JACS:
JSON Agent Communication Standard

hai.ai

AI Alliance, June 12, 2025

jonathan@hai.io
https://www.linkedin.com/in/jonathanhendler/

About me

Small, early-stage startup backend generalist. Co-founded an ed-tech company. Eng at mental health care, Air Quality IoT.

Read papers on neural nets in 2001 while backpacking in Asia, before I went to college for CS. Hacked Semantic Web, NLP, and deep learning trying create structured data from the web before LLMs/Transformers.

https://www.linkedin.com/in/jonathanhendler/

Agents access data and services the same way a human can.

Creative, unpredictable, potentially malicious.

Agents create new information from data they have accessed. As observers we don’t know if a human was in the loop or what other systems changed the data.

We secure APIs, email, databases and valuable web content through many disparate, and uncoordinated auth systems. Each can be difficult to set up for small developers or for in complex heterogeneous data environments.

hai.ai

JACS for data provenance

1. A file is sitting on a server. Where did it come from? Who had access to it? (e.g. email, shared docs)

2. An MCP server gets a request from an unknown agent, the oauth flow doesn't guarantee the identity of the client after the initial handshake.

3. A document is modified by multiple human and AI collaborators. Which one is latest, correct version? Did the collaborators agree?

hai.ai

What is JACS?

JACS is a set of JSON Schema definitions that provide headers for cryptographic signatures.

- **Embeddable library** in multiple programming languages to sign JSON or any binary files by attaching a JACS header - Rust lib used in Python and Typescript
- JSON Schema https://json-schema.org/ - used in JSON RPC (MPC) and OpenAI tools, and A2A Agent definitions.
- Hashing and signing libraries: supports RSA, Ring ED25519 and (experimental) post quantum via dilithium
- Observability with Open Telemetry

JACS Origin Story

Had an email project using AI for my startup two years ago and found shortcomings validating content across systems.   DNS/SPF, PGP, left gaps for other use cases like an email chain, multiple recipients, attachments.

Also, the web's *agents.txt* have little ability to identify and manage clients. For identity ARC is email only. DKIM is domain only.

With chat taking over, email seemed less important overall. I needed something new.

Current Technology used

- OAuth 2/OpenID
- JWT
- PGP
- DKIM, SPF
- DPKI and blockchain
- W3C DID
- SAML
- mTLS, TLS, x509
- Kerberos
- FIDO2/WebAuthn/Passkeys
- ARC
- DPoP
- Checksums (RFC 6249., RFC 2068, RFC 3230, new HTTP Digest-Headers)

JACS for Auth

Making it easy to set up trusted identity

- Works in both the MCP server and MCP client
- Every request is verified in source identity and content
- Business logic can be built around decentralized identity

hai.ai

Roadmap

1. PKI and KeyServer infra integration
2. Integration with A2A, physical devices
3. RBAC middleware for http
4. RBAC for Data Lakes, Filesystems, and Databases
5. full data lake integration

hai.ai

What JACS needs today

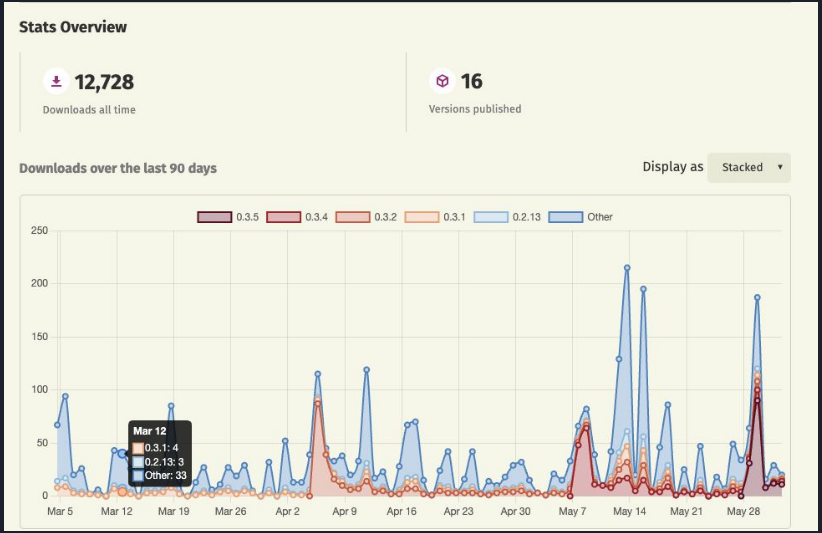
1. Users - integrations, use cases
2. Contributors - extensions, modularity
3. Strategic Partners - paths to adoption

hai.ai

Open Source

https://github.com/HumanAssisted/JACS

https://crates.io/crates/jacs



Stats Overview

12,728
Downloads all time

16
Versions published

Downloads over the last 90 days

Display as Stacked

Verify this document

View the JACS header file: https://raw.githubusercontent.com/HumanAssisted/JACS/refs/heads/main/jacs/docs/presentation/jacs/aialliance.presentation.jacs.json

Download document: https://raw.githubusercontent.com/HumanAssisted/JACS/refs/heads/main/jacs/docs/presentation/aialliance.presentation.pdf

Download public key: https://raw.githubusercontent.com/HumanAssisted/JACS/refs/heads/main/jacs/docs/presentation/jacs_keys/aialliance.presentation.key.pub

JACS header was created with:

```
$ JACS_PRIVATE_KEY_PASSWORD=hello jacs document create -v --attach aialliance.presentation.pdf -e false -o
aialliance.presentation.jacs.json -a
jacs/agent/08a79c8b-464c-41fb-b071-937e6543871d\:ba3cf18d-60f1-4fc7-bf0a-d1ec517ccbbe.json
```

From the jacs cli, verify with:

```
$JACS_PRIVATE_KEY_PASSWORD=hello jacs document verify -v -f jacs/aialliance.presentation.jacs.json -a
jacs/agent/[your agent id].json
```

hai.ai

License

Apache 2.0 - with Common Clause. *(Considering pure Apache 2.0)*

https://commonsclause.com/

hai.ai

Easy MCP and Web Auth

1. Rust Lib `$ cargo install jacs`
2. CLI `$ jacs init`
3. Python `$ pip install jacs`
4. Node `$ npm install jacs`

Python MCP Server

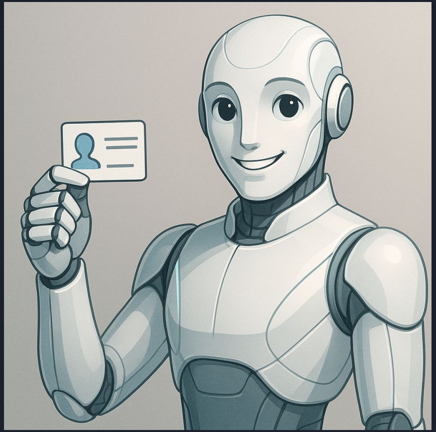
```
jacs_config_path = current_dir / "jacs.server.config.json"
os.environ["JACS_PRIVATE_KEY_PASSWORD"] = "hello"
jacs.load(str(jacs_config_path))
mcp = JACSMCPServer(FastMCP("Authenticated Echo Server"))
```

Python MCP Client

```
client = JACSMCPClient(server_url)
```

hai.ai

Give an Agent an ID ™



hai.ai