidence bounds, and source data references. Conservative estimates MUST be used where data is incomplete; figures MUST NOT be extrapolated beyond what the directly observed data supports.

The governing body SHOULD seek independent validation of the methodology from an established environmental computing research organisation.

1.10. Index API — Core

1.10.1. HATEOAS Navigation Model

The Index API MUST follow Hypermedia as the Engine of Application State (HATEOAS) principles. A consuming agent MUST be able to discover and navigate the entire index starting from a single, stable entry-point URL, without out-of-band knowledge of endpoint paths.

Every response MUST include a _links object containing hypermedia controls for navigation. Link relations MUST use IANA-registered relation types where applicable, and APIX-specific relations where not.

1.10.2. Discovery Endpoint

The APIX exposes a single globally stable entry-point URL:

https://apix.example.org/

A GET request to this URL returns the Index root resource. The root resource includes base navigation links common to all implementations, plus profile-specific links defined in applicable profile documents.

```
{
  "apix_version": "1.0",
  "total_services": 12483,
  "last_updated": "2026-04-25T00:00:00Z",
  "registry_versions": {
    "protocols": "1.0",
    "capabilities": "1.0",
    "presence_modes": "1.0"
  },
  "_links": {
    "self": {
      "href": "https://apix.example.org/"
    },
    "search": {
      "href": "https://apix.example.org/search{/api_version}{?...}",
      "templated": true
    },
    "browse": {
      "href": "https://apix.example.org/browse"
    },
    "capabilities": {
      "href": "https://apix.example.org/capabilities"
    },
    "devices": {
      "href": "https://apix.example.org/devices{?capability,...}",
      "templated": true
    },
    "docs": {
      "href": "https://apix.example.org/docs"
    },
    "apix:ecological-impact-stats": {
      "href": "https://apix.example.org/stats/ecological-impact"
    }
  }
}
```

The {?q,...} placeholder above is abbreviated. The complete search URI template (parameters grouped for readability; the value is a single uninterrupted string at runtime):

```
https://apix.example.org/search{/api_version}
  {?q,capability,protocol,language,pricing_model,
   auth_method,deployment_region,near,coverage_radius_km,
   custom_key,org_level_min,service_level_min,spec_consistency,
   max_ping_age,uptime_30d_min,lifecycle_stage,
   include_superseded,page,page_size}
```

The lifecycle_stage parameter accepts values defined by each profile document. Valid values differ by service type and are not a shared enum. See [APIX-SERVICES] and [APIX-IOT] for the valid values applicable to each service type.

The devices link template (defined in [APIX-IOT]):

```
https://apix.example.org/devices
  {?capability,protocol,online,api_version,
   endpoint_confidence,page,page_size}
```

Profile-specific links (e.g., the devices link defined in [APIX-IOT]) are present in the root resource when the implementation includes support for that profile. Consuming agents MUST follow links rather than constructing URLs independently; the presence or absence of a link in the root resource is the authoritative signal of whether a capability is supported.

1.10.3. Transport Encoding

The Index API is consumed by autonomous agents at machine speed. Response payloads are structured JSON with highly repetitive field names across result arrays. Transport-layer compression achieves 7085% size reduction on typical search result payloads with no information loss and no application-layer schema changes.

Compression support requirements:

The Index API MUST support the following Accept-Encoding values:

Encoding	Requirement	Notes
gzip	MUST	Universally supported baseline
br (Brotli)	SHOULD	Higher compression ratio than gzip
zstd	SHOULD	Similar ratio to Brotli; faster decompression

Table 4

The Index API MUST perform content negotiation via the Accept-Encoding request header. Responses MUST include a Content-Encoding header identifying the applied encoding. If a client sends no Accept-Encoding header, the server MAY respond uncompressed.

Consuming agents SHOULD include Accept-Encoding: zstd, br, gzip in all Index API requests.

The Index API MAY additionally support CBOR (RFC 8949) as a binary alternative to JSON. A client that prefers CBOR MUST signal this via Accept: application/cbor. CBOR responses carry identical information to JSON responses. Clients MUST NOT assume CBOR support. JSON over compressed transport is the normative interchange format.

1.11. Index API Versioning

1.11.1. Version Identification

The root resource returned at https://apix.example.org/ MUST include an apix_version field identifying the version of the Index API schema in use. Version values are of the form MAJOR.MINOR (e.g., "1.0", "1.2", "2.0").

Consuming agents MUST read apix_version at the start of each session. Agents MUST NOT cache apix_version across sessions: the version field is the authoritative signal that the schema has changed.

1.11.2. Compatibility Rules

The APIX follows a semantic versioning policy for the Index API:

Non-breaking changes (MINOR increment):

* Adding new fields to Service Records or the root resource

* Adding new optional query parameters to the search endpoint

* Adding new _links relations to any response

* Expanding an enumerated value registry (new capability terms, new protocol types)

* Increasing rate limits

Minor version increments are backward compatible. A consuming agent written for 1.0 MUST be able to operate correctly against a 1.x endpoint, provided it ignores unknown fields.

Consuming agents MUST follow the robustness principle: ignore unknown fields and unknown link relations rather than failing. This requirement is normative.

Breaking changes (MAJOR increment):

* Removing or renaming fields in Service Records
* Changing the type or semantics of an existing field
* Removing or renaming existing query parameters
* Changing the structure of the HATEOAS _links object
* Changing the URL of the single entry-point

A MAJOR version increment MUST NOT occur without a concurrent deprecation notice for the prior version (see below).

1.11.3. API Deprecation and Migration

When a new MAJOR version is released, the prior MAJOR version MUST remain supported for a minimum of *24 months* from the date the new version becomes available. During this period:

* Both versions MUST be simultaneously queryable
* The root resource of the prior version MUST include a deprecated flag with the sunset_date of the old version
* Consuming agents that include the IETF Sunset header ([RFC8594]) in their responses MUST use it to signal the old version’s sunset date

the governing body MUST NOT sunset a MAJOR version without giving consuming agents at least 24 months to migrate.

1.11.4. Service api_version Immutability Invariant

The api_version field in an APM and the version path segment in the search endpoint (/search/v{major}.{minor}/) rest on a single foundational guarantee: a published api_version value has an immutable field structure definition.

This invariant MUST be stated unambiguously to consuming agents and service operators:

* A field present in version v2.4 will be present in every service that declares api_version: "2.4.x" for the lifetime of that registration.
* A field absent from version v2.4 will never appear in a v2.4.x service record without a version increment.

* Removing a field, changing a field's type, or making any other breaking change REQUIRES a new major version. The major bump is the explicit, sufficient notice to consumers. No deprecation period within a major version is required or expected.

* Adding a field requires a new minor version. Even additive changes are not permitted within a published version — a service that adds a field mid-life has implicitly created a new contract and MUST increment api_version accordingly.

This invariant enables the version path filter to be an unconditional schema contract: an agent that pins to /search/v2.4/ receives results with a fixed, permanent field set. Service owners are freed from the pressure to retain unwanted fields for backwards compatibility — the correct action is always to increment the version and move forward cleanly.

1.11.5. No Cross-Version Response Mapping

The APIX does NOT perform cross-version response mapping. The api_version path segment is a strict storage filter: only service registrations whose api_version field matches the specified prefix are returned. The index never synthesises a response of one version from a record stored at a different version.

The consequence is deliberate and unambiguous:

* A service that has upgraded from v2.4 to v3.0 is stored as a separate record. The v3.0 record does not appear in /search/v2/ results. There are no null substitutions for dropped fields, no type coercions for changed fields, and no partial responses. A v3 record is a different resource; it is not a transformed view of a v2 record.

* The v2.4 record remains in the index — immutably — until the service owner advances it through the lifecycle (deprecated → sunset) or the record is superseded and eventually removed. An agent pinned to /search/v2/ continues to see v2.4 registrations for as long as they exist in the index at that lifecycle stage.

* As services migrate to newer major versions, the v2 result set shrinks. Diminishing or empty results at a pinned version are not a failure condition — they are the designed signal that the consuming agent's version pin no longer covers the current service landscape and an upgrade of consumer code is warranted.

Upgrade path discovery: The Level 2 Service Record for a superseded version MUST include a populated superseded_by field pointing to the current version's record. A consuming agent that finds a v2.4 result with superseded_by set SHOULD follow the link to inspect the v3.0 record and determine whether upgrading its version pin is feasible. This is the mechanism by which agents discover that a newer contract is available without being forced off the old one before they are ready.

A consuming agent that receives only empty results for its pinned version SHOULD query GET /search/ with no path segment and no query parameters. This returns the version landscape only — a summary of available api_version prefixes, service counts, and lifecycle status — and executes no content query. The agent uses this response to identify which version prefix covers the current service population and then issues a new scoped query (e.g., /search/v3/?...) with explicit filters. A parameter-less /search/ MUST NOT return service records; it exists solely as a version discovery resource.

1.11.6. Registry Version Tracking

The root resource exposes a registry_versions object (Section 10.2). Consuming agents that cache capability taxonomy or protocol type data MUST compare the current registry_versions values against their cached version on each session. A change in any registry version MUST trigger a cache refresh before the agent applies trust filtering or capability matching.

1.12. Operator Security and Self-Governance

1.12.1. Purpose and Scope

APIX centralises knowledge that has intrinsic intelligence value: the identity and capability of every registered service, the network location of every online IoT device instance, the query patterns of every consuming agent, and the contact details of law enforcement authorities across accepted jurisdictions. This concentration makes the Bot Standards Foundation an ultra-high-value target for state-sponsored actors, criminal organisations, and corporate adversaries.

The Non-Surveillance Commitment (Section 5) defines what APIX will not do to the ecosystem it serves. This section defines what the Bot Standards Foundation MUST do to protect itself and the ecosystem from being exploited involuntarily — through compromise, coercion, insider threat, or organisational capture.

The requirements in this section are normative obligations on the Bot Standards Foundation as operator. They are not addressed to Service Owners or consuming agents.

1.12.2. Technical Security Requirements

the governing body MUST operate APIX infrastructure under the following technical constraints:

Infrastructure separation: The token store, tamper-evident audit log, and LER processing queue MUST be hosted on systems with no shared network path to the public-facing Index API query infrastructure. Compromise of the query layer MUST NOT provide lateral access to the token store or audit log.

Air-gapped token issuance: Instance token batches for IoT device classes MUST be generated on infrastructure with no persistent internet connection. Issuance systems MUST use hardware security modules (HSMs) for all cryptographic operations. The issuance network MUST be physically separated from the token delivery network.

Geographic distribution: Core APIX systems MUST be distributed across at least two independent physical jurisdictions. No single legal order from any one jurisdiction MUST be sufficient to take the full system offline or compel full data access.

Zero-trust internal architecture: No governing body system MUST grant implicit trust to requests from other governing body systems. All inter-system communication MUST be authenticated and authorised independently of network location. Lateral movement within governing body infrastructure MUST require separate credentials at each boundary.

Cryptographic floor: All external-facing endpoints MUST use TLS 1.3 or higher ([RFC8446]). All signing operations MUST use asymmetric keys stored in hardware-backed key storage. Key material MUST NOT be exportable from the HSM in plaintext under any operational procedure.

Mandatory penetration testing: The governing body MUST commission an independent penetration test of its production infrastructure at least annually. A summary of findings (severity distribution, remediation status) MUST be published in the governing body’s annual security report within 90 days of the test. The identity of the testing firm MUST be disclosed.

Responsible disclosure programme: The governing body MUST maintain a public responsible disclosure policy at a stable URL and MUST acknowledge vulnerability reports within 5 business days.

1.12.3. Organisational Security Requirements

Personnel vetting: All staff and contractors with access to the token store, LER queue, sanctions screening pipeline, or audit log MUST undergo documented background verification commensurate with the sensitivity of the systems they can access, prior to access being granted. Access MUST be reviewed annually.

Segregation of duties: No individual staff member MUST hold simultaneous access to more than two of the following: token store, audit log, LER queue, sanctions pipeline, board signing keys. This constraint MUST be enforced technically, not procedurally.

Least-privilege access: Access rights MUST be scoped to the minimum required for the role. Privileged access MUST expire after a defined session window and MUST require re-authentication. No standing privileged sessions are permitted.

Security awareness: All governing body staff MUST complete security awareness training annually, covering at minimum the threat types and unlawful request scenarios relevant to an operator under the security obligations defined in this section.

Insider threat detection: The governing body MUST operate anomalous access pattern detection across all privileged systems. Anomalies MUST generate alerts to a security function independent of the alerted staff member’s reporting line.

Whistleblower protection: Any governing body staff member or contractor who receives an instruction — from any source, including governing body board members — that would cause the governing body to act contrary to the Non-Surveillance Commitment (Section 5) or the requirements of this section MUST have a protected right to report that instruction to an external body without prior internal approval. This right MUST be codified in the governing body’s founding charter charter and in every employment and contractor agreement. It MUST NOT be waivable by board resolution or individual contract term.

1.12.4. Political Independence and Anti-Capture Measures

Structural domicile: the governing body MUST maintain its Swiss Stiftung domicile as a structural defence. The Swiss legal system, political neutrality, and the oversight of the Eidgenssische Stiftungsaufsicht provide a foundation that is not subject to the data access regimes of any single major power.

Golden share: the governing body's charter MUST maintain a governance mechanism equivalent to a 51% golden share that prevents any acquisition, merger, or board supermajority from overriding the charter's core purpose. No commercial transaction MUST be permitted to subordinate the governing body's neutrality obligations to the interests of a single organisation or jurisdiction.

Board composition: No single nation-state's citizens or residents MUST hold a majority of board seats. No individual MUST hold more than one vote on any board decision. Board composition MUST be published annually in the transparency report.

Infrastructure jurisdiction policy: The governing body MUST NOT host core APIX systems — token store, audit log, LER queue — in jurisdictions that impose secret data access orders (orders that legally prohibit the recipient from disclosing that the order was received). The governing body MUST maintain a published list of approved hosting jurisdictions, reviewed annually by the board. Removal of a jurisdiction from the approved list MUST trigger migration of any systems hosted there within 180 days.

Lawful pressure resistance: If the governing body receives a government demand for data access, system access, or operational changes that does not satisfy the LER criteria defined in [APIX-IOT] Section 9.8, The governing body MUST refuse the demand. The governing body MUST record the demand in the audit log and MUST report its existence — without operational detail that would compromise an ongoing investigation — in the next annual transparency report. The governing body MUST NOT comply with informal diplomatic pressure, agency-level requests, or extra-judicial demands regardless of the requesting party's political standing.

Anti-capture review: The board MUST conduct an annual review of whether any commercial relationship, grant dependency, or staff composition creates a conflict of interest with the governing body's neutrality obligations. Findings MUST be published in the transparency report.

1.12.5. Crisis Governance Protocol

The following conditions each independently trigger the governing body crisis governance protocol:

* Credible evidence that APIX production infrastructure has been compromised by an external actor

* Receipt of a demand that the governing body's legal counsel assesses as an attempt to compel action contrary to the charter

* Attempted hostile acquisition, board capture, or charter amendment by a party with a conflict of interest

* Regulatory action that threatens loss of Swiss Stiftung domicile

Obligations on trigger:

1. The discovering party MUST notify all board members within 4 hours.

2. The governing body MUST publish a public statement acknowledging the trigger event within 72 hours of confirmation. The statement MUST describe the nature of the threat in general terms without disclosing operational detail that would aid the attacker.

3. The governing body MUST activate its continuity-of-operations plan, ensuring Index API availability is maintained independently of any compromised or coerced system.

4. If Swiss domicile is threatened or lost, the board MUST convene within 30 days to activate a pre-agreed organisational relocation framework. The destination jurisdiction MUST satisfy the infrastructure jurisdiction policy defined above. The relocation framework MUST be prepared and approved by the board before APIX reaches production operation and MUST be reviewed annually.

No single board member and no external party MUST have the authority to suspend or delay execution of steps 13 above.

1.12.6. Data Minimisation as Security Policy

The least-held data is the least-leakable data. The following constraints apply to all APIX operational systems:

* APIX MUST NOT log consuming agent query patterns beyond the minimum required for liveness monitoring and abuse detection. Query logs MUST be purged after 30 days unless retained under a specific, documented, time-limited LER investigation scope.

* Device instance network location data (network.ipv6, as published in the instance record) MUST be purged from APIX systems within 72 hours of the instance transitioning to offline status, subject to any active LER retention obligation on that instance. The internally observed source IPv4 address (observed_source_ipv4, retained for abuse detection and geo-routing and not surfaced in the instance record) is subject to the same purge obligation and timeline.

* APIX MUST NOT build or maintain cross-session behavioural profiles of consuming agents. Each query session MUST be treated as independent.

* Every data field collected or retained by APIX MUST have a documented functional justification. Fields without a current functional justification MUST be deleted from the data model in the next schema revision. This review MUST be a standing agenda item at each the governing body board meeting.

1.12.7. Annual Security Report

the governing body MUST publish an annual security report within 90 days of the close of each calendar year. The security report is separate from the transparency report defined in Section 5.6 and MUST contain:

* Summary of the year’s penetration test findings: severity distribution (critical / high / medium / low count), remediation status of prior findings, identity of testing firm

* Summary of infrastructure changes affecting the attack surface

* Staff access review outcomes: number of access rights granted, revoked, and modified

* Count of external demands received that did not meet LER criteria, and how each was handled

* Count of whistleblower reports received and their resolution status (no identifying detail)

* Board attestation that the infrastructure jurisdiction policy was reviewed and remains current

The same unilateral publication right defined for the transparency report (Section 5.6) applies to the security report: if the board fails to publish within 90 days of period close, any individual board member MUST be empowered to publish it unilaterally. This right MUST NOT be waivable by board resolution.

1.13. Security Considerations

1.13.1. Abuse and Fake Listings

The mandatory Terms of Service acceptance at registration provides a first barrier against malicious actors listing fake or harmful services. For O-0 and O-1, identity verification is limited; consuming agents SHOULD NOT rely solely on index presence for trust at these levels. For O-2 and above, the formal B2B contractual relationship and progressively stronger identity and compliance verification substantially raise the cost of abuse.

Consuming agents SHOULD apply Trust Policies that exclude O-0 services for any task involving sensitive data or consequential actions.

the governing body MUST maintain an abuse reporting mechanism and MUST be able to suspend or remove a Service Record within 24 hours of confirmed abuse. Suspended service records MUST remain in the index with a status: suspended flag and MUST NOT be silently deleted, to provide transparency to agents that had cached the record.

1.13.2. Trust Level Spoofing

Organisation and Service trust levels in the Service Record are set only by the APIX itself, not by the Service Owner. APM submissions that include trust field values MUST have those values overwritten by the APIX upon processing. The Index API MUST NOT expose self-asserted trust values.

1.13.3. Transport Security Requirements

The Index API MUST be served exclusively over TLS ([RFC8446]). Certificate validity MUST be verified by consuming agents. Agents MUST NOT bypass TLS certificate verification when querying the Index API.

All entry_point and spec.url values submitted in APM registrations MUST use the https scheme. The Index MUST reject APM submissions that provide HTTP (non-TLS) values for these fields.

1.13.4. Bot Consumer Risks

The APIX provides discovery and trust metadata. It does not guarantee the safety, correctness, or availability of listed services. Consuming agents MUST NOT assume that a service listed in the APIX is safe to use without applying their own Trust Policy.

Consuming agents SHOULD treat Index API responses as untrusted input and validate the structure of Service Records before acting on them.

2. References

2.1. 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>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/rfc/rfc5646>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8594] Wilde, E., "The Sunset HTTP Header Field", RFC 8594, DOI 10.17487/RFC8594, May 2019, <https://www.rfc-editor.org/rfc/rfc8594>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[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>.

[RFC9116] Foudil, E. and Y. Shafranovich, "A File Format to Aid in Security Vulnerability Disclosure", RFC 9116, DOI 10.17487/RFC9116, April 2022, <https://www.rfc-editor.org/rfc/rfc9116>.

2.2. Informative References

[A2A] Linux Foundation, "Agent2Agent (A2A) Protocol", June 2025, <https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents>.

[AAIF] Linux Foundation, "Linux Foundation Agentic AI Foundation (AAIF)", December 2025, <https://www.linuxfoundation.org/press/linux-foundation-announces-the-formation-of-the-agentic-ai-foundation>.

[AGNTCY] Linux Foundation, "AGNTCY: Open Multi-Agent System Infrastructure", July 2025, <https://www.linuxfoundation.org/press/linux-foundation-welcomes-the-agntcy-project-to-standardize-open-multi-agent-system-infrastructure-and-break-down-ai-agent-silos>.

[APIX-IOT] Rehfeld, C., "APIX IoT Device Profile: Discovery and Presence for Connected Device Services", Work in Progress, Internet-Draft, draft-rehfeld-apix-iot-00, n.d., <https://datatracker.ietf.org/doc/draft-rehfeld-apix-iot/>.

[APIX-SERVICES] Rehfeld, C., "APIX Services Profile: Discovery Infrastructure for Web API and Bot Services", Work in Progress, Internet-Draft, draft-rehfeld-apix-services-00, n.d., <https://datatracker.ietf.org/doc/draft-rehfeld-apix-services/>.

[I-D.aiendpoint-ai-discovery] Choi, Y., "The AI Discovery Endpoint: A Structured Mechanism for AI Agent Service Discovery and Capability Exposure", March 2026, <https://datatracker.ietf.org/doc/draft-aiendpoint-ai-discovery/>.

[I-D.am-layered-ai-discovery-architecture] Moussa, H. and A. Akhavain, "A Layered Approach to AI discovery", Work in Progress, Internet-Draft, draft-am-layered-ai-discovery-architecture-00, 14 March 2026, <https://datatracker.ietf.org/doc/html/draft-am-layered-ai-discovery-architecture-00>.

[I-D.batum-aidre] Batum, F., "AI Discovery and Retrieval Endpoint (AIDRE)", Work in Progress, Internet-Draft, draft-batum-aidre-00, 5 April 2026, <https://datatracker.ietf.org/doc/html/draft-batum-aidre-00>.

[I-D.cui-ai-agent-discovery-invocation]
Cui, Y., Chao, Y., and C. Du, "AI Agent Discovery and Invocation Protocol", Work in Progress, Internet-Draft, draft-cui-ai-agent-discovery-invocation-01, 12 February 2026, <https://datatracker.ietf.org/doc/html/draft-cui-ai-agent-discovery-invocation-01>.

[I-D.drake-agent-identity-registry]
Drake, J., "Agent Identity Registry System: A Federated Architecture for Hardware-Anchored Identity of Autonomous Entities", 2026, <https://datatracker.ietf.org/doc/draft-drake-agent-identity-registry/>.

[I-D.hood-agtp-api]
Hood, C., "AGTP-API: Verbs, Paths, Endpoints, and Synthesis", Work in Progress, Internet-Draft, draft-hood-agtp-api-00, 11 May 2026, <https://datatracker.ietf.org/doc/html/draft-hood-agtp-api-00>.

[I-D.hood-agtp-discovery]
Hood, C., "AGTP Agent Discovery and Name Service", Work in Progress, Internet-Draft, draft-hood-agtp-discovery-00, 23 March 2026, <https://datatracker.ietf.org/doc/html/draft-hood-agtp-discovery-00>.

[I-D.hood-agtp-trust]
Hood, C., "AGTP Trust and Verification Specification", Work in Progress, Internet-Draft, draft-hood-agtp-trust-00, 11 May 2026, <https://datatracker.ietf.org/doc/html/draft-hood-agtp-trust-00>.

[I-D.hood-independent-agtp]
Hood, C., "Agent Transfer Protocol (AGTP)", Work in Progress, Internet-Draft, draft-hood-independent-agtp-07, 11 May 2026, <https://datatracker.ietf.org/doc/html/draft-hood-independent-agtp-07>.

[I-D.ietf-scitt-architecture]
Birkholz, H., Delignat-Lavaud, A., Fournet, C., Deshpande, Y., and S. Lasker, "An Architecture for Trustworthy and Transparent Digital Supply Chains", Work in Progress, Internet-Draft, draft-ietf-scitt-architecture-22, 10 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-scitt-architecture-22>.

[I-D.meunier-webbotauth-registry]
Guerreiro, M., Kirazci, U., and T. Meunier, "Registry and Signature Agent card for Web bot auth", Work in Progress, Internet-Draft, draft-meunier-webbotauth-registry-01, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-meunier-webbotauth-registry-01>.

[I-D.mozley-aidiscovery]
Mozley, J., Williams, N., Sarikaya, B., and R. Schott, "AI Agent Discovery (AID) Problem Statement", Work in Progress, Internet-Draft, draft-mozley-aidiscovery-01, 16 April 2026, <https://datatracker.ietf.org/doc/html/draft-mozley-aidiscovery-01>.

[I-D.mozleywilliams-dnsop-dnsaid]
Mozley, J., Williams, N., Sarikaya, B., and R. Schott, "DNS for AI Discovery", Work in Progress, Internet-Draft, draft-mozleywilliams-dnsop-dnsaid-01, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-mozleywilliams-dnsop-dnsaid-01>.

[I-D.narajala-courtney-ansv2]
Courtney, S., Narajala, V. S., Huang, K., Habler, I., and A. Sheriff, "Agent Name Service v2 (ANS): A Domain-Anchored Trust Layer for Autonomous AI Agent Identity", Work in Progress, Internet-Draft, draft-narajala-courtney-ansv2-01, 13 April 2026, <https://datatracker.ietf.org/doc/html/draft-narajala-courtney-ansv2-01>.

[I-D.pioli-agent-discovery]
Pioli, R., "Agent Registration and Discovery Protocol (ARDP)", Work in Progress, Internet-Draft, draft-pioli-agent-discovery-01, 24 February 2026, <https://datatracker.ietf.org/doc/html/draft-pioli-agent-discovery-01>.

[I-D.vandemeent-ains-discovery]
van de Meent, J. and R. AI, "AINS: AInternet Name Service - Agent Discovery and Trust Resolution Protocol", Work in Progress, Internet-Draft, draft-vandemeent-ains-discovery-01, 29 March 2026, <https://datatracker.ietf.org/doc/html/draft-vandemeent-ains-discovery-01>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/rfc/rfc6763>.

[ROBOTS] Koster, M., "The Web Robots Pages", 1994, <https://www.robotstxt.org/>.

[UDDI] Clement, L., Hately, A., von Riegen, C., and T. Rogers, "UDDI Version 3.0.2", OASIS Committee Draft uddi-v3.0.2-20041019, 19 October 2004, <https://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm>.

[W3C-AGENTPROTOCOL] Chang, G. and S. Xu, "W3C AI Agent Protocol Community Group", 8 May 2025, <https://www.w3.org/community/agentprotocol/>.

[WEBBOTAUTH-WG] "webbotauth IETF Working Group", n.d., <https://datatracker.ietf.org/wg/webbotauth/>.

Appendix A. Change Log

draft-rehfeld-apix-core-00: Initial submission, April 2026.

draft-rehfeld-apix-core-01: Related Work section expanded to cover AGNTCY (Linux Foundation), A2A Protocol (Linux Foundation), draft-drake-agent-identity-registry, and the Linux Foundation Agentic AI Foundation (AAIF). Positioning paragraph updated to reflect the consolidation of communication and invocation standards under the AAIF and APIX's complementary position as the discovery layer. MCP entry updated with AAIF governance note. Four new informative references added: AAIF, AGNTCY, A2A, I-D.drake-agent-identity-registry. "The Discovery Shift" section scoped to a precise technical problem statement — strategic framing removed to keep the section appropriate for an IETF specification document. AGNTCY scope comparison corrected: "commercial services" replaced with "agent-consumable services and IoT device classes" to reflect the full scope of both APIX profiles.

A.1. IANA Considerations

This document requests no IANA actions. Registry structures defined here are maintained by the governing body at apix.example.org/registry/. Initial registry values are defined in [APIX-SERVICES] and [APIX-IOT].

A.2. References

A.2.1. Normative References

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

* [RFC8259] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 8259, December 2017.

* [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.

* [RFC8594] Wilde, E., "The Sunset HTTP Header Field", RFC 8594, May 2019.

* [RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, May 2019.

* [RFC9110] Fielding, R., et al., "HTTP Semantics", RFC 9110, June 2022.

* [RFC9116] Foudil, E., Shafranovich, Y., "A File Format to Aid in Security Vulnerability Disclosure", RFC 9116, April 2022.

A.2.2. Informative References

* [APIX-SERVICES] Rehfeld, C., "APIX Services Profile", draft-rehfeld-apix-services-00.

* [APIX-IOT] Rehfeld, C., "APIX IoT Device Profile", draft-rehfeld-apix-iot-00.

* [UDDI] Clement, L., et al., "UDDI Version 3.0.2", OASIS, 2004.

* [ROBOTS] Koster, M., "The Web Robots Pages", 1994.

* [I-D.pioli-agent-discovery], [I-D.narajala-courtney-ansv2], [I-D.vandemeent-ains-discovery], [I-D.aiendpoint-ai-discovery], [I-D.meunier-webbotauth-registry], [I-D.cui-ai-agent-discovery-invocation], [I-D.am-layered-ai-discovery-architecture], [I-D.hood-agtp-discovery], [I-D.mozleywilliams-dnsop-dnsaid], [I-D.batum-aidre], [I-D.mozley-aidiscovery] - Related Internet-Drafts, Section 1.6.

* [W3C-AGENTPROTOCOL] Chang, G., Xu, S., "W3C AI Agent Protocol Community Group", 2025.

* [WEBBOTAUTH-WG] "webbotauth IETF Working Group".

A.3. Author’s Address

Carsten Rehfeld Email: carsten@botstandards.org

Author’s Address

Carsten Rehfeld
Email: carsten@botstandards.org