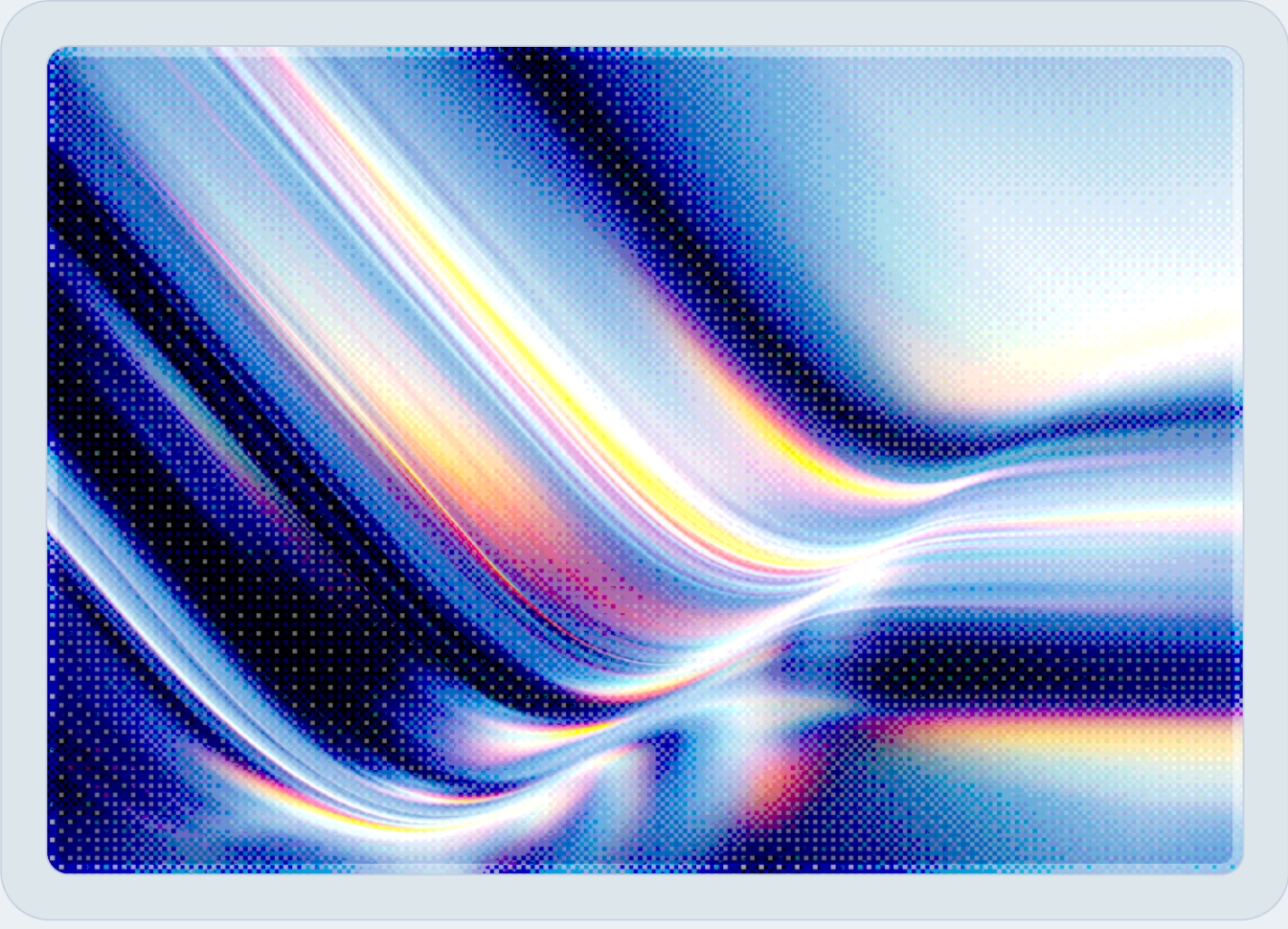
CSAI Foundation | Cloud Security Alliance

Agent Disclosure

A lightweight disclosure convention for AI agents, built on IETF Web Bot Auth

November 2026

AI-assisted Rapid Research



CSAI

cloud security alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Draft 0.1. The vocabulary may change before `v=1` is frozen. Try it, implement it, tell us what breaks – but don't ship it to production yet.

Try it now:

```
curl -s https://disclosure.cloudsecurityalliance.ai/assess \
  -H 'Agent-Disclosure: v=1, type=ai, mode=autonomous,
authority=delegated'
```

Then watch it appear on the live feed. A reader who runs that line before reading the rest will read the rest differently.

Key Takeaways

- A bank receiving a signed request today can establish **which key** sent it. It cannot establish **what kind of actor** is behind that key — a person, an agent with a human approving each step, or an agent acting alone. Those three cases warrant different handling of a payment instruction.
- We propose **one HTTP header**. No new cryptography, no new identifier, no registry of agents. It is a semantic profile layered on IETF Web Bot Auth, which already supplies the signature.
- Every value is **an assertion by the holder of the signing key, not a proof**. The convention is designed for honest actors and says so plainly.
- A relying party sorts traffic into three buckets: verified agent, declared agent, unknown actor. **The absence of the header never means a human.**
- A public relying party is running now. We are looking for one organization to implement that side independently, and for review from the Web Bot Auth community. We are not seeking adoption commitments.

The problem

Web Bot Auth is solving the identity half of this problem well, and we have no interest in solving it twice. An agent signs its request, a relying party verifies the signature and resolves the operator's key directory, and the relying party learns that the request genuinely came from a key controlled by a named operator. That is real and it is useful.

What the signature does not carry is the thing a risk engine actually needs. Web Bot Auth says so explicitly: a signature "does not express authorization, delegation, or consent," and those meanings "are deployment policy, or are carried in separately signed fields" (§5.2.2). So a bank that has verified a request from `acme.ai` still does not know whether a customer is sitting at a keyboard, whether an agent is executing a workflow with defined human approval points, or whether an agent is running unattended at three in the morning.

Today that gap gets filled by guesswork — User-Agent sniffing, behavioral heuristics, and the quiet assumption that unremarkable traffic is probably human. Each of those is wrong often enough to matter, and the error rate gets worse as agent traffic grows. Practitioners report the same gap in their own logs [4].

The proposal

One header field, in the Structured Fields format HTTP already uses:

```
Agent-Disclosure: v=1, type=ai, mode=autonomous,
authority=delegated,
                  operator="acme.ai"
```

A version and three required values, each a closed set:

Member	Values	Answers
`v`	`1`	Which version of this profile
`type`	`ai` , `automation`	Is this a model-driven agent or a deterministic tool

Member	Values	Answers
`mode`	`assisted`, `supervised`, `autonomous`, `unknown`	How independently is it operating **on this request**
`authority`	`operator`, `delegated`, `unspecified`	Acting for its own operator, or on behalf of someone else

There is deliberately no `human` value. An unverifiable claim to be a person is the single most attractive thing for an adversary to forge, and the one claim this convention must never appear to support. A plain browser driven by a person sends nothing; a person using an AI assistant sends `mode=assisted`.

Two optional members round it out: `operator`, a DNS name, and `card`, an HTTPS link to a published description of the agent. Unknown members are ignored rather than treated as errors, so the vocabulary can grow without stranding deployed verifiers.

Where the sender already signs under Web Bot Auth, it includes `agent-disclosure` among the signed components. This is not a new mechanism — Web Bot Auth already provides for signing additional headers and offers, as its own worked example, that "an agent might include an HTTP header expressing its intent and sign it" (§5.2.4). The field inherits integrity from machinery that exists, at no extra round trip.

What a relying party does with it

Every request sorts into exactly one of three classes. A **verified agent** has a valid signature that covers this field and a key directory that resolves. A **declared agent** sent a well-formed field without that proof. An **unknown actor** sent nothing, or sent something malformed.

The third class is where the design earns its keep, so it is worth being blunt about it. **Unknown actor does not mean human.** It means undeclared, and it is handled by exactly the controls a relying party already applies to unattributed traffic — session authentication, device signals, behavioral risk scoring, step-up. Adopting this convention changes nothing about that population. What it adds is the ability to identify a *cooperating* subset and treat it differently.

Here is the example policy the demo enforces, reproduced from the receivers page, with a demo limit of 5,000:

Operation	verified · assisted	verified · supervised	verified · autonomous	declared	unknown
read	allow	allow	allow	allow (rate-limited)	existing controls
pay ≤ 5,000	allow	allow + record approval evidence	allow only if delegation scoped	step-up	existing controls
pay > 5,000	step-up	require verified customer approval	deny	deny	existing controls
change ownership	step-up	deny	deny	deny	existing controls

One caveat about the demo, so nobody files it as a bug: signature verification is still a stub there, so the service returns only `declared-agent` or `unknown-actor` and the three `verified` columns are not currently reachable. The classification logic is complete; the signature layer is pending. The demo's source is public [7], including the stub, so you can see exactly where that seam is.

Two properties of that table are the whole argument. A declared agent never gets more than a verified agent at the same mode, because a declaration is a claim and a signature is evidence. And the `unknown` column is not a reward — if undeclared traffic were ever treated *better* than declared traffic, the convention would teach agents to hide, and it would deserve to fail.

What it deliberately does not do

It does not prove anything. `mode=supervised` is a claim that a human approval step exists, and nothing in the header distinguishes a workflow where a named officer approves a payment from one where a second model emits an approval event. Relying parties must keep **claimed supervision** separate from **verified approval**, and must not accept the former where their risk policy calls for the latter. Verified approval, where it matters, comes from a channel the relying party controls — token introspection, a consent record, a step-up it initiated.

It does not grant authorization. `authority=delegated` says a delegation is *claimed*, so that the absence of a corresponding credential becomes a detectable inconsistency rather than a silent assumption. The delegation itself still gets validated where it always was: in the token, the consent record, or the payment rail's own signed artifact.

It does not detect anything. There is no fingerprinting, no anti-evasion, no attempt to unmask agents that stay quiet. This is an honest-actor convention, and a sender that lies is handled by the controls that already handle unattributed traffic.

It does not register agents, mint identifiers, define cryptography, or claim to satisfy any statute — including Article 50 of the EU AI Act, which binds providers and runs toward the person interacting with a system, not toward a server receiving a request.

Three objections worth answering now

"Why not just put this in the agent's published card?" Because autonomy is a property of the request, not of the operator. The same agent may run attended at nine in the morning and unattended at midnight, and a static document cannot say which is happening now. The relying party also needs the answer *before* it answers the request, and a card is fetched out of band on its own cache schedule. The signature already covers headers, so a header costs nothing extra.

"Isn't DNS the place for this?" Partly. The IETF DNS-AID draft lets an organization publish its agents in DNS [5]; that is the callee side, a cacheable record of what an organization runs. The header is the caller side, a per-request claim. They do not overlap, and the demo uses both: when a request names an `operator`, the relying party checks in the background whether that domain publishes agents, and whether the header matches the default they publish. The result is reported in a `corroboration` block and never changes the classification — DNS is a verifiable transport for metadata, not proof of the claim. That corroboration, and the two private-use keys it reads (`disclosure=` and `webbotauth-keys=`), are experimental and appear in no draft.

"Agents will simply lie, or say nothing." Some will. The convention's value is bounded by the fraction of operators who choose to cooperate, and that fraction is a function of incentives, not cryptography. The incentive offered here is positive and concrete: declare yourself truthfully and get predictable, documented treatment — including access that relying parties would otherwise deny to unattributed automation. That is the same bargain `robots.txt` made, and it held for more than thirty years because following it was cheaper than being blocked.

Status, and what we are asking for

Version 0.1 is a working draft. The vocabulary may change before `v=1` is frozen; do not deploy to production on it yet. It is not an IETF submission and has no IETF standing, and it registers nothing with IANA. A full specification exists with a JSON schema, twenty conformance test vectors, and five reference implementations — client libraries in Python and TypeScript, and relying-party verifiers for FastAPI, Express, and Cloudflare Workers — all passing all twenty vectors. It is being prepared for publication, and the site will link it when it lands.

A public relying party is running at `disclosure.cloudsecurityalliance.ai`, its source is on GitHub [7], and two CSA agents — both listed on the adopters page — send the header to it on a schedule from unattended workflows. What has not happened is an implementation by someone other than the author. That is the success criterion for v0.1 — one AI agent identifying itself to an independently operated application, and that application making a different policy decision because of it.

So the ask is small and specific, and there is somewhere to do each part of it:

- **If you operate agents**, send the header and tell us what was awkward to determine. Implement it has copy-paste setup for curl, Python, JavaScript, Claude Code and OpenClaw; try it lets you build a header in the browser and see how it classifies.
- **If you receive agent traffic**, implement the relying-party side and tell us what breaks. The classification rules and the example policy table are on the receivers page, and the demo's own implementation is on GitHub under Apache-2.0, to read, run or reuse.
- **Either way**, watch your requests land on the live feed, and get listed on adopters — an unvetted, self-reported list that is not a registry and verifies nothing.
- **If you work on Web Bot Auth**, tell us whether this belongs in the working group instead. CSA would rather contribute this vocabulary than maintain a competing mechanism.

References

[1] Meunier, T. and S. Major. "HTTP Message Signatures for automated traffic." `draft-ietf-webbotauth-httpsig-protocol-01`, IETF Internet-Draft, 29 October 2026 (work in progress). Quotations above are from §5.2.2 and §5.2.4 of that revision.

[2] Backman, A., Richer, J., and M. Sporny. "HTTP Message Signatures." RFC 9421, February 2024.

[3] Nottingham, M. and P-H. Kamp. "Structured Field Values for HTTP." RFC 9651, September 2024.

[4] Cloud Security Alliance. "Identity and Access Gaps in the Age of Autonomous AI." 23 March 2026. Survey of 228 practitioners, fieldwork January 2026, vendor-sponsored.

[5] Mozley Williams, et al. "DNS for AI Discovery." `draft-mozleywilliams-dawn-dnsaid-00`, IETF Internet-Draft, 2 November 2026 (work in progress).

[6] Cloud Security Alliance with OWASP AI Exchange. "Agentic AI Red Teaming Guide." 28 May 2025.

[7] Cloud Security Alliance. "Agent Disclosure public demo." Source for the demo relying party at `disclosure.cloudsecurityalliance.ai`, Apache-2.0. Draft 0.1; signature verification is a stub pending a later phase.

© 2026 Cloud Security Alliance. AI-assisted Rapid Research.