



VerifiedProxy

OPEN SPECIFICATION · DRAFT 0 · OCTOBER 2026

Commissioned Agent Records

Proving which company commissioned an AI agent, and what it asked for. An open standard built on Web Bot Auth.

Identity

Which key signed this request? Web Bot Auth, unchanged.

Commission

Which company sent this agent, and what may it do?

Record

Did it really ask for this? An independent, provable record.

Published by VerifiedProxy, the first independent record operator.
Free to implement. No part of the standard requires VerifiedProxy.

For discussion
verifiedproxy.com

A platform should be able to say: yes, that is Halden & Co's agent, asking for that.

AI agents now place orders, book travel and pull financial data on behalf of companies. When a request arrives, the platform receiving it needs more than a cryptographic identity. It needs to know who is accountable, what that company allowed, and whether anyone independent saw the request.

At a glance

- **Web Bot Auth**, being standardised at the IETF, lets an agent prove which key signed a request. It does not say which company stands behind the agent, what it is allowed to do, or whether anyone independent saw the request.
- **Commissioned Agent Records** adds two layers on top: a **commission** the company publishes on its own domain, and a **record** an independent operator writes before the request is sent.
- A platform checks all three with standard building blocks — HTTP Message Signatures, JWS, DNS and a Certificate Transparency-style log — and never has to take the operator's word for anything.
- It works whether a company runs its own agents or uses one an **AI provider** runs for it.
- It is **open**: anyone can run an operator. Companies name theirs in DNS; platforms find it automatically.

Contents

1	Why identity is not enough	9	An open ecosystem of operators
2	Three layers	10	Agents run by AI providers
3	How one request is verified	11	Adopting it
4	The commission	12	VerifiedProxy, the first operator
5	The signed request	13	Status, open questions and how to take part
6	Records and the log	A	The specification (Draft 0)
7	Verification	B	References
8	Why you can trust it		

Halden & Co, halden.example, supplies.example and other .example names are fictional examples used throughout. "Commissioned Agent Records" is a working name.

Why identity is not enough

Web Bot Auth gives agents what TLS certificates gave websites: a way to prove which key is speaking. An agent signs each HTTP request with an Ed25519 key, and the platform checks the signature against a key directory the agent publishes. That is an essential foundation, and this standard changes nothing about it.

But a key is not a company, and a signature is not an authorisation. When Halden & Co's purchasing agent places a £4,200 order at a supplier, three questions matter — and identity alone answers only the first:

THE PLATFORM NEEDS TO KNOW	WEB BOT AUTH	WITH COMMISSIONED AGENT RECORDS
Which key signed this request?	Yes	Yes — unchanged
Which company is accountable for this agent, and what did it allow?	No	A commission , signed with the company's domain key and served from its own domain
Did the agent really make <i>this</i> request, within scope — and did anyone independent see it?	No	A record in an independent operator's append-only log, with a proof the platform checks itself

Without the second and third answers, every disputed agent action becomes one party's word against another's. The platform cannot tell whether the order was within the agent's authority; the company cannot prove what its agent did or did not send; an insurer or regulator has nothing independent to look at.

The future of agent verification is an independent record showing that a named company commissioned an agent, and that the agent made a specific request.

This document describes an open standard that provides that record, how it works end to end, and why it is safe to rely on. It is written for the people who will need to agree on it: platform engineers, companies running agents, agent builders, potential operators and participants in the IETF Web Bot Auth working group.

Three layers, each building on the one below

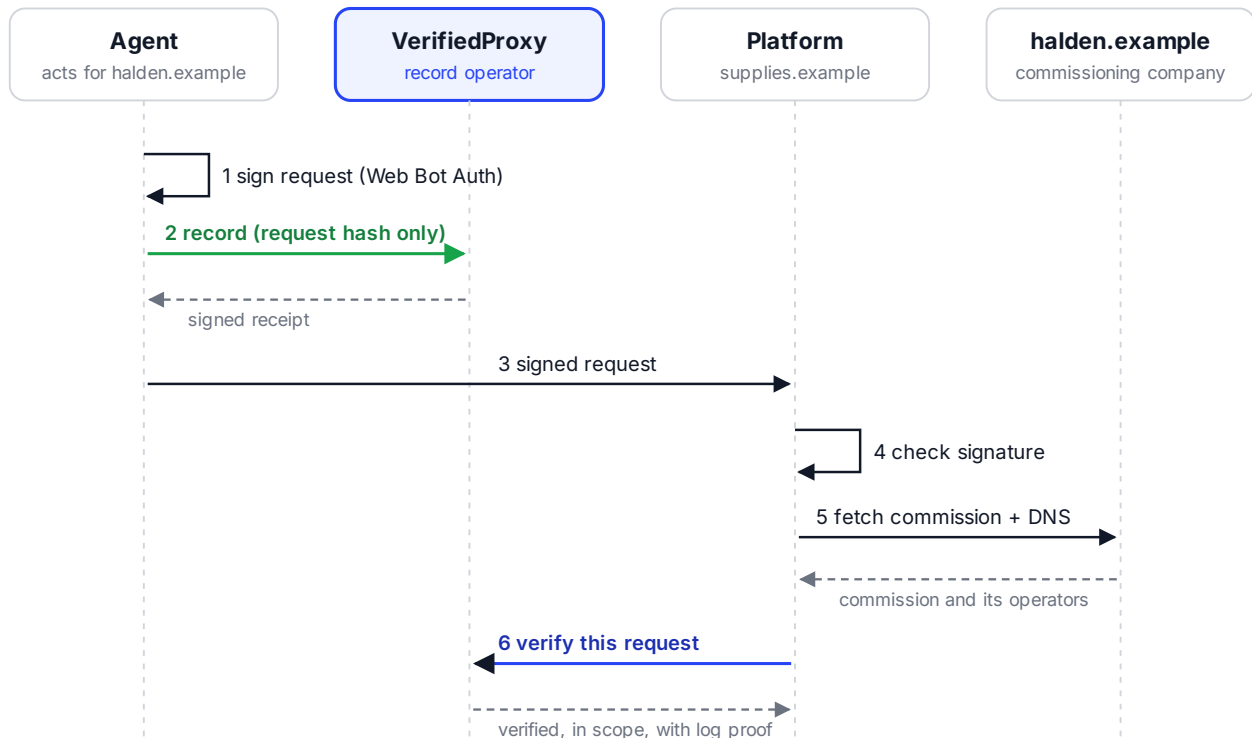
Each layer answers a question the layer beneath it cannot. Identity already exists; the commission and the record are what this standard adds.

Record New in this standard	QUESTION IT ANSWERS Did this agent really make this request, within scope?	PROOF An independent operator's signed entry in an append-only Merkle log, with an inclusion proof
Commission New in this standard	QUESTION IT ANSWERS Which company commissioned this agent, and for what?	PROOF A JSON document signed with the company's domain key and served from its own domain
Identity Exists today: Web Bot Auth	QUESTION IT ANSWERS Which key signed this request?	PROOF An HTTP message signature checked against the agent's published key

The layers are deliberately separable. A platform that checks only identity still benefits; one that also checks the commission knows who is accountable; one that checks the record has independent evidence of the specific request.

How one request is verified

Before any request, Halden publishes a **commission** on its own domain. It names the agent's public key, its scope (for example, purchase orders up to £10,000 at approved suppliers), an expiry date and the operator that holds its records.



The record is written before the request leaves. In proxy mode a VerifiedProxy proxy on the agent's own machine does steps 1–3; in direct mode the agent's SDK does them.

- 1 Signed request.** The agent signs its request with Web Bot Auth and includes a pointer to its commission *inside the signed headers*, so the pointer cannot be swapped.
- 2 Record.** Before the request leaves, the agent (or its proxy) writes a record with the operator — at the default level, a salted hash of the signed request, never its path or body. The operator refuses if the request is out of scope, the commission is revoked, or the nonce has been used before. A refused request is never sent.
- 3 Send.** The request reaches the platform carrying the agent's signature.
- 4 Identity check.** The platform verifies the signature against the agent's published key, exactly as it would for any Web Bot Auth agent.
- 5 Commission check.** The platform fetches the commission from halden.example and Halden's DNS record, confirms Halden signed it, that it names this agent's key, and that the request is within scope.
- 6 Record check.** The platform asks the operator named in Halden's DNS whether it holds a record of this exact request. It gets yes or no, the time, the scope and a proof from the log — and never sees Halden's other activity.

A company says, on its own domain: this agent works for us

A commission is a small JSON document, signed with the company's domain key as a JWS (EdDSA) and served from its own domain over HTTPS. Serving it from the domain proves control of the domain; the signature makes it portable evidence. Removing it from the domain withdraws it within five minutes, the longest a platform may cache it; revoking at the operator is instant.

```
{
  "v": "ACR1",
  "id": "cm_Xq7RmfB2kTzW4pLs9vNc1A",
  "commissioner": "halden.example",
  "agent_keys": ["kq3V...9xA"],           // JWK thumbprints = Web Bot Auth keyid
  "scope": {
    "sites": ["supplies.example"],
    "actions": ["create_order"],
    "limit": { "currency": "GBP", "per_order": 10000, "per_day": 25000 }
  },
  "issued_at": "2026-10-05T12:00:00Z",
  "valid_until": "2026-12-31T23:59:59Z",
  "record_operators": ["verifiedproxy.com"],
  "record_level": 2,                      // 1 destination · 2 commitment · 3 full content
  "outage_policy": "fail_closed"          // or "fail_open"
}
```

Discovery: one DNS record

A TXT record tells platforms which operators may vouch for the company and where its commission-signing keys live. An operator that is not listed cannot vouch for the company — which stops anyone building their own operator to impersonate it.

```
_agent-commission.halden.example. TXT "v=ACR1; ops=verifiedproxy.com;
keys=https://halden.example/.well-known/acr-keys.json"
```

Scope

- **sites** — hostnames the agent may use (subdomains included; `*` for any).
- **actions** — actions the agent may declare, such as `create_order`.
- **limit** — a currency, a per-order cap, and `per_day` and `total` caps the operator enforces across orders, so fifty orders just under the limit can't slip through.

Publication rules

- HTTPS on the company's domain, status 200, no redirects.
- One canonical URL per commission: no query, fragment or port.
- Revocation follows the commission (company + `id`), wherever it is published.
- IDs carry 128 random bits and are never listed, so commissions can't be found by guessing.

Keeping trading partners private. A commission's sites show who a company trades with, so companies issue one commission per platform: each platform sees only its own entry, and the agent presents whichever commission covers the request.

The company keeps its domain key. With VerifiedProxy, commissions are signed on the company's own machine; the operator never holds the key that makes a commission yours.

The signed request

A commissioned request is an ordinary Web Bot Auth request with two extra signed headers: `Agent-Commission`, pointing to the commission, and — when the request performs a commercial action — `Agent-Intent`, declaring the action and amount.

```
POST /orders HTTP/1.1
Host: supplies.example
Content-Digest: sha-256=:X48E9q0okqrvdts8n0JRJN30WDUoyWxBf7kbu9DBPE=:
Signature-Agent: "https://agents.halden.example"
Agent-Commission: "https://halden.example/.well-known/agent-commissions/cm_Xq7RmfB2kTzW4pLs9vNc1A"
Agent-Intent: action="create_order", amount=4200.0, currency="GBP"
Agent-Record-Salt: :3q2+7wAJr0yC4m8X1Tn6fV0bRK5hPzLs9QwE2uYdHcA=:
Signature-Input: sig1=("@method" "@authority" "@path" "@query" "content-digest"
    "signature-agent" "agent-commission" "agent-intent" "agent-record-
    salt");created=1791230400;expires=1791230460;
    keyid="kq3V...9xA";alg="ed25519";nonce="e8N7...qg=";tag="web-bot-auth"
Signature: sig1=:jdg0Sq0wHdyHr9+r5jw3iYZH6aNGKijYp/EstF4RQTQ...:
```

Because the site, method, path, query, body digest, commission and intent are all covered by the signature, none of them can be changed without breaking it. The same signed material is what gets recorded.

`Signature-Agent` points to Halden's own key directory, so the identity layer shows Halden — not whoever operates its records. The `Agent-Record-Salt` goes to the platform only, never to the operator (section 6).

Records and the log

Recording the actual action would normally mean decrypting traffic, which most companies will not accept from a third party. The standard solves this with three record levels, chosen per commission:

LEVEL	WHAT IS RECORDED	OPERATOR DECRYPTS?	BEST FOR
1 · Destination	Agent key, commission, site, time	No	Low-risk browsing and research
2 · Commitment (default)	Level 1 plus a hash of the signed request	No	Most commercial actions
3 · Full content	Level 2 plus the complete request, encrypted to the company's own key	No — only the company can read it	Regulated firms needing a full trail

Level 2: proving the exact request without receiving it

The Web Bot Auth signature already covers the site, method, path and, through a content digest, the body. The operator records a salted hash of that signed material:

```
request_hash = base64url( HMAC-SHA-256( key = Agent-Record-Salt, RFC 9421 signature base ) )
```

The platform recomputes the same hash when it checks the signature, so it can confirm the exact request. The operator never receives the method, path, query, headers or body, and because the salt goes only to the platform it can't test a guess at a predictable request. It does learn, at every level, the agent key, commission, site, time and any declared action and amount.

An append-only log, modelled on Certificate Transparency

- **Merkle tree.** Each record is a leaf in a Merkle tree hashed as RFC 9162 (Certificate Transparency 2.0). Records can be added, but not changed or removed without detection.
- **Signed tree heads.** The operator signs the tree's size and root hash. Every verification answer includes an inclusion proof against a signed tree head.
- **Receipts.** Writing a record returns a signed promise that it is in the log.
- **Independent time.** Tree heads are timestamped by an independent time-stamping authority (RFC 3161), so no record's time rests on the operator's clock alone.
- **Consistency proofs** show a later tree head extends an earlier one. Only tree heads and proofs are public.

When the operator is unreachable

Each commission sets an outage policy. **Fail closed** blocks requests until the operator is back; **fail open** lets them proceed and sends the record afterwards, marked *late*.

One API call, and a proof you check yourself

The platform sends the commission URL, key ID, nonce and request hash to the operator named in the company's DNS. The operator answers **verified**, **unknown**, **out of scope** or **revoked**, with the scope and a proof.

```
POST https://app.verifiedproxy.com/acr/v1/verify
{ "commission": "https://halden.example/.well-known/agent-commissions/cm_Xq7RmfB2kTzW4pLs9vNc1A",
  "key_id": "kq3V...9xA", "nonce": "e8N7...qg=", "request_hash": "Qm9...", "site": "supplies.example" }

→ { "status": "verified", "commission": { "commissioner": "halden.example", "scope": { ... } },
    "record": { "id": "rec_...", "level": 2, "verified_before": false },
    "proof": { "leaf": "...", "leaf_index": 1233, "audit_path": [ ... ] }, "tree_head": { "signed": "
<JWS>", ... } }
```

What a platform checks

- 1 **The request is for us.** The signed authority is a host this platform serves — a request signed for another site and relayed here is rejected.
- 2 **Identity.** The Web Bot Auth signature, its expiry and the body digest.
- 3 **Commission.** Served from the company's domain, signed with a key its DNS points to, naming this key, unexpired, and the request is in scope.
- 4 **Record and proof.** The operator's answer, then the proof: the tree head's signature, that the leaf names this commission, key, nonce, site and request hash, and that the leaf is included under the signed root.

Because the proof is checked against a signed, publicly monitored log, the platform does not have to take the operator's word for it.

ANSWER	TYPICAL REASONS
verified	This exact request was recorded under an active commission, within scope.
unknown	No record, a request that differs from the record, or a commission this operator doesn't know.
out of scope	Site, action, amount or currency outside what the company allowed.
revoked	Commission revoked, withdrawn or expired, or key revoked.

When to verify

Identity and commission checks are local and cached; the record check is a call to the operator. Requests that declare an intent or change state are checked **before acting**. Read-only requests may be accepted at once and their records checked afterwards, in batches or by sampling.

If an agent skips the record, verification fails — so skipping it only hurts the agent.

Caching and revocation

Platforms cache commissions, DNS and domain keys for **at most five minutes**. Revoking at the operator takes effect on the next request; removing a commission from the domain takes effect within five minutes.

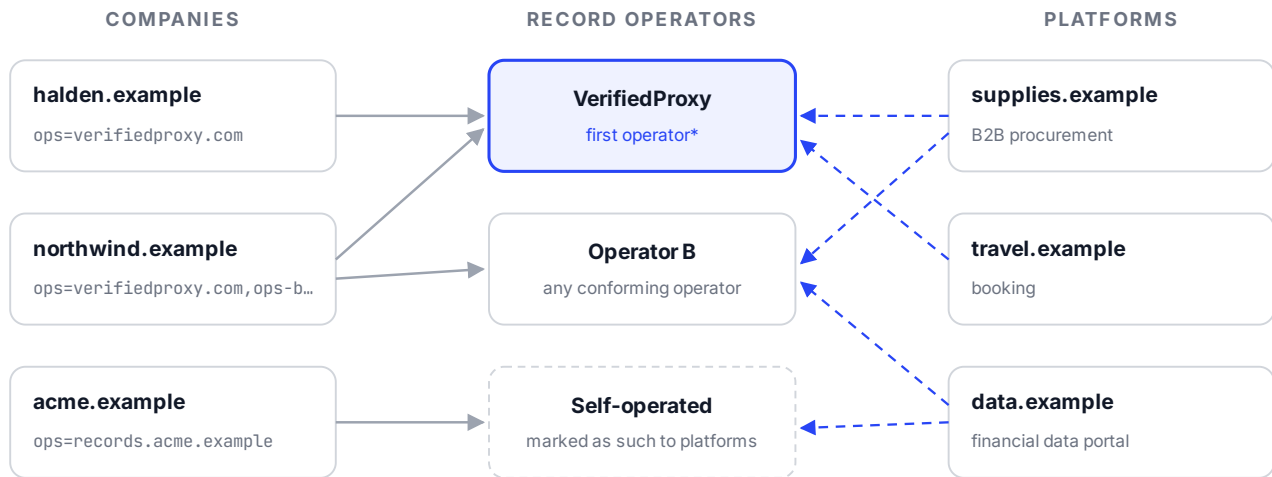
Why you can trust it

The design assumes every party might misbehave — including the operator — and makes each kind of misbehaviour either impossible or detectable.

THREAT	HOW THE STANDARD ANSWERS IT
A forged commission	A commission only counts if it is served from the company's domain and signed with a key the company publishes, found through its DNS.
A fake operator	Platforms only ask operators listed in the company's DNS <i>and</i> in the commission. Platforms can further limit themselves to certified operators.
The operator alters or backdates records	Signed tree heads, an inclusion proof with every answer, consistency proofs, independent RFC 3161 timestamps, and monitors — other operators, auditors, customers — watching for inconsistencies. Companies can export their records with proofs at any time.
A stolen key or replayed request	Keys are short-lived and rotated; every signature carries a nonce and an expiry; the operator rejects reused nonces; platforms are told when a record has already been verified; revocation is real time.
A provider key shared by many customers	A commissioned key must act for that company alone. Providers issue a key per customer, marked with <code>acr_principal</code> ; platforms reject it under any other company's commission.
Many orders just under the limit	Daily and lifetime caps, enforced by the operator in the same step as writing the record — concurrent orders can't overshoot.
Exposing who a company trades with	One commission per platform, IDs that can't be guessed, and no listings.
A request relayed to another platform	The site is part of the signature; a platform only accepts requests signed for a host it serves.
Someone claims another company's agent key	Operators require proof of possession — a signature by the key itself — before registering it.
The operator is the point of failure	Each commission chooses fail closed or fail open; late records are marked as such.
Personal data	Salted hashes rather than paths or bodies by default; Level 3 content is encrypted to the company's own key; leaf contents are never public; retention is configurable and deleting a record's contents leaves other proofs valid.

An open ecosystem of operators

The standard is free to implement, and none of it requires VerifiedProxy. DMARC is the precedent: an open email standard published in DNS, with a market of companies operating the monitoring and enforcement around it.



Grey: a company names its operators in DNS (ops=).

Blue: platforms discover them from DNS and verify.

* VerifiedProxy intends to be the first operator to pass the certified-operator programme, then hand the programme to a neutral foundation once other operators exist.

- **Platforms never sign up with an operator.** They read the company's DNS record, fetch <https://<operator>/.well-known/acr-operator.json> and call its verification API. A new operator works with every platform on day one.
- **Companies choose, and can switch.** A company can list more than one operator, and moving is a DNS change.
- **Anyone can run one.** The operator's job is fully specified: accept records, keep the log, answer verification requests, publish tree heads and consistency proofs.
- **Self-operated records are allowed**, and are marked as self-operated so platforms can weigh them accordingly.
- **Platforms can set their own bar** — for example, accepting only certified operators.

The question a platform asks is not “do they use VerifiedProxy?” but “which company is accountable, and is there an independent witness?” The operator is the witness.

Agents run by AI providers

Many companies won't run their own agents — they'll use one an AI provider runs in its own cloud. The standard has to work for them too.

A commission names agent keys, so it vouches for **everyone who can sign with those keys**. If a provider signed every customer's requests with one key, a company that commissioned it would be vouching for every one of the provider's customers. And a proxy on the company's premises can't sit inside the provider's cloud.

One key for every customer — not allowed

A key shared across customers can't be commissioned by any of them: it would make Halden accountable for requests made by everyone else using the same provider.

A key per customer — the standard

The provider gives each customer its own signing key and publishes it in its key directory, marked with the company it acts for. Halden commissions its own key.

```
// https://agents.provider.example/.well-known/http-message-signatures-directory
{ "keys": [
  { "kty": "OKP", "crv": "Ed25519", "x": "...", "kid": "Pz8...", "acr_principal": "halden.example" },
  { "kty": "OKP", "crv": "Ed25519", "x": "...", "kid": "Lm4...", "acr_principal": "northwind.example" }
] }
```

- **Both layers tell the truth.** The provider stays the `Signature-Agent`, so identity shows who runs the agent; the commission shows the company accountable for it.
- **Mistakes are caught.** When a key carries `acr_principal`, platforms reject it under any other company's commission.
- **The provider writes the records** before sending each request, using the same direct-mode interface as any other agent.
- **No change to Web Bot Auth.** Per-customer keys are ordinary Web Bot Auth keys.

For providers with very many customers, a single provider key plus a signed customer-account header is an alternative we would like to discuss (section 13).

Adopting it

Value arrives at every stage of adoption: platforms already verifying Web Bot Auth get identity with no new work; companies get an audit trail even before any platform checks; and each additional platform that checks records makes commissioned agents more welcome everywhere.

Companies running agents

- Publish a domain key and one DNS record.
- Sign a commission for each agent and publish it on your domain.
- Pick an operator. With VerifiedProxy: `domain init, commission create, commission register`.

Platforms

- Verify the Web Bot Auth signature as you may already do.
- Fetch the commission and check scope.
- Make one call to the operator and check the proof — or use the reference verifier, which does all of it.

Agent builders

- Sign requests with Web Bot Auth and add `Agent-Commission`.
- Declare commercial actions with `Agent-Intent`.
- Write the record before sending, and respect the operator's refusal.

Operators

- Implement the record and verification APIs and the RFC 9162 log.
- Publish operator metadata, tree heads and consistency proofs.
- Join the certified-operator programme.

Platform integration with the reference verifier

```
import { createVerifier, fromNodeRequest } from "verifiedproxy/verifier";

const verifier = createVerifier({ hosts: ["supplies.example"] });
const result = await verifier.verify(fromNodeRequest(req, body));

if (result.ok) {
  result.commission.commissioner // "halden.example"
  result.intent                  // { action: "create_order", amount: 4200, currency: "GBP" }
  result.record                  // { operator: "verifiedproxy.com", id, level, recordedAt, treeHead
}
}
```

Requests that perform a commercial action should carry an `Agent-Intent`; the platform checks the declared action and amount match what the request actually does — only the platform knows its own semantics.

VerifiedProxy, the first operator

VerifiedProxy wrote this draft and is building the first independent operator on it. The standard is free; operating it well is the business. Our implementation includes:

- **Record operator** for all three levels, with the RFC 9162 log, signed tree heads, receipts, consistency proofs, RFC 3161 checkpoints and exports with proofs.
- **Proxy mode**: a local proxy that signs every request, records it first and blocks anything out of scope. HTTPS is opened on the agent's own machine, so VerifiedProxy never decrypts traffic.
- **Direct mode**: an SDK for agents that call platforms themselves.
- **Reference verifier** for platforms, checking identity, commission and the log proof.
- **Governance console**: commissions, records, blocked requests and instant revocation.
- **Level 3 records** encrypted to the company's own key — readable by the company, never by us.

We aim to win on what an open market rewards: neutrality — we are neither the AI provider nor the platform — and evidence-grade records.

Status, open questions and how to take part

This is **Draft 0**, published to start a conversation. We intend to bring it to the IETF Web Bot Auth working group, and we want it shaped by the people who would use it — platforms, companies running agents, agent providers and would-be operators — before then. How to take part is on the last page.

Open questions we would value input on

- 1 **The name.** “Commissioned Agent Records”, `_agent-commission`, `Agent-Commission` and `ACR1` are working names, all on the wire.
- 2 **One-key provider delegation.** A single provider key plus a signed customer-account header, as an alternative to per-customer keys.
- 3 **People, not just companies.** A consumer profile: an agent provider commissions on a user's behalf, with a different pseudonymous ID per platform.
- 4 **Agent-Intent.** Whether declared actions belong here or in a separate standard, and whether actions need a shared vocabulary.
- 5 **Key directories.** Directory self-signatures for multi-tenant `Signature-Agent` hosts, and rotating short-lived keys without re-signing commissions.
- 6 **Monitoring.** Gossip of tree heads between monitors, and how platforms report inconsistencies.
- 7 **Levels.** Whether Level 2 stores the destination in clear (scope enforcement) or hashed (consumer privacy).
- 8 **Certification.** What certified operators must meet, and when the programme moves to a neutral foundation.

The specification (Draft 0)

Normative text. Draft 0, October 2026 — working name, editors VerifiedProxy, free to implement. The key words MUST, SHOULD and MAY are to be read as described in RFC 2119.

This document specifies how to prove **which company commissioned an AI agent** and **what that agent asked for**, on top of Web Bot Auth. The key words MUST, SHOULD and MAY are to be read as described in RFC 2119.

1. Roles and layers

ROLE	EXAMPLE	DOES
Agent	Halden's purchasing agent	Signs requests, writes a record before each one
Commissioner	halden.example	Publishes a signed commission on its own domain
Operator	verifiedproxy.com	Keeps records in an append-only log and answers verification requests
Platform	supplies.example	Receives the request and checks identity, commission and record

Each layer answers a question the one below it cannot:

LAYER	QUESTION	PROOF
Identity (Web Bot Auth)	Which key signed this request?	HTTP message signature checked against the agent's published key
Commission (§3)	Which company commissioned this agent, and for what?	A commission signed with the company's domain key
Record (§4–5)	Did this agent really make this request, within scope?	An operator's signed entry in an append-only log

The layers are separable. A platform that checks only identity still benefits.

2. Identity: the Web Bot Auth profile

ACR1 requests are signed with HTTP Message Signatures (RFC 9421) following Web Bot Auth:

- Algorithm Ed25519 (`alg="ed25519"`).
- `keyid` is the JWK SHA-256 thumbprint (RFC 7638) of the agent's public key.
- `tag="web-bot-auth"` .
- `created` and `expires` MUST be present; `expires - created` SHOULD NOT exceed 300 seconds.
- `nonce` MUST be present and MUST be unique per key. It SHOULD be 64 random bytes, base64-encoded.
- The `Signature-Agent` header (an sf-string `"https://..."`) names the origin whose `/.well-known/http-message-signatures-directory` lists the key. It SHOULD be an origin controlled by whoever runs the agent: the company itself (for example `https://agents.halden.example`), or the AI provider when the provider runs the agent (§3.8). A third party that hosts key directories for many companies becomes the visible identity of all of them, so it SHOULD be used only when the company cannot publish its own directory.
- A directory key MAY carry an `acr_principal` member naming the one domain the key acts for. When present, platforms MUST reject a commission whose `commissioner` differs (§3.8).

- Agents SHOULD send an `Agent-Record-Salt` header (§4.2) so the operator cannot test guesses against the request hash.

2.1 Covered components

An ACR1 signature MUST cover, in this order:

```
"@method" "@authority" "@path" "@query"
"content-digest"      (when the request has a body; RFC 9530, sha-256)
"signature-agent"
"agent-commission"
"agent-intent"        (when present)
"agent-record-salt"   (when present)
```

Covering `agent-commission` means the pointer to the commission cannot be swapped; covering `agent-intent` binds the declared action and amount.

2.2 Example

```
POST /orders HTTP/1.1
Host: supplies.example
Content-Type: application/json
Content-Digest: sha-256=:X48E9q0okqqrvdts8n0JRJN30WDUoyWxBf7kbu9DBPE=:
Signature-Agent: "https://agents.halden.example"
Agent-Commission: "https://halden.example/.well-known/agent-commissions/cm_Xq7RmfB2kTzW4pLs9vNc1A"
Agent-Intent: action="create_order", amount=4200.0, currency="GBP"
Agent-Record-Salt: :3q2+7wAJr0yC4m8X1Tn6fV0bRk5hPzLs9QwE2uYdHcA=:
Signature-Input: sig1=("@method" "@authority" "@path" "@query" "content-digest" "signature-agent"
"agent-commission" "agent-intent" "agent-record-
salt");created=1791230400;expires=1791230460;keyid="kq3V...9xA";alg="ed25519";nonce="e8N7...
qq=";tag="web-bot-auth"
Signature: sig1=:jddq0Sq0wHdyHr9+r5jw3iYZH6aNGKijYp/EstF4RQTQ...:
```

3. The commission

3.1 Document

A commission is a JSON object:

FIELD	REQUIRED	MEANING
<code>v</code>	yes	"ACR1"
<code>id</code>	yes	Unique within the commissioner and unguessable: at least 128 random bits, written as 22–128 characters from <code>A-Z a-z 0-9 . _ ~ -</code> (§3.7)
<code>commissioner</code>	yes	The company's domain, lower case
<code>agent_keys</code>	yes	JWK thumbprints of the agent keys commissioned
<code>scope</code>	yes	What the agent may do (§3.5)
<code>issued_at</code>	yes	RFC 3339 time
<code>valid_until</code>	yes	RFC 3339 time after which the commission is void
<code>record_operators</code>	yes	Operator IDs that may hold records for it
<code>record_level</code>	no	1, 2 or 3 (§4.1). Default 2
<code>outage_policy</code>	no	"fail_closed" (default) or "fail_open" (§4.6)
<code>content_key</code>	Level 3	X25519 public JWK that Level 3 content is encrypted to

```
{
  "v": "ACR1",
  "id": "cm_Xq7RmfB2kTzW4pLs9vNc1A",
  "commissioner": "halden.example",
  "agent_keys": ["kq3V...9xA"],
  "scope": {
    "sites": ["supplies.example"],
    "actions": ["create_order"],
    "limit": { "currency": "GBP", "per_order": 10000, "per_day": 25000 }
  },
  "issued_at": "2026-10-05T12:00:00Z",
  "valid_until": "2026-12-31T23:59:59Z",
  "record_operators": ["verifiedproxy.com"],
  "record_level": 2,
  "outage_policy": "fail_closed"
}
```

3.2 Signing

The commission is a compact JWS (RFC 7515) with protected header `{"alg":"EdDSA","kid":"<domain key thumbprint>","typ":"acr-commission+jws"}`. EdDSA (Ed25519) is the only permitted algorithm.

3.3 Publication

The commission MUST be served over HTTPS from the commissioner's domain (or a subdomain of it), with status 200 and no redirects, at a URL of the company's choosing. `https://<domain>/well-known/agent-commissions/<id>` is RECOMMENDED. It SHOULD be served as `application/jose`.

The URL MUST be in canonical form, so that one commission has exactly one URL: `https://`, a lower-case host with no trailing dot, no port, userinfo, query or fragment, and a path of `A-Z a-z 0-9 . _ ~ - /` with no empty or dot segments. Operators and platforms MUST reject any other form. Revocation applies to the commission (`commissioner` + `id`), wherever it is published.

Serving it proves control of the domain; the signature makes it portable evidence. **Removing it from the domain withdraws it** — within the cache limits of §5.3. Revocation that must take effect immediately goes through the operator (§6.3).

Commission URLs are capability URLs: only platforms that receive a request learn them. Servers **MUST NOT** list commissions or make them discoverable, and **SHOULD** send `X-Robots-Tag: noindex`.

3.4 Discovery

A DNS TXT record at `_agent-commission.<domain>`:

```
_agent-commission.halden.example.  TXT  "v=ACR1; ops=verifiedproxy.com;
keys=https://halden.example/.well-known/acr-keys.json"
```

- `v=ACR1` — required, first.
- `ops=` — comma-separated operator IDs that may vouch for this domain.
- `keys=` — HTTPS URL, on the domain or a subdomain, of a JWKS holding the Ed25519 keys that sign commissions.

An operator not listed in `ops=` cannot vouch for the company, even if a commission names it. Platforms **MUST** only query operators that appear in both `ops=` and the commission's `record_operators`.

3.5 Scope

- `sites` — hostnames. A request is in scope if its host equals an entry or is a subdomain of it. `"*"` allows any site. IP literals match exactly.
- `actions` — optional list of action names. When present, a declared `Agent-Intent` action **MUST** be in the list (`"*"` allows any).
- `limit` — optional `{currency, per_order, per_day, total}` with `currency` and at least one amount. A declared currency **MUST** equal `currency` (a declared amount without a currency is in `currency`).
 - `per_order` — no single declared amount may exceed it.
 - `per_day` — declared amounts recorded under this commission in any 24-hour period may not exceed it.
 - `total` — declared amounts recorded under this commission over its life may not exceed it.

The operator enforces `per_day` and `total` when it writes records, because only it sees every amount; a recorded request was within the caps when it was recorded. A commission with `per_day` or `total` **MUST** use `outage_policy: "fail_closed"`, so that no unrecorded spending can happen.

Requests that declare no intent (for example reading pages) are checked against `sites` only. A platform **MUST** require an `Agent-Intent` on any request it treats as one of its commercial actions, and **SHOULD** check that the declared action matches what the request does — only the platform knows its own semantics.

3.6 Headers

- `Agent-Commission` — sf-string: the commission's URL.
- `Agent-Intent` — sf-dictionary: `action` (string, required), `amount` (non-negative integer or decimal), `currency` (ISO 4217 string).
- `Agent-Record-Salt` — sf-binary: 32 random bytes, new for every request (§4.2).

3.7 Keeping trading relationships private

A commission's `sites` reveal who a company trades with, and anyone who receives a request learns its commission URL.

- Companies SHOULD issue a **separate commission for each platform** (or for each group of platforms that may see each other), so a platform learns only its own entry. An agent MAY hold many commissions and MUST present the one whose scope covers the request.
- Commission IDs MUST be unguessable (§3.1), so commissions cannot be found by enumerating URLs.
- The DNS record and `keys=` document reveal only the chosen operators and the company's commission-signing keys.

3.8 Agents run by AI providers

A commission names agent keys, so it vouches for **everyone who can sign with those keys**. When a company runs its own agent this is the company. When an AI provider runs the agent in its own cloud, a key the provider uses for all of its customers would vouch for every one of them — a company MUST NOT commission such a key.

- A key named in `agent_keys` MUST act for the commissioner alone.
- A provider that runs agents for many companies MUST give each company its own signing key (for example, one per customer workspace) and publish it in the provider's key directory with `acr_principal` set to that company's domain. The provider remains the `Signature-Agent`: the identity layer shows the provider, and the commission shows the company.
- The provider writes the records (direct mode, §4.4), because a proxy on the company's premises cannot sit inside the provider's cloud. How the provider is authorised to write records for the company's commission is operator-specific.
- Platforms MUST check `acr_principal` when it is present (§2).

Per-customer keys need no change to Web Bot Auth. Delegation with a single provider key and a signed customer-account header is an open question (§9).

4. Records

4.1 Levels

Levels are cumulative. The commission's `record_level` is the minimum an agent MUST record; it MAY record at a higher level.

LEVEL	RECORDED	OPERATOR DECRYPTS?	FOR
1. Destination	Agent key, commission, site, declared intent, time, nonce	No	Low-risk browsing, research
2. Commitment (default)	Level 1 + the request hash (§4.2)	No	Most commercial actions
3. Full content	Level 2 + the complete request encrypted to <code>content_key</code>	No — only the company can read it	Regulated firms

At every level the operator learns the agent key, the commission, the site, the time and any declared action and amount. At Levels 1 and 2 it never receives the method, path, query, headers or body.

4.2 Request hash

```
# with Agent-Record-Salt (recommended)
request_hash = base64url( HMAC-SHA-256( key = salt, message = signature base ) )

# without it
request_hash = base64url( SHA-256( signature base ) )
```

The signature base is the RFC 9421 signature base of the ACR1 signature (§2.1), which covers the site, method, path, query, body digest, commission and intent. The platform recomputes it when it checks the signature, so it can confirm the exact request without the operator receiving the path or body.

The salt is sent to the platform in `Agent-Record-Salt` and never to the operator. Without it, an operator that can guess a predictable request (a fixed path and a small body) could test the guess against the hash. Agents SHOULD send a salt; platforms MUST use it when it is present and covered by the signature.

4.3 Level 3 content

The agent serialises the request as `{"method","url","headers","body"(base64)}` and encrypts it as a compact JWE (RFC 7516) with `alg: "ECDH-ES"`, `enc: "A256GCM"` to the commission's `content_key`. The operator stores the ciphertext; the record leaf carries `content_hash = base64url(SHA-256(JWE))`.

4.4 Writing a record

The agent (or its proxy) sends the record to an operator **before** the request leaves. Authentication to the operator is operator-specific.

```
POST {api}/records
{
  "commission": "https://halden.example/.well-known/agent-commissions/cm_Xq7RmfB2kTzW4pLs9vNc1A",
  "key_id": "kq3V...9xA",
  "nonce": "e8N7...qg=",
  "level": 2,
  "request_hash": "...",
  "site": "supplies.example",
  "intent": {"action": "create_order", "amount": 4200, "currency": "GBP"},
  "signature_created": 1791230400,
  "late": false,
  "content": "..." (Level 3 only)
}
```

The operator MUST refuse (and the request MUST NOT be sent in proxy mode) when:

CODE	CONDITION
<code>commission_revoked</code> , <code>commission_withdrawn</code> , <code>commission_expired</code>	The commission cannot be used
<code>key_not_in_commission</code>	<code>key_id</code> is not in <code>agent_keys</code>
<code>key_revoked</code>	The key is revoked or expired
<code>level_too_low</code>	<code>level</code> < the commission's <code>record_level</code>
<code>out_of_scope</code>	<code>site</code> or <code>intent</code> is outside the scope, including <code>per_day</code> and <code>total</code> caps (reasons <code>over_daily_limit</code> , <code>over_total_limit</code>)
<code>replayed_nonce</code>	This key has used this nonce before
<code>late_not_allowed</code>	<code>late</code> is set but the commission fails closed

Before first use the operator MUST fetch and validate the commission as a platform would (§3.2–3.4), including that its own ID is listed in both `ops=` and `record_operators`. While a commission is in use the operator MUST re-read it at least every 300 seconds, so narrowed scopes and withdrawals take effect.

The operator MUST check `per_day` and `total` caps and write the record as one atomic step, so concurrent requests cannot together exceed a cap.

On success the operator returns the record leaf, its index and a **receipt**: a JWS (`typ: "acr-receipt+jws"`) over `{v, operator, record, leaf_index, leaf_hash, recorded_at}` — a signed promise that the record is in the log.

4.5 The record leaf and the log

The log leaf is the UTF-8 JSON record as the operator wrote it:

```
{
  "v": "ACR1",
  "id": "rec_...",
  "operator": "verifiedproxy.com",
  "commission": "https://.../cm_Xq7RmfB2kTzW4pLs9vNc1A",
  "key_id": "kq3V...9xA",
  "nonce": "e8N7...",
  "level": 2,
  "request_hash": "...",
  "site": "supplies.example",
  "intent": {
    "action": "create_order",
    "amount": 4200,
    "currency": "GBP"
  },
  "signature_created": 1791230400,
  "recorded_at": "2026-10-05T12:00:01Z",
  "late": false,
  "content_hash": null
}
```

Leaves are appended to a Merkle tree hashed exactly as RFC 9162 §2.1 (Certificate Transparency 2.0): `leaf_hash = SHA-256(0x00 || leaf)`, `node = SHA-256(0x01 || left || right)`.

Signed tree heads are JWS (`typ: "acr-tree-head+jws"`) over:

```
{
  "v": "ACR1",
  "operator": "verifiedproxy.com",
  "log_id": "...",
  "tree_size": 1234,
  "root_hash": "...",
  "timestamp": "2026-10-05T12:00:02Z"
}
```

`log_id` is base64url SHA-256 of the operator's public key. Operators SHOULD publish timestamped checkpoints at least every minute while the log grows: the tree head payload is timestamped by an independent RFC 3161 authority and the token published alongside it.

Leaf contents are private to the commissioner and to platforms that present the matching nonce; only tree heads and proofs are public.

4.6 Outage policy

- `fail_closed` — if the operator cannot be reached the request is not sent.
- `fail_open` — the request proceeds; the record is sent when the operator returns, with `late: true`. The operator records it as late. A platform verifying in the meantime gets `unknown` and MAY retry.

5. Verification

5.1 Platform procedure

- Identity.** Check the request's authority is a host this platform serves (a request signed for another site may have been relayed). Verify the Web Bot Auth signature (§2) with the key from the `Signature-Agent` directory. Check `created` / `expires` and the content digest.
- Commission.** Check the signature covers the components in §2.1. Fetch the `Agent-Commission` URL (HTTPS, no redirects); its host MUST be the `commissioner` domain or a subdomain. Read `_agent-commission.<domain>`, fetch `keys=`, verify the JWS. Check `keyid` is in `agent_keys`, that the directory key's `acr_principal` (if any) is the commissioner, that `valid_until` is in the future, and that the request's host and declared intent are in scope.
- Record.** For an operator in both `ops=` and `record_operators`, fetch `https://<operator>/.well-known/acr-operator.json` and call its verify API (§5.2) with the commission URL, key ID, nonce and request hash (salted when `Agent-Record-Salt` is present). §5.4 says when this may be deferred.

4. **Proof.** Verify the tree head's JWS with the operator's key; decode the leaf and check it names this operator, commission, key, nonce, site and (for Level ≥ 2) request hash; check the inclusion proof against the tree head's root. The platform need not take the operator's word for anything.

5.2 Operator verify API (step 3)

```
POST {api}/verify
{"commission": "...", "key_id": "...", "nonce": "...", "request_hash": "...", "site": "supplies.example",
"intent": {...}}
```

Response:

```
{
  "v": "ACR1",
  "status": "verified",
  "reason": null,
  "operator": "verifiedproxy.com",
  "commission": {"url": "...", "commissioner": "halden.example", "scope": {...}, "valid_until": "...",
"record_level": 2, "status": "active"},
  "record": {"id": "rec_...", "level": 2, "recorded_at": "...", "late": false, "verified_before": false},
  "proof": {"leaf": "<base64url leaf>", "leaf_index": 1233, "tree_size": 1234, "audit_path": [...]}},
  "tree_head": {"tree_size": 1234, "root_hash": "...", "timestamp": "...", "signed": "<JWS>", "tsa_token":
null}
}
```

STATUS	REASON
verified	—
unknown	commission_unknown, key_not_in_commission, no_record, commission_mismatch, record_purged, request_mismatch, site_mismatch, intent_mismatch
out_of_scope	site_not_in_scope, action_not_in_scope, over_limit, currency_not_in_scope
revoked	commission_revoked, commission_withdrawn, commission_expired, key_revoked

Verification is real time: revoking a commission takes effect on the next request. `verified_before` is true when the record has already been verified; platforms SHOULD treat that as a possible replay. When the company lists several operators, a platform SHOULD ask the next one after an `unknown` answer; any other answer is final.

5.3 Caching and revocation

- Platforms MAY cache a commission, the `_agent-commission` record and the `keys=` document for **at most 300 seconds**, and less when HTTP caching headers say so. Key directories follow Web Bot Auth, for at most 3600 seconds.
- Withdrawing a commission by removing it from the domain therefore takes effect at platforms within 300 seconds.
- **Instant revocation goes through the operator.** An operator revocation applies to the next record and the next verification. Platforms MUST NOT reuse a record check for another request.

5.4 When to verify

Identity and commission checks are local and cacheable; the record check is a network call to the operator.

- A platform MUST complete all three checks **before acting** on a request that declares an `Agent-Intent`, or that it treats as a commercial or state-changing action (orders, bookings, payments, account changes).

- For requests that only read (safe methods, no intent), a platform MAY accept on identity and a cached commission, and check the record afterwards — asynchronously, in batches or by sampling. A deferred check that fails SHOULD lead the platform to stop trusting that key.

6. Operators

6.1 Metadata

`https://<operator-id>/well-known/acr-operator.json`:

```
{
  "v": "ACR1",
  "operator": "verifiedproxy.com",
  "api": "https://app.verifiedproxy.com/acr/v1",
  "log_id": "...",
  "keys": [
    {
      "kty": "OKP",
      "crv": "Ed25519",
      "x": "...",
      "kid": "..."
    }
  ],
  "record_levels": [1, 2, 3],
  "timestamping": "rfc3161"
}
```

`api` MUST be an HTTPS URL on the operator's own domain (or a subdomain).

6.2 Public log endpoints

- `GET {api}/tree-head` — latest signed tree head; `?size=N` for a past one.
- `GET {api}/consistency?first=M&second=N` — RFC 9162 consistency proof between two published sizes.

Independent monitors (other operators, auditors, customers) fetch tree heads, check consistency proofs and compare what they and platforms have seen.

6.3 Commissioner access

Operators MUST let commissioners export their records with inclusion proofs at any time, and MUST support revocation that takes effect on the next record or verification — the only way to revoke faster than platform caches expire (§5.3).

6.4 Self-operated records

A company MAY run its own operator. When the operator ID is the commissioner's domain or a subdomain, its records are **self-operated** and platforms SHOULD weigh them accordingly. Platforms MAY restrict themselves to certified operators.

7. Security considerations

- **Forged commissions.** A commission counts only when served from the company's domain and signed with a key that domain publishes in DNS-linked `keys=`.
- **Fake operators.** Platforms query only operators listed in the company's DNS.
- **Shared keys.** A key that signs for several principals cannot be commissioned by any of them (§3.8); `acr_principal` lets platforms catch mistakes.
- **Stolen keys and replay.** Agent keys SHOULD be short-lived and rotated; signatures expire within minutes; operators reject reused nonces; platforms see `verified_before`. Revocation is effective on the next request.
- **Operator tampering.** Tree heads are signed, timestamped by an independent authority and checked for consistency by monitors; every verification carries an inclusion proof.
- **Fetching untrusted URLs.** Operators fetch URLs supplied by customers, and platforms fetch URLs supplied by agents. Both MUST restrict them to HTTPS on the default port, refuse private, loopback and reserved addresses (including IPv4-mapped IPv6), not follow redirects, and cap response sizes.
- **Commission enumeration.** Unguessable IDs and no listings (§3.3, §3.7).

- **Overspending.** Per-order limits alone allow many orders just under the limit; `per_day` and `total` caps are enforced by the operator (§3.5).
 - **Key registration.** How an agent registers keys with an operator is operator-specific, but operators MUST require proof of possession (a signature by the key) so that nobody can claim another party's public key.
-

8. Privacy considerations

- The operator learns the agent key, commission, site, time and declared intent of every recorded request (§4.1). At Levels 1 and 2 it never receives the method, path, query, headers or body; with `Agent-Record-Salt` it cannot test guesses against the request hash. Level 3 content is encrypted to the company's own key.
 - Commissions reveal trading relationships; per-platform commissions and unguessable IDs limit who learns them (§3.7).
 - A platform learns only about the request it received; leaf contents are never public.
 - Operators SHOULD offer configurable retention. Deleting a record's contents leaves its leaf hash in the tree, so other proofs remain valid.
-

9. Not yet specified

- **Provider delegation with one key:** a single provider signing key plus a signed header naming the customer account, as an alternative to per-customer keys (§3.8) for providers with very many customers.
- **Consumer profile** (paper §5.10): an agent provider commissioning on a user's behalf, with per-platform pseudonymous user IDs.
- **Company-wide caps** across several commissions.
- Directory self-signatures (`tag="http-message-signatures-directory"`) for multi-tenant `Signature-Agent` hosts.
- Tree head gossip between monitors, and the certified-operator programme.

References

Web Bot Auth	IETF Web Bot Auth working group (webbotauth) — architecture and HTTP Message Signatures directory drafts
RFC 9421	HTTP Message Signatures
RFC 9530	Digest Fields (Content-Digest)
RFC 9651	Structured Field Values for HTTP
RFC 9162	Certificate Transparency Version 2.0 (Merkle tree hashing and proofs)
RFC 3161	Internet X.509 PKI Time-Stamp Protocol
RFC 7515 / 7516	JSON Web Signature / JSON Web Encryption
RFC 7518	JSON Web Algorithms (ECDH-ES, A256GCM)
RFC 7638 / 8037	JWK Thumbprint / CFRG curves (Ed25519, X25519) in JOSE
RFC 7489	DMARC — the precedent for an open standard published in DNS with a market of operators
RFC 2119	Key words for use in RFCs to indicate requirement levels

Take part

Read the specification, try it against your own platform or agents, and tell us what breaks, what is missing and what you would change. We want to hear from platforms, companies running agents, agent providers and anyone who would run an operator. Contact us at **verifiedproxy.com**. We intend to publish the reference verifier and agent SDK alongside the draft.



VerifiedProxy

The first independent record operator for Commissioned Agent Records · verifiedproxy.com