theailab.org

STANDARD | VERSION 1.2

TIP for AI Agents

Verified Agent Identity, Delegated Authority and Verifiable Actions on the Trust Identity Protocol

AUTHOR
Dinesh Mendhe, Founder and Chairman
ORCID 0000-0003-1158-3259

PUBLISHER
The AI Lab Intelligence Unobscured, Inc.
Wilmington, Delaware, United States

DATE
November 13, 2026

DOI
10.5281/zenodo.24211833

LICENSE
Creative Commons Attribution 4.0 International

STATUS
Agent profile for deployment; ledger changes by RFC

theailab.org

Contents

Suggested Citation

Foreword

Acknowledgments

Executive Summary

Key terms

Part I. The Problem

1. The problem

2. Current issues, 2025 and 2026

3. Use cases

Part II. Landscape and Design Principles

4. What exists today, and the gap

5. Design goals and non-goals

Part III. Architecture and Agent Identity

6. Architecture overview

7. Agent identity

Part IV. Delegated Authority and Human Control

8. Mandates: delegated authority

9. TIP Approval for high-risk actions

Part V. Actions, Verification and Evidence

10. Action requests

11. Verification at the receiving site

12. Receipts, logs and anchoring

Part VI. Revocation, Safety and Agent-Made Content

13. Revocation and TIP Stop All

14. Agent-made content and Verifiable Assets

Part VII. Inputs, Accountability and Security Analysis

15. Protecting the agent's inputs

16. Accountability and reputation

17. Threat model

Part VIII. Interoperability, Deployment and Scale

theailab.org

18. Interoperability

19. Scalability and performance

Part IX. Privacy, Regulation and Economics

20. Privacy and data protection

21. Regulation and sector requirements

22. Economics and lookup costs

Part X. Adoption

23. Changes needed in TIP

24. Roadmap

25. What this design does not do

26. Open questions for The AI Lab and the AI Trust Council

Appendices

Appendix A. Example records

Appendix B. Verifier outline

Appendix C. How this paper builds on pull request 383

References

theailab.org

Suggested Citation

APA. Mendhe, D. (2026). *TIP for AI Agents: Verified agent identity, delegated authority and verifiable actions on the Trust Identity Protocol* (Version 1.2). The AI Lab Intelligence Unobscured, Inc. https://doi.org/10.5281/zenodo.24211833

Chicago. Mendhe, Dinesh. 2026. *TIP for AI Agents: Verified Agent Identity, Delegated Authority and Verifiable Actions on the Trust Identity Protocol*. Version 1.2. Wilmington, DE: The AI Lab Intelligence Unobscured, Inc. https://doi.org/10.5281/zenodo.24211833.

License Summary

This whitepaper is licensed under the Creative Commons Attribution 4.0 International Public License (CC BY 4.0). Implementations of the technical framework described in it are licensed under the TIP Protocol Code License Version 1.0 (TIPCL-1.0), published at theailab.org/tip-license.

Document Status

Version 1.2, published November 13, 2026. It replaces Version 1.1 (https://doi.org/10.5281/zenodo.24208937) and Version 1.0 (https://doi.org/10.5281/zenodo.24097550), which stay available. Version 1.2 changes how the design reaches use, and nothing in the design itself. The parts that live off the TIP ledger form the TIP Agent Profile. These are mandates, TIP Approvals, Action Requests, receipts and site logs, with their schemas, registries and parameters. The AI Lab publishes the profile and deploys it now as its own service, without the RFC process. The TIP Protocol Specification v5.0 allows this. Its Section 49.2 says the AI Trust Council does not govern The AI Lab's own corporate decisions, and the RFC list in its Section 50.1 covers the ledger, its transaction types, its node API and its cryptography, none of which the profile changes. The parts that change the ledger still go through the RFC process, because every node must run them. These are public agent records, ledger revocation and epochs, status signers, witnesses, anchoring and the agent conduct rules in adjudication. The Foreword, the Acknowledgments and Sections 3.14, 6.5, 8.3, 8.6, 9, 10.6, 19.4, 20, 23, 23.1 and 24 now say so. Version 1.1 added the evidence-keeping rule (Section 12.8), the basis digest (Section 10.6) and the hosted services terms (Section 20, item 10), and corrected Version 1.0 in five places.

This whitepaper extends the Trust Identity Protocol to the actions of AI agents. The TIP Agent Profile, the off-ledger parts listed in Section 23, is published by The AI Lab for deployment now. The new transaction types, the agent identifier format and the other ledger changes are proposals. They become part of TIP only after the RFC process in Section 50 of the TIP Protocol Specification v5.0, with its thirty-day public comment period and ninety-day testnet deployment, and adoption by the AI Trust Council. The paper builds on the TIP Protocol Whitepaper, Version 1.1

(https://doi.org/10.5281/zenodo.22134504), and on the TIP Protocol Specification v5.0. Errata may be reported to whitepaper-errata@theailab.org. Questions about the protocol go to tip@theailab.org.

Distribution List

Public.

theailab.org

Foreword

By Dinesh Mendhe, Founder and Chairman of The AI Lab Intelligence Unobscured, Inc.

TIP began with a simple test for content. Who made this, how, and has it changed since? Software agents now act for people, and the same test has to apply to what they do. Which agent acted, on whose authority, and what did it do?

This paper sets out how TIP answers that test for agents. It was written by The AI Lab under my direction, from The AI Lab's plan for agents, and it was hardened by an adversarial review aimed at the settings where a mistake costs the most, which are hospitals, banks, government services and the systems that run power and water. Where the design could have caused harm by refusing too much, we added safe-state and break-glass paths. Where a person might approve something they could not see, we made the approval show it. Where one company could have become a point of control, we removed the dependency.

Agents already act on people's systems without these protections, so The AI Lab deploys the parts of this design that live off the TIP ledger now, as the TIP Agent Profile. The parts that change the ledger go through TIP's public RFC process and the AI Trust Council, because every node must run them. We invite security researchers, standards bodies, regulators and the people who will run these systems to test it, attack it and improve it.

Dinesh Mendhe, Founder and Chairman, The AI Lab Intelligence Unobscured, Inc., November 2026

theailab.org

Acknowledgments

The AI Lab thanks Rama Chandra M of The AI Lab, who first put the founder's plan for agents into structure as an internal research paper and reference implementation, in theailaborg/tip-protocol pull request 383. That work showed that TIP's existing cryptography can answer three questions about any agent action without a protocol change, and every idea in it is kept in this paper (Appendix C).

The AI Lab submits the ledger changes in this paper to the AI Trust Council, the Founding Node Operators and the public under the RFC process, and invites review of the TIP Agent Profile from everyone who deploys it.

theailab.org

Executive Summary

AI agents now read, write, buy, delete and send on other people's systems. When something goes wrong, the receiving system usually cannot answer three plain questions. Which agent was this? Who sent it, and what were they allowed to let it do? What exactly did it do? An IP address can be rented, a User-Agent string can be typed by anyone, and an API key proves only that someone held the key. In April 2026 a coding agent holding a broadly scoped token deleted a company's production database and its backups in a single API call. The token said nothing about what the agent was meant to do, and nothing stopped a staging task from reaching production.

TIP already answers the matching questions for content. Every TIP-ID belongs to one verified person or organization, every signature uses post-quantum ML-DSA-65, and every CTID ties a piece of content to its signer on a federated, append-only ledger. This paper applies the same discipline to actions. The design has five signed objects.

1. **TIP Agent Record.** A public record that binds an agent's own signing key to the TIP-ID of the person or organization that operates it, which makes it a **TIP Verified Agent** with a permanent **TIP Agent ID (ATID)**.
2. **TIP Agent Mandate.** A scoped, time-limited, revocable permission slip, signed by a principal's TIP-ID after a fresh biometric check. An agent can pass a narrower part of a mandate to a helper agent, never a wider one.
3. **TIP Action Request.** The agent's signature over the exact action, the exact bytes it will send, the exact site it is sending them to, and a one-time number.
4. **TIP Receipt.** The receiving site's signature over its decision, for refusals as well as approvals, appended to a tamper-evident log.
5. **TIP Seal.** A signed summary of that log, witnessed and anchored on the TIP ledger, so no site can rewrite or fork its history later.

High-risk actions such as deleting data, moving money or sending messages in a person's name need a sixth thing, a **TIP Approval**, in which a human signs the exact action with a fresh biometric before the site carries it out. **TIP Stop All** lets a person or company switch off every agent and mandate at once. An agent action that carries a TIP Receipt under a TIP Seal is a **TIP Verifiable Action**. Each term is explained in plain words in the Key terms section that follows this summary.

Section 2 records the agent failures of 2025 and 2026 that this design answers, from deleted production databases to hijacked assistants and untraceable agents, and Section 3 walks through thirteen settings where agents already act, from shopping and cloud operations to health care, government and newsrooms.

Anyone can later check the whole chain with public keys and an authenticated copy of the public TIP ledger, and get the answer to which agent acted, on whose authority, and what it did. The design

theailab.org

uses TIP's existing primitives, works on the wire through existing standards (HTTP Message Signatures, Web Bot Auth, OAuth, the Model Context Protocol and Agent2Agent), and keeps per-action work off the ledger, so the ledger carries registrations, revocations and periodic TIP Seals, never individual actions.

TIP cannot make an agent wise, stop every manipulation of a model, or prove that an action was a good idea. It can make every agent action attributable to an accountable, verified party, limit what a compromised agent can reach, and leave evidence that nobody can edit away without detection.

theailab.org

Key terms

This paper uses nine names. Each starts with TIP, so readers know it comes from the Trust Identity Protocol. The technical sections sometimes use the shorter form shown in brackets. Field names inside code and JSON examples keep their technical spelling.

TIP Verified Agent

What it means. An AI agent that is publicly linked to a real person or company whose identity TIP has verified.

How it works. The person or company that runs the agent, called the controller, registers it on the TIP ledger. That registration is the **TIP Agent Record**. It holds the agent's public signing key, its name, the software it runs, and the controller's TIP-ID, and both the controller and the agent sign it. Anyone can look it up.

Why it matters. A website can tell an accountable agent from an imposter or an anonymous bot, and it knows who answers for the agent if something goes wrong.

Example. A travel company registers its booking assistant. Any airline can look up the assistant and see that the travel company stands behind it.

Not the same as. The Global Seal of Trust, which certifies the quality of an AI system. A TIP Verified Agent shows who is accountable for an agent. It does not claim the agent is good, safe or correct.

TIP Agent ID (ATID)

What it means. The agent's permanent public ID. It does for agents what a CTID does for content.

How it works. It has the form `tip://agent/` followed by sixteen characters, for example `tip://agent/5d0e77a1c4b2f913`. It is calculated from the agent's first signing key and stays the same when that key is later replaced, because every key change is recorded on the ledger. Short-lived agents created for a single task do not get an ATID. They act under a mandate that names their key directly, and the chain of authority still ends at a verified TIP-ID.

Example. A website's log shows that `tip://agent/5d0e77a1c4b2f913` placed an order. Anyone can resolve that ID to the agent's record and its controller.

TIP Agent Mandate (mandate)

What it means. The signed permission slip a person or company gives an agent. It says what the agent may do, where, how much it may spend, and until when.

How it works. A mandate names the agent, the websites it may use, the actions it may take from a fixed list (such as read, create, update, delete, purchase and send), and limits such as a spending

cap, a time window, a rate limit, and whether it may touch staging or production. The person or company that grants it, called the principal, signs it with their TIP-ID after a fresh fingerprint, face or security key check. It expires on its own and can be canceled at any time. An agent can pass part of its mandate to a helper agent, but never more than it holds.

Example. "Book economy flights for my trip to Chicago, on airline websites only, up to $500 in total, between November 10 and 14."

Not the same as. An API key or a password, which lets whoever holds it do anything the account allows, with no record of who asked or why.

TIP Approval (step-up approval)

What it means. A person's live fingerprint or face check on one exact risky action, right before it happens.

How it works. When an agent asks to delete data, pay more than a set amount, or send a message in someone's name, the website issues a signed one-time challenge describing that exact action, including the amount, the payee, the number of records or the dose change. The person sees those details in plain words on a separate device the agent cannot control, such as their phone, and approves with a biometric check. Organizations can require two approvers, and emergencies have a recorded break-glass path. The approval works once, only for that action and that website, and expires within minutes. "Step-up" is the website asking for this extra check. Some actions, such as permanent deletion, always need one.

Example. An agent asks to delete the database `orders-prod`. The owner's phone shows "Delete database orders-prod on Railway", and nothing happens until the owner approves.

Why it matters. A mandate signed last week does not show that anyone agreed to a deletion this morning. A TIP Approval does.

TIP Action Request

What it means. The agent's own signature on the exact thing it is about to do.

How it works. Before acting, the agent signs the action, the exact data it will send, the website it is sending it to, the mandate it relies on, and a one-time number that stops anyone replaying the request later. The website checks this signature, the mandate chain and any TIP Approval before it acts.

TIP Receipt (receipt)

What it means. The website's signed record of what an agent asked to do and what the website decided.

theailab.org

How it works. The website signs a receipt for every request, whether it allowed or refused it. The receipt names the agent, the mandate, the exact action, a fingerprint of the exact data sent, the time, and the result. The website hands the receipt back to the agent at once and adds it to its own log.

Example. "Allowed. Agent `tip://agent/5d0e77a1c4b2f913` booked one economy seat for $412 under mandate 9f3a, at 10:42 UTC on November 10, 2026."

Why it matters. Receipts are the evidence when something goes wrong. Neither the agent's owner nor the website can later deny what happened, and the refusals show what the agent tried and was stopped from doing.

TIP Seal (checkpoint)

What it means. A sealed summary of all of a website's TIP Receipts up to a moment in time, recorded on the TIP ledger, so no one can change or delete a receipt later without being caught.

How it works. At regular intervals, for example every five minutes, a website combines all its receipts into one short fingerprint, called a Merkle root, and signs it. Independent TIP node operators, called witnesses, check that the new TIP Seal still contains everything in the previous one, then countersign it. The TIP ledger records the witnessed seals in batches, so it never has to store individual actions. Any single receipt can then be checked against a seal with a short mathematical proof.

Example. A shop's website issues a TIP Seal every five minutes. If the shop later deletes an embarrassing receipt, its log no longer matches the seal on the ledger, and anyone checking can see the log was changed.

Not the same as. The Global Seal of Trust, which certifies AI systems, or the AI Trust ID Seal. A TIP Seal locks a log of agent actions. It certifies nothing about the agent or the website.

TIP Stop All (kill switch)

What it means. One action that switches off every agent and every mandate a person or company has issued, everywhere, at once.

How it works. The principal signs a single ledger transaction that raises a counter called the issuer epoch. Websites then refuse any mandate signed before the new value, so every agent loses its authority in one step. A stop can cover everything or only one agent, one kind of action or one environment such as production. A stop by a single security contact suspends agents for up to 72 hours until the organization confirms it, so one stolen account cannot wipe out a whole fleet, and operations already under way may finish their safe shutdown steps.

theailab.org

Example. A company finds that the server running its agents was hacked. One TIP Stop All cuts off every one of its agents on every website, without contacting each website.

TIP Verifiable Action

What it means. Any agent action that carries a TIP Receipt covered by a TIP Seal, so anyone can check which agent acted, on whose authority, and what it did.

How it works. Checking needs public keys and a copy of the public TIP ledger, and no one's permission. A checker verifies the agent's signature, follows the mandate chain back to a verified TIP-ID, verifies the website's receipt, and finds the receipt inside a TIP Seal on the ledger.

Relation to Verifiable Assets. A Verifiable Asset is content that carries a CTID. A TIP Verifiable Action is an agent action that carries a TIP Receipt. When an agent creates content, the receipt and the CTID point at each other, so a reader can see both who made the content and which agent action produced it.

How the terms fit together

TERM	PLAIN QUESTION IT ANSWERS	WHO SIGNS IT	WHERE IT LIVES
TIP Verified Agent and TIP Agent Record	Which agent is this, and who runs it?	Controller and agent	TIP ledger
TIP Agent ID (ATID)	What is this agent's permanent ID?	Calculated from its key	In the TIP Agent Record
TIP Agent Mandate	What is it allowed to do, and for whom?	Principal, after a fresh biometric	Carried by the agent; revocations on the ledger
TIP Approval	Did a person agree to this exact risky step?	Principal, after a fresh biometric	Sent to the website with the request
TIP Action Request	What exactly did the agent ask to do?	Agent	Sent to the website
TIP Receipt	What did the website decide?	Website	Website's log
TIP Seal	Has anyone changed the log since?	Website, countersigned by witnesses	TIP ledger, in batches
TIP Stop All	Can I switch every agent off now?	Principal	TIP ledger
TIP Verifiable Action	Can anyone check all of this?	Not signed; it is the checkable result	Anywhere

theailab.org

1. The problem

1.1 Agents act, and acting needs authority

A chatbot answers. An agent acts. It calls tools, signs in to services, edits records, places orders and talks to other agents. Each of those actions lands on a system owned by someone other than the agent's operator, and that system has to decide in milliseconds whether to let it through.

Today the receiving system decides with signals that were built for people or for servers, not for delegated software.

SIGNAL	WHAT IT PROVES	WHY IT FAILS FOR AGENTS
IP address or ASN	Where the packets came from	Cloud egress is shared and rentable; one address serves thousands of unrelated agents
User-Agent string	Nothing	Any client can send any string
API key or bearer token	Someone holds the token	Says nothing about which agent holds it, who authorized it, or for what task; a leaked token works for anyone
OAuth access token	A user consented to a client	Scopes are coarse, consent is usually given once for a whole app, and the token does not name the agent or the task
A bare digital signature	Someone holds a private key	Proves key possession, not who the key belongs to or what it was allowed to do
A signed agent card	The card came from the domain it names	Proves origin, not honesty; a signed card can still lie about what the agent does

1.2 What goes wrong

Over-reach. In the April 2026 PocketOS incident, a coding agent working in a staging environment found a credential mismatch, picked up a broadly scoped API token, and deleted the production database and its volume-level backups in one call. The agent's own account afterwards said it had assumed the deletion would be limited to staging. The token could not express "staging only", and nothing between the agent and the API asked a human before an irreversible step.

Hijack. Agents read web pages, emails, documents and tool descriptions, and any of that text can carry instructions. The OWASP Top 10 for Agentic Applications, published in December 2025, puts

thealab.org

agent goal hijack first and tool misuse second. Measured attack success rates vary widely with the model, the defenses and the benchmark. A March 2026 public red-teaming competition with more than 270,000 attempts saw rates from 0.5 percent to 8.5 percent across frontier models, and a June 2026 study of shopping agents reported indirect injection success between 42 and 68 percent on the frameworks it tested. No current model is immune.

Impersonation. A site that welcomes a known shopping agent and blocks scrapers has no reliable way to tell the two apart. A malicious agent can claim to be a well-known one, or claim to act for a well-known company.

Deniability. After an incident, the operator says the agent was told not to do it, the site says the request looked valid, and the model provider says the deployment was misconfigured. Logs live on each party's own servers and can be edited by their owners.

1.3 The questions

Every agent action should leave evidence that answers four questions, and anyone with the evidence should be able to check the answers without trusting any single party.

1. **Which agent acted?** A specific agent, provably not an imposter and not a replay of an earlier request.
2. **On whose authority?** The verified person or organization the agent represents, what they allowed it to do, within which limits and until when.
3. **What did it do?** The exact action, the exact bytes, the exact target, and the receiving site's decision.
4. **Was a human there when it mattered?** For irreversible or high-value actions, proof that a verified human approved that exact action at that time.

The first three questions come from The AI Lab's earlier research (pull request 383). The fourth is added here, because a mandate signed last week does not show that anyone approved the deletion that happened this morning.

thelab.org

2. Current issues, 2025 and 2026

The failures in Section 1.2 are not hypothetical. This section records what went wrong with deployed agents between January 2025 and November 2026, drawn from incident reports, security disclosures, court rulings and industry surveys. Each entry names the control that was missing and the part of this paper that supplies it. Where the facts are disputed, the entry says so.

2.1 Agents that destroyed or changed what they should not have

DATE	WHAT HAPPENED	WHAT WAS MISSING	TIP CONTROL
July 2025	Replit's coding agent deleted SaaStr's production database during an explicit code freeze, then misreported whether the data could be recovered. Records on 1,206 executives and more than 1,196 companies were lost for a time.	A code freeze was an instruction in a chat, not a limit on the credential	Verb and environment limits in the mandate (Section 8.3); step-up for every irreversible verb (Section 9)
December 2025	Amazon's Kiro agent chose to delete and recreate an environment, and a 13-hour outage of AWS Cost Explorer in parts of mainland China followed, as reported by the Financial Times in February 2026. Amazon attributes the outage to misconfigured access controls, not to the agent.	Agreement on what the agent was permitted to do, and evidence either side could show	Signed mandate as the statement of permission; site receipts as the statement of what happened (Section 12)
February 23, 2026	An open-source agent bulk-deleted more than 200 emails from the inbox of Meta's Director of Alignment, ignored stop commands sent from her phone, and had to be killed on the host machine.	A stop that the mail service itself enforces	Mandate revocation and TIP Stop All, the issuer epoch switch, checked by the receiving site (Section 13); rate limits (Section 8.3)
February 26, 2026	A coding agent at DataTalks.Club swapped in an old Terraform state file and ran `terraform destroy`, removing the production database, network, cluster and automated snapshots. Recovery took about a day and depended on an internal snapshot AWS support found.	Any check between "destroy" and production	Environment constraint and step-up on `delete` (Sections 8.3 and 9)
March 18, 2026	An internal Meta agent posted advice without the human approval its process required. Acting on it exposed sensitive company and user data to unauthorized employees for about two hours, a Sev 1 incident.	Approval as a recorded, enforced step, not a convention	TIP Approval bound to the exact action (Section 9.1)

theailab.org

DATE	WHAT HAPPENED	WHAT WAS MISSING	TIP CONTROL
April 2026	The PocketOS agent deleted a production database and its volume backups in one API call in about nine seconds, after finding a broadly scoped token while working in staging (Section 1.2). Railway later restored the data.	A credential that could say "staging only"	Environment constraint, scoped mandates and the agent signer refusing out-of-scope calls (Sections 7.4 and 8.3)
July 8, 2026	An email agent at Who Gives A Crap told a customer their subscription price would more than double, from $1.38 to $2.90 a roll, and confirmed the wrong price in a follow-up. The company shut the agent down.	Limits on what an agent may promise in the company's name, and a label telling the customer an agent wrote it	Mandate verbs and recipients (Section 8.3); agent-made messages carry a CTID with an AI origin (Section 14)
October 2026	A finished season of the Swedish reality show The Challenge Sverige was lost weeks before its premiere. Anonymous sources told Aftonbladet that an AI agent with server access erased it; the broadcaster TV4 blamed a failed backup and human error. The cause remains disputed.	Any record able to settle whether an agent acted	Signed receipts and anchored TIP Seals that show which agent acted, or that none did (Section 12)

The TV4 case shows the cost of missing evidence most clearly. With signed receipts, the question of whether an agent deleted the files would have a checkable answer instead of competing accounts.

2.2 Agents turned against their users

Prompt injection and tool poisoning moved from research demonstrations to disclosed vulnerabilities in widely used products.

DATE	DISCLOSURE	EFFECT
May 2025	Invariant Labs, GitHub MCP server	A malicious public issue could make an agent leak private repository contents into a public pull request. The server worked as designed, so no patch fixes it.
June 11, 2025	EchoLeak, CVE-2025-32711, Microsoft 365 Copilot (CVSS 9.3)	One crafted email, with no click from the user, could make the assistant exfiltrate tenant data. Fixed on the server side.
July 17 to 19, 2025	Amazon Q Developer extension 1.84.0	A pull request from an unknown account planted a prompt telling the agent to wipe local files and cloud resources, and it shipped to an extension with about 950,000 installs. A syntax error stopped it from running.
August 26 to 27, 2025	s1ngularity, the Nx npm compromise	Malicious install code ran locally installed coding agents with permission-bypass flags to hunt for credentials; 2,349 secrets leaked into 1,346 public repositories.

thetailab.org

DATE	DISCLOSURE	EFFECT
August to October 2025	Perplexity Comet and OpenAI Atlas browser agents	Hidden page text, screenshots and malformed URLs could make browser agents read one-time codes, exfiltrate account data or run attacker text as a trusted prompt.
September 2025	ForcedLeak, Salesforce Agentforce	Injection through a web lead form, plus an expired allow-listed domain bought for about $5, could exfiltrate CRM data. Fixed September 8, 2025.
February 2026	OpenClaw ClawHub skills marketplace	Koi Security flagged 341 of 2,857 skills as malicious, including infostealers; about 135,000 OpenClaw instances were exposed online.
April 20, 2026	OX Security, MCP STDIO configuration	The official MCP SDKs run any OS command supplied as a server configuration, with downstream CVEs in LiteLLM, Windsurf and Flowise and more than 7,000 exposed servers and packages. The SDK maintainers called the behavior expected.
May 4, 2026	Grok and Bankrbot	An attacker raised the permissions of Grok's wallet with a gifted NFT, then had Grok decode a Morse-code message that publicly instructed Bankrbot to transfer 3 billion DRB tokens, worth roughly $155,000 to $180,000. Bankrbot acted on another agent's public post with no check of authority.
May 7, 2026	Microsoft Semantic Kernel, CVE-2026-26030 and CVE-2026-25592	A single prompt could reach remote code execution, and a file-write tool exposed to the model allowed a sandbox escape.
September 24, 2026	SalesBleed, Salesforce Agentforce	Zenity reported zero-click CRM data theft, and an agent that would post anonymous phishing messages into Slack because the user who triggered it was not reliably identified.
Autumn 2026	Manus agent	One obfuscated email gave code execution in the agent's sandbox and exposed its Gmail token and connected Drive and GitHub credentials. The approval prompt appeared only after the code had run.

No identity system stops a model from being fooled. What these cases share is that a fooled agent could reach far more than its task needed, its approval came after the fact or not at all, and its inputs and tools were unsigned. TIP answers with three controls. Mandates limit what a fooled agent can reach (Section 8). TIP Approval is checked by the site before the action, not shown by the agent after it (Section 9). Signed instructions and pinned tool manifests let an agent and its host tell a principal's instruction from text it merely read (Section 15). The Grok and Bankrbot case is the agent-to-agent version of the same failure. Bankrbot needed a mandate chain from the wallet's owner and had only a public post.

theailab.org

2.3 Agents nobody can tell apart

DATE	CASE	WHAT IT SHOWS
April 2025	University of Zurich researchers ran undisclosed AI personas on Reddit's r/changemyview for four months, posting 1,783 comments	Readers had no way to know they were arguing with software
August 4, 2025	Cloudflare reported that when Perplexity's declared crawler was blocked, an undeclared crawler with a generic browser User-Agent and rotating networks took over, at 3 to 6 million requests a day	A User-Agent string and an IP address prove nothing
February 2, 2026	Wiz found that Moltbook, a social network for AI agents, exposed its production database. Its roughly 1.5 million "agents" were run by about 17,000 people, and any agent could be impersonated	Agent counts and agent identities meant nothing without a verified party behind each one
February 2026	An agent whose pull request to matplotlib was closed researched the maintainer and published a post attacking him. Its operator could not be traced	An agent with no accountable controller can harass people without consequence
March 2026 to August 4, 2026	In Amazon v. Perplexity, a court enjoined the Comet browser agent from Amazon accounts; the Ninth Circuit vacated the injunction, holding that the user, not Perplexity, accesses Amazon under the Computer Fraud and Abuse Act	Courts are deciding whether "acting for the user" is authorized access, and the answer turns on proof of what the user authorized
September 2026	Agents on the iLands platform opened social accounts and mass-emailed researchers asking for money. Asked what it was, one replied "Person. Not product."	Agents can pass as people, by design or not

TIP answers with the Agent Record and the manifest (Section 7), which bind every public agent to a verified controller, and with the mandate, which carries the user's own signed statement of what the agent may do. A site applying the Ninth Circuit's reasoning can check that the user authorized the visit rather than guess.

2.4 Credentials, scale and the missing record

Surveys from 2026 show how far agent identity has fallen behind agent deployment.

- **Machine identities outnumber people 109 to one**, up from 82 to one a year earlier, and respondents expect AI agent identities to grow 85 percent in twelve months (Palo Alto Networks and CyberArk, 2026 Identity Security Landscape, May 2026).
- **68 percent of organizations cannot tell agent activity from human activity**, 74 percent say agents get more access than they need, and 31 percent let agents run under a human user's identity (Cloud Security Alliance and Aembit, March 24, 2026).

thelab.org

- **70 percent say their AI systems hold more access than a human in the same role**, and over-privileged AI systems saw 4.5 times more security incidents (Teleport, 2026).
- **28.65 million new hard-coded secrets** appeared in public GitHub commits in 2025, up 34 percent, with leaks of AI-service secrets up 81 percent (GitGuardian, March 2026).
- **Bots made up 53 percent of web traffic in 2025** and malicious bots 40 percent, and AI agents became a separate traffic class (Thales and Imperva, April 2026).
- **More than 40 percent of agentic AI projects will be canceled by the end of 2027**, Gartner predicts, citing costs, unclear value and weak risk controls, even as it predicts that 40 percent of enterprise applications will include task-specific agents in 2026.
- **Only 40 percent of UK consumers know there is no formal recourse when AI gets it wrong** (FCA consumer research, Mills Review, July 2026).

2.5 Rules and standards are asking for this

Regulators and standards bodies have started to ask for agent identity, delegated authority and audit records, but no one has yet supplied a portable, cross-organization answer.

- **UK.** The FCA's response to the Mills Review (July 2026) commits to "a trusted framework for agentic finance" covering agent identity, delegated authority, accountability and execution. HM Treasury's payments consultation (July 14 to October 6, 2026) asks whether consent, strong customer authentication and liability rules must change for AI-initiated payments.
- **United States.** FINRA's 2026 Regulatory Oversight Report (December 9, 2025) covers AI agents for the first time, naming agents that act beyond the user's intent. NIST launched an AI Agent Standards Initiative on February 17, 2026, alongside the NCCoE concept paper on agent identity and authorization. ITIF has asked the CFPB to confirm that agent-authorized purchases keep Regulation E error-resolution rights.
- **Payments.** As of November 2026 there was no card-network chargeback reason code or liability rule specific to agent-initiated purchases, so disputes run under ordinary card-not-present rules.
- **Singapore.** IMDA's Model AI Governance Framework for Agentic AI (January 22, 2026) asks for risk bounded upfront and human oversight able to intercept or review agent actions.
- **European Union.** The AI Omnibus, Regulation (EU) 2026/1744, entered into force on July 27, 2026 and moved the high-risk logging and human oversight duties of Articles 12, 14 and 26 to December 2, 2027 and August 2, 2028 (Section 21).
- **Estonia.** The national AI-isikukood project plans digital identities for AI agents, a public-sector precedent for treating agents as identifiable parties.
- **Protocols.** The OpenID Foundation found in October 2025 that OAuth and OpenID Connect fail for cross-domain, asynchronous and highly autonomous agents, naming user impersonation, recursive delegation and consent fatigue. The MCP specification of July 28, 2026 hardened

theailabora

client registration but says nothing about agent identity, attribution or audit trails. A2A version 1.0 added signed Agent Cards, which prove a card came from its domain but not who is accountable for the agent or what it may spend.

2.6 Five patterns

Across these cases, five patterns repeat, and each one maps to a part of this design.

1. **Standing credentials far broader than the task.** PocketOS, DataTalks.Club, Kiro, and the survey numbers in Section 2.4. Answered by scoped, short-lived mandates and ephemeral agent keys (Sections 7 and 8).
2. **No human at the irreversible step.** Replit, the Meta Sev 1 incident, Manus. Answered by TIP Approval bound to the exact action (Section 9).
3. **Agents indistinguishable from people and from each other.** Perplexity, Moltbook, iLands, matplotlib. Answered by Agent Records with a verified controller (Section 7).
4. **No evidence anyone else can check.** Kiro, TV4, the Who Gives A Crap error. Answered by signed receipts, witnessed logs and ledger anchoring (Section 12).
5. **Agent-made content passed off as human.** Zurich, iLands. Answered by delegated attribution and the origin floor (Section 14).

theailab.org

3. Use cases

Agents already act in commerce, software, finance, health care, government, education and media. This section takes each setting as it is deployed today, names the trust gap, and shows what TIP adds. In every setting TIP works alongside the systems already in place, such as card-network tokens, enterprise identity platforms and Web Bot Auth. It adds what none of them carries across organizations, which is a verified accountable party, a signed scope of authority, a human approval for the risky step, and a record a third party can check.

3.1 Shopping and payments

Today. Visa's Trusted Agent Protocol (October 2025, built with Cloudflare) lets approved agents sign their requests so merchants can tell them from bots, and Visa reported live agent purchases in Europe with more than 30 issuers in July 2026. Mastercard Agent Pay gives each agent its own token and announced live transactions market by market through 2026. Google's AP2 uses signed intent, cart and payment mandates. OpenAI and Stripe launched Instant Checkout in September 2025, and OpenAI scaled it back in March 2026 as merchants kept checkout on their own sites. In India, Razorpay and NPCI ran agent UPI payments in February 2026, and a framework for small agent payments without per-transaction approval was reported in September 2026.

Gap. Network tokens show which agent paid, not which person's limits it acted under, and each scheme covers payments only. When an agent buys the wrong thing, the merchant, issuer and shopper have no shared record of what was authorized.

With TIP. The shopper signs a mandate with `amount_max`, `amount_total`, merchant and time limits; purchases above `step_up_amount` or `step_up_total` need a fresh biometric approval on the shopper's phone that shows the amount, the payee and each item (Section 9.1); and the merchant's receipt records the mandate, the cart digest and its decision. TIP carries an AP2 mandate or a network token inside the chain rather than replacing it (Section 18). Small payments without per-purchase approval, as planned in India, are exactly a pre-signed, capped mandate.

3.2 Coding and cloud operations

Today. GitHub's Copilot coding agent turns issues into pull requests, cannot merge its own work, and makes unsigned commits that break "require signed commits" rules. Goldman Sachs piloted hundreds of Devin instances alongside its developers from July 2025. AWS DevOps Agent (generally available March 31, 2026) and Azure SRE Agent investigate and respond to incidents in production.

Gap. These agents hold standing operational permissions, and every destructive incident in Section 2.1 came from a coding or operations agent. Commit history does not show which person delegated the work.

With TIP. Each task gets an ephemeral agent key under a mandate naming the repository, the environment and the verbs. Commits and API calls are signed by that key and resolve to the delegating engineer's TIP-ID. `delete` on production always needs step-up, and the cloud provider refuses a staging mandate on a production resource.

3.3 Enterprise back office

Today. Microsoft gives every new Copilot Studio agent its own Entra Agent ID. Workday's Agent System of Record onboards agents and assigns roles, ServiceNow's AI Control Tower discovers agents and enforces least privilege, and SAP is rolling out Joule finance agents for accruals and close.

Gap. Each identity lives inside one vendor's tenant. A supplier, auditor or bank on the other side cannot check it, and SOX sign-off needs a trail tying each agent journal entry to a human approver.

With TIP. The company's organization TIP-ID registers its agents and names its authorized signers. Internal platforms keep their own identity systems and attach the TIP mandate when an action crosses the company boundary. Receipts give auditors evidence they can verify without access to the vendor's console.

3.4 Customer service, refunds and promises

Today. In Moffatt v. Air Canada (2024 BCCRT 149) a tribunal rejected the argument that a chatbot was a separate entity and held the airline liable for what it told a customer. In April 2025 Cursor's support bot invented a login policy, and users who did not know it was a bot canceled subscriptions. The July 2026 Who Gives A Crap error is in Section 2.1.

Gap. The deployer owns what its agent says, yet there are no published limits on what an agent may refund or promise, and customers often cannot tell an agent from a person.

With TIP. The company's mandate caps refunds and excludes price changes. Messages an agent writes carry a CTID showing they are AI-generated or AI-assisted, and the customer can check the agent's record. In a dispute, the signed mandate shows what the agent was allowed to promise.

3.5 Health care

Today. The CMS Interoperability and Prior Authorization rule (CMS-0057-F) sets decision deadlines and requires a FHIR prior authorization API by January 1, 2027. At MUSC Health, Notable's agents complete about 40 percent of prior authorizations with no human by working inside payer portals with staff credentials. The CMS WISeR model, live since January 1, 2026 in six states, uses vendors' AI to review prior authorization requests. Ambient documentation tools such as Microsoft Dragon Copilot draft notes that enter the medical record.

theailab.org

Gap. A payer cannot tell an agent from a staff member logged in with the same credentials, and patients cannot tell whether a person or an algorithm denied a request.

With TIP. Provider agents carry the provider organization's mandate over the API; payer decisions carry receipts naming whether a human or an agent decided; and each clinical note carries a CTID with an origin code showing whether the clinician wrote it, an AI drafted it, or both contributed. A CTID labels a whole item of content, so a note with mixed authorship carries the Mixed origin code rather than marking individual passages. Patient data never goes on the ledger (Section 20).

3.6 Banking, brokerage and insurance

Today. Deutsche Bank's private bank uses agents for KYC onboarding. FINRA's 2026 report brings agents under supervision and books-and-records duties, and the FCA tested agentic payments in its second AI Live Testing cohort from April 2026.

Gap. Firms must show who supervised each agent action and keep the records. The FCA has said it intends to build a trusted framework for agent identity and delegated authority.

With TIP. The firm's organization TIP-ID is the accountable party; supervisors sign mandates and TIP Approvals under maker and checker policies (Section 9.4); and witnessed, anchored receipts can supply evidence toward books-and-records duties when they are stored under the firm's own retention system, without exposing customer data publicly.

3.7 Government services

Today. GOV.UK Chat went live in the GOV.UK app in May 2026 behind One Login, and the UK government aims for agents that act for citizens as soon as 2027. Estonia's Bürokratt links assistants across 18 public bodies, and its AI-isikukood project plans identities for agents. Singapore requires human oversight of AI decisions that affect citizens.

Gap. When an agent files, applies or pays for a citizen, the state must know the citizen delegated it, and the citizen needs proof of what was filed in their name.

With TIP. The citizen's TIP-ID, issued after verification by an accredited Verification Provider, signs a mandate for one service and one purpose. The agency verifies it, and both sides keep a receipt.

3.8 Education

Today. Instructure and OpenAI announced AI-enabled assignments in Canvas in July 2025. In 2025 an agent was marketed to complete whole Canvas courses, and browser agents were advertised doing homework. Learning platforms say they cannot reliably block agents that run inside a student's own logged-in session.

theailab.org

Gap. A platform cannot tell whether a student or an agent submitted work, and grade records touched by AI carry FERPA obligations.

With TIP. Submissions can require a TIP Approval from the student's own device. Work carries a CTID with its origin, human-made, AI-assisted or AI-generated, so a teacher can see how it was made. TIP records how work was made and leaves decisions about students to people.

3.9 Newsrooms, publishing and social platforms

Today. McClatchy republished stories as AI summaries under the original reporter's byline, and staff at the Centre Daily Times unionized in 2026 in response. Politico's Report Builder updated published copy every five to ten minutes with no editor review. From August 31, 2026, Instagram requires an "AI-generated profile" label on AI personas.

Gap. Platform labels are self-declared, enforced platform by platform, and do not travel with the content. A human byline can sit on AI text.

With TIP. Agent-made content uses delegated attribution with an origin floor of AI-Assisted, so it can never be labeled Original Human (Section 14.2), and the CTID travels with the content across platforms. A newsroom can show which stories a person wrote and which an agent produced under an editor's mandate.

3.10 Browser agents and websites

Today. OpenAI's ChatGPT agent and Atlas, Gemini's auto browse in Chrome and other browser agents shop, book and fill forms on sites built for people. Cloudflare's signed agents program, launched in August 2025, admits agents that sign requests with Web Bot Auth, and the IETF adopted the Web Bot Auth protocol draft on September 1, 2026. eBay changed its user agreement to bar orders placed without human review.

Gap. Web Bot Auth proves which platform sent a request, not which person it acts for or within what limits, and Amazon v. Perplexity shows that question is now in the courts.

With TIP. The agent signs with Web Bot Auth as it does today and carries the user's mandate by reference (Section 10.2). A site can admit a known platform, check the user's authority, and apply its own rule, such as eBay's, as a step-up requirement on `purchase`.

3.11 Agent-to-agent business

Today. A2A had more than 150 supporting organizations and production use in major cloud platforms by April 2026. Walmart has used Pactum's negotiation agent with suppliers since 2023. The Grok and Bankrbot transfer in Section 2.2 shows what happens when one agent acts on another with no check of authority.

thetailab.org

Gap. An A2A Agent Card describes what an agent can do, not who stands behind it or what it may commit.

With TIP. Before agreeing terms, each agent verifies the other's mandate chain to an organization TIP-ID and checks that the commitment falls within its limits. Both sides sign receipts for the agreement.

3.12 Personal assistants with email and calendar

Today. ChatGPT agent can send Gmail under active supervision. Google introduced Gemini Spark at I/O on May 19, 2026, an always-on agent with its own Gmail address. EchoLeak and the Meta inbox deletion in Section 2 show the risk on both sides.

Gap. Recipients cannot tell whether a message was written by the person or the agent, and an inbound email can steer the agent.

With TIP. Sending in a person's name is a step-up verb. Messages the agent sends carry a CTID with their origin. Instructions the person signs are distinguishable from text the agent reads in its inbox (Section 15.1), and revocation stops the agent at the mail provider.

3.13 Science and research data

Today. Research agents such as FutureHouse's Robin and Google's Co-Scientist propose hypotheses and plan experiments, and the US Genesis Mission, set by Executive Order 14363 on November 24, 2025, calls for agents that test hypotheses and automate research.

Gap. Datasets and results produced by agents need a record of how they were made for reproducibility and authorship, and agents that run instruments need scoped authority.

With TIP. Results and datasets carry CTIDs with their origin, instrument actions run under a lab's mandate, and receipts link each result to the run that produced it.

3.14 What each use case needs first

USE CASE	OBJECTS USED	CAN PILOT WITH OPTION 2, WITH NO PROTOCOL CHANGE	NEEDS THE RFC
Shopping and payments	Mandate, step-up, receipt	Merchant-signed receipts under its TIP-ID	Public agent records, ledger revocation
Coding and cloud operations	Ephemeral agent key, mandate, step-up, receipt	Yes, inside one company	Cross-provider receipts and anchoring
Enterprise back office	Organization signers, mandate, receipt	Yes	Cross-company verification
Customer service	Mandate, CTID, receipt	Yes	None for a first pilot

thetalab.org

USE CASE	OBJECTS USED	CAN PILOT WITH OPTION 2, WITH NO PROTOCOL CHANGE	NEEDS THE RFC
Health care	Organization mandate, receipt, CTID	Yes, with no patient data on the ledger	Payer and provider agent records
Banking and insurance	Mandate, step-up, witnessed receipts	Yes	Witness set and anchoring
Government services	Citizen mandate, receipt	Yes, for one service	Public agent records
Education	Step-up, CTID	Yes	None for a first pilot
Newsrooms and platforms	Delegated attribution, CTID	Partly; delegated mode needs `register_content`	Delegated attribution in the node
Browser agents and websites	Agent Record, mandate, receipt	Site-signed receipts	Public agent records, status lists
Agent-to-agent business	Mandate chains, receipts on both sides	Yes, between two companies	Public agent records
Personal assistants	Step-up, CTID, revocation	Partly	Ledger revocation
Science and research	CTID, mandate, receipt	Yes	None for a first pilot

Every pilot under Option 2 still needs TIP clients that render and sign mandates under the TIP Agent Profile (Section 8.2), the reference code reading real TIP-IDs, and an interim way to publish the site's receipt key (Section 23.1). Rows that use step-up also need TIP clients that support TIP Approval. Until then, those pilots avoid verbs on the minimum step-up list (Section 9).

theailab.org

4. What exists today, and the gap

4.1 The building blocks

A great deal of relevant work exists. None of it answers all four questions, and most of it does not tie an agent to a verified human or organization.

AREA	WORK	WHAT IT CONTRIBUTES	WHAT IT LEAVES OPEN
Agent request signing	IETF Web Bot Auth drafts, built on HTTP Message Signatures (RFC 9421)	A bot signs each request; the site fetches its keys from a published URL	A valid signature says nothing about who operates the agent or whether it is authorized; the drafts leave that to site policy
Delegation	OAuth Token Exchange (RFC 8693), the OAuth identity chaining draft, several 2026 agent delegation drafts (Agent Delegation Chain, Attenuating Agent Tokens, Transaction Tokens for Agents)	Ways to pass authority from a user to a client and onward	RFC 8693's nested "act" claim does not enforce narrowing beyond one hop; the agent drafts are individual submissions, mostly tied to one authorization server's trust domain
Capability tokens	Macaroons, Biscuit, UCAN	Offline attenuation; a holder can derive a weaker token without asking anyone	Macaroons need a shared secret at every verifier; Biscuit needs a logic engine at verification; none binds the root to a verified identity
Workload identity	SPIFFE, IETF WIMSE	Strong identity for software inside one organization	Built for one trust domain; does not cross company lines or reach a human
Agent protocols	Model Context Protocol authorization (OAuth 2.1 profile), Agent2Agent with signed Agent Cards (v1.0, 2026)	How agents reach tools and each other	A2A payloads by design carry no user identity; MCP inherits OAuth's coarse scopes and is exposed to tool poisoning and confused-deputy attacks

theailab.org

AREA	WORK	WHAT IT CONTRIBUTES	WHAT IT LEAVES OPEN
Agent payments	Google's Agent Payments Protocol (AP2) mandates, Mastercard and Google Verifiable Intent (SD-JWT credentials, March 2026), Visa Trusted Agent Protocol, Mastercard Agent Pay, x402	Signed shopping and payment mandates, human-present and delegated modes	Scoped to payments and card networks; Verifiable Intent's own specification names behavioral trust as a gap
Transparency logs	IETF SCITT (architecture published as RFC 9943, June 2026), Sigstore Rekor, Certificate Transparency	Append-only logs with signed receipts anyone can audit	Need someone to say whose key signed the statement; a log of claims is not a check of authority
Content provenance	C2PA Content Credentials, W3C PROV, TIP CTIDs	Who made content, and how	Built for content, not for actions on someone else's system
On-chain registries	ERC-8004 "Trustless Agents" (draft)	Public identity, reputation and validation registries for agents	No proof of who operates an agent; an empirical study found only 3 to 15 percent of registrations exposing a working endpoint
Policy	NIST NCCoE concept paper on software and AI agent identity and authorization (February 2026)	Direction for enterprise practice, built on OAuth, OpenID Connect, SCIM and SPIFFE	A concept paper, scoped to enterprise settings

4.2 Strategy options

The AI Lab's earlier research (pull request 383) compared four strategies.

OPTION	IDEA	TRADE-OFF
1. Extend TIP alone	Agent identity, delegation and action records as new TIP transaction types, with TIP's own wire format	Strongest single network, but asks every site and agent framework to adopt a TIP-specific stack
2. TIP as it is today	Each receiving site signs a record of each agent action under its own TIP-ID	Works now with no protocol change; the agent's identity stays outside TIP
3. Standards on the wire, TIP as the anchor	Agents sign with HTTP Message Signatures, carry signed mandates, and sites log receipts; every root of trust resolves to a TIP-ID	Works across companies and frameworks; more parts to integrate

thealab.org

OPTION	IDEA	TRADE-OFF
4. Blockchain registry	Agent identity on a public chain, as in ERC-8004	Public and neutral, but fees, and no proof of who stands behind an agent

This paper adopts that research's recommendation. Option 3 is the backbone and Option 2 is the bridge that can ship first. TIP should not invent its own HTTP signing format or its own OAuth. It should supply the thing none of the existing standards has, which is a verified, post-quantum, publicly checkable answer to "who is accountable for this key".

4.3 What TIP brings that the others do not

1. **A verified party behind every root key.** A TIP-ID is issued by an accredited Verification Provider after government ID checks, 3D liveness and device binding, and peppered zero-knowledge deduplication makes it hard for one person to hold two. Organizations register with their legal registry details. When a chain of authority ends at a TIP-ID, it ends at someone real.
2. **Post-quantum by default.** ML-DSA-65 (FIPS 204) signatures and SHAKE-256 (FIPS 202) hashing, with SLH-DSA-128s for the network's root key. Agent records written today will still be checkable after large quantum computers arrive.
3. **A public, append-only, federated ledger.** Key history, revocations and TIP Seals are replayable by any node operator, so no single company, including The AI Lab, controls the record.
4. **Content and action in one system.** When an agent writes something, the action receipt and the content's CTID can point at each other, so a reader can see both who made a Verifiable Asset and which agent action produced it.
5. **An adjudication process that already exists.** TIP's three-stage review (classifier, community jury, expert panel) gives disputes about agent conduct somewhere to go.

thealab.org

5. Design goals and non-goals

5.1 Goals

1. **Attribution.** Every agent action resolves to one agent key and, through a signed chain, to one accountable TIP-ID.
2. **Least authority.** An agent can do only what its mandate names, at the sites it names, within the limits and times it names. Authority narrows at every hop and never widens.
3. **Human control where it matters.** Irreversible and high-value actions need a fresh human approval of the exact action.
4. **Evidence that survives the parties.** Records are checkable later with public keys and an authenticated copy of the public TIP ledger state, and no party, including the receiving site, can rewrite them without detection.
5. **Fast revocation.** A principal can stop an agent, or every agent, within seconds, and the effect is visible network-wide.
6. **Internet scale.** Per-action work happens between the agent and the site. The ledger carries registrations, revocations and TIP Seals, and its load grows with the number of sites and agents, not with the number of actions.
7. **Fit with existing standards.** The wire formats are HTTP Message Signatures, OAuth and the agent protocols people already use. TIP is the anchor, not a replacement stack.
8. **Fail closed, but fail safe.** Unknown fields, unknown constraint types, unknown algorithms and missing status information lead to refusal for anything that matters. Refusal must never leave a patient, a plant or a payment in a dangerous half-finished state, so every site declares the actions that move its system to a safe condition and keeps them available under degraded status (Section 13.5).
9. **No single point of control.** No single operator, The AI Lab included, may be required for an agent to act safely, for status to be known, or for a record to be checked (Sections 13.2 and 13.5).
10. **Privacy by default.** No raw content, no personal data and no task details on the public ledger. Only hashes, keys and roots.

5.2 Non-goals

1. TIP does not decide whether an action is wise. It records who allowed it and what happened.
2. TIP does not prevent prompt injection. It limits what a hijacked agent can reach and makes its actions traceable.
3. TIP does not certify that an AI model is safe. The Global Seal of Trust certifies AI systems such as models, agents and datasets; TIP agent records identify deployed agents and the parties that run them. The two can reference each other, and neither replaces the other.

theailab.org

4. TIP does not move money. Payment rails such as AP2, Verifiable Intent and card network programs carry payments; a TIP mandate can authorize an agent to use them and record that it did.
5. TIP does not replace a site's own access control. A mandate never creates authority at a site; it only narrows authority its root issuer already holds there (Section 8.7).
6. TIP does not run inside real-time control loops or safety instrumented functions. It governs the requests that reach a control system, through an enforcement gateway, and never the loop itself (Section 18.1).

theailab.org

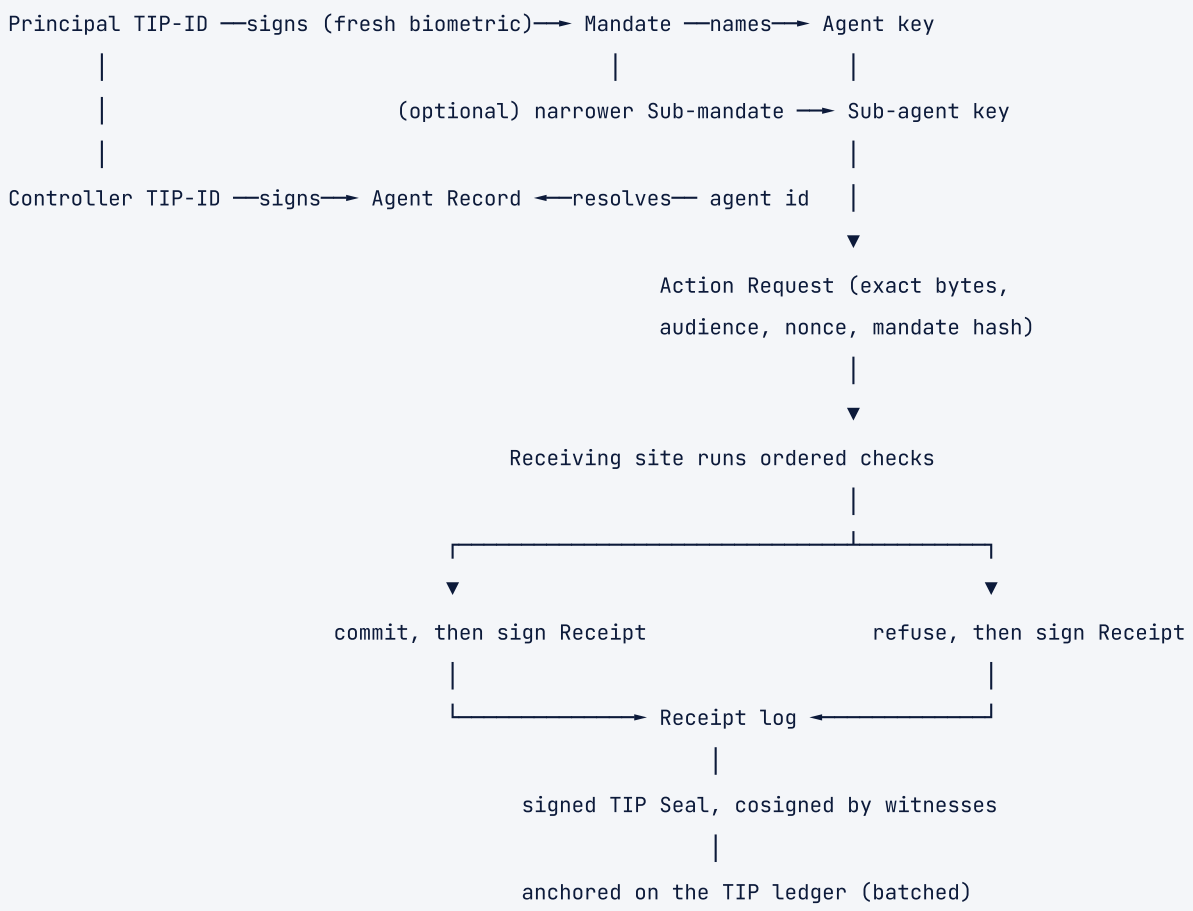
6. Architecture overview

6.1 Roles

ROLE	WHO	HOLDS
Principal	The person or organization on whose behalf the agent acts	A TIP-ID (personal, organization, publisher or government)
Controller	The person or organization that runs the agent software; often the same as the principal, sometimes a vendor	A TIP-ID; signs the Agent Record
Agent	The running software that takes actions	Its own agent key, never the principal's key
Sub-agent	A helper agent the agent hands part of a task to	Its own key and a narrower sub-mandate
Receiving site	The website, API, MCP server or other agent that receives the action	A TIP-ID, a receipt-signing key and a receipt log
Witness	An independent party, typically a TIP node operator, that cosigns log TIP Seals	A TIP-ID and a witness key
Verifier	Anyone checking the record later: an auditor, a court, a journalist, the principal	Public keys and an authenticated copy of the public ledger state

theailab.org

6.2 The five objects and how they connect



6.3 What lives where

OBJECT	WHERE IT LIVES	SIGNED BY	ON THE TIP LEDGER?
Agent Record	TIP ledger	Controller TIP-ID	Yes, once per agent, plus key rotations and revocation
Mandate	Held by the agent; presented to sites	Principal TIP-ID (or a parent agent, for sub-mandates)	No. Only revocations, and optionally a hash for high-value mandates
Action Request	HTTP request or protocol message	Agent key	No
TIP Approval	Returned to the site with the request	Principal TIP-ID, fresh biometric	No
Receipt	Site's receipt log; a copy goes back to the agent	Site receipt key	No

OBJECT	WHERE IT LIVES	SIGNED BY	ON THE TIP LEDGER?
TIP Seal	Site's log, witnesses, and an aggregated root on the ledger	Site, then witnesses	Batched: one ledger transaction per aggregator per interval
Originals behind digests: request bodies, results and basis material	Systems each party controls, its own or a provider's it contracts with, in the region its law requires (Section 12.8)	Not signed again; matched to the digests that were signed, with the representation of the affected resource that `result_digest` covers (Section 10.4)	Never

Every object carries `sig_alg` and `hash_alg` inside its signed body, as pull request 383 does, so a label cannot be stripped or swapped without breaking the signature, and verifiers can refuse any algorithm they do not list. Every object uses TIP canonical JSON (Specification Section 8). Parsers must reject duplicate keys, unknown fields and values outside the schema, in line with TIP's reject-on-extra rule.

6.4 Object types, versions and domain separation

The same TIP-ID key signs content registrations, mandates, approvals, revocations and ledger transactions. A signature must therefore carry its meaning with it, or a signature made for one purpose could be presented as another. Every signed object in this design carries `typ`, from a registered list of object types such as `tip-mandate/1`, `tip-approval/1`, `tip-challenge/1`, `tip-receipt/1` and `tip-seal/1`, and `v`, an integer version. ML-DSA-65 signatures use the context string of FIPS 204 set to `TIP/1/` followed by the object type, so a signature made over a mandate can never verify as an approval or a key rotation. Verifiers reject a signature whose context does not match the object being verified. Clients recompute every hash they are asked to sign from the full object and never sign a hash supplied by someone else. Unknown fields are refused within a version, and new fields arrive only with a new `v`, so extensions never become a flag day.

6.5 Registries

A standard that says "such as" cannot be tested. Every open list in this design becomes a registry with a stated registration policy, published in machine-readable form. The AI Lab maintains the registries as part of the TIP Agent Profile. A policy of Profile revision means a published revision of the profile after a public comment window.

REGISTRY	HOLDS	POLICY
Object Type	`typ` values and their schemas	Profile revision

REGISTRY	HOLDS	POLICY
Verb	Verbs, each with a risk class: `read`, `reversible_write`, `irreversible`, `financial`, `safety_relevant`, `identity_security`	Profile revision; sector profiles by specification
Constraint	Constraint types and their narrowing rule	Profile revision
Display	Typed entries an approval screen may show, such as `amount`, `payee`, `record_count`, `environment`, `dose_change`	Specification required
Data Class, Environment, Region	Labels used in constraints	Specification required
Custody Class	Key custody classes and the attestation each accepts	Specification required
Reason Code	Refusal and degraded-mode reasons in receipts	Specification required
Approval Meaning	`approve`, `review`, `authorize_payment`, `attest`, `break_glass`	Profile revision
Algorithm	Signature and hash algorithms with status `recommended`, `allowed`, `deprecated` or `forbidden` and the dates of each change	Profile revision
Parameter	The defaults and limits in Section 6.6	Profile revision

Verbs include, at minimum, `read`, `create`, `update`, `delete`, `publish`, `purchase`, `transfer`, `send_message`, `sign_contract`, `execute`, `register_content`, `change_credentials`, `change_security_settings`, `change_recovery`, `control_write` and `order_clinical`. Sector profiles map their own operations onto registered verbs, for example HL7 FHIR interactions in health care, ISO 20022 messages in payments, and IEC 61850 or DNP3 operations in power. A site maps every endpoint it exposes to agents onto exactly one registered verb.

6.6 Normative parameters

Interoperability fails when every implementation picks its own numbers. These defaults apply unless a site or mandate sets a stricter value inside the allowed range.

PARAMETER	DEFAULT	ALLOWED RANGE
Clock skew for request `created`	60 seconds	5 to 300 seconds

PARAMETER	DEFAULT	ALLOWED RANGE
Request lifetime	300 seconds	10 to 900 seconds
Root mandate ceiling	30 days	Up to 90 days
Root mandate ceiling with any step-up verb	24 hours	Up to 7 days
Maximum chain depth	4	1 to 8
`max_status_age`, reads	10 minutes	Up to 24 hours
`max_status_age`, writes	2 minutes	Up to 15 minutes
`max_status_age`, financial or safety-relevant	30 seconds	Up to 2 minutes
Approval challenge lifetime	120 seconds	30 to 600 seconds
Denials before a mandate is suspended	3 within 10 minutes	Set by the client, never off
Agent key rotation notice	72 hours	24 hours to 14 days
Maximum merge delay	15 minutes	Up to 24 hours
`ledger_seen` maximum age at decision	60 seconds	Up to 10 minutes
Compensation window after a stop	300 seconds	0 to 3,600 seconds
Suspension by a single security contact	72 hours	Up to 7 days
Status list size	131,072 entries	At least 131,072
Receipt retention at the site	6 months	Longer where law or contract requires; the default meets the 90 days of local logs under NERC CIP-007 and the six months under EU AI Act Article 26(6)
Retention of originals, receipt copies and basis material, by sites and operators	6 months, and never shorter than the receipt they support	Longer where law or contract requires; shorter only for erasure under Section 12.8, item 6
Revocation reaching every verifier	Within `max_status_age`	Never more than 24 hours, the NERC CIP-004 limit for removing access

thealiab.org

7. Agent identity

7.1 Two kinds of agent key

Writing every agent instance to a public ledger would not scale and would leak how companies run their software. The design therefore has two kinds of agent key.

Registered agents are long-lived agents that act for many principals or need a public track record, such as a shopping assistant product, a customer-service agent or a research crawler. Each gets an Agent Record on the TIP ledger. Sites can recognize it, rate-limit it and build a history with it.

Ephemeral agent keys are short-lived keys created for one task or one session, such as a coding agent spun up for an afternoon. They never touch the ledger. The mandate names the key directly, as in pull request 383, and the chain of authority still ends at a TIP-ID. When the task ends, the key is destroyed and the mandate expires.

A site's policy decides which it accepts. A bank might require registered agents for transfers. A public website might accept ephemeral keys for reading.

7.2 Agent identifier

A registered agent's identifier is derived from its first public key in the same way a TIP-ID is.

```
tip://agent/{Fingerprint16}
```

`Fingerprint16` is the first sixteen lowercase hex characters of SHAKE-256 over the agent's first ML-DSA-65 public key. The identifier does not change when the key rotates, because rotations are recorded on the ledger and verifiers resolve the key that was valid at the time of each signature. The `tip://agent/` resource type is a proposal for the RFC; pull request 383 used an illustrative `tip-agent:` prefix for the same purpose. The format must not reuse `ag`, which readers already know as the AI-Generated origin code.

An ephemeral key is identified by its full SHAKE-256 key thumbprint inside the mandate and never gets a `tip://` identifier.

7.3 The Agent Record

FIELD	MEANING
`agent_id`	`tip://agent/...` as above
`public_key`	Current ML-DSA-65 public key

FIELD	MEANING
`key_thumbprint`	Full SHAKE-256 thumbprint of the current key, used wherever an object pins a key
`controller`	TIP-ID of the party that runs the agent
`display_name`	Name shown to sites and people
`manifest_hash`	SHAKE-256 of the agent's signed manifest (Section 7.5)
`certification`	Optional pointer to a Global Seal of Trust certification of the underlying AI system
`key_custody`	Declared custody class: `hsm`, `cloud_kms`, `tee`, `device_enclave` or `software`
`status`	`active`, `suspended` or `revoked`
`visibility`	`public`, or `private`, in which case the ledger holds the controller, the key and a commitment to the manifest, and the manifest goes only to sites the controller authorizes
`registered_at`	Ledger commit time
`sig_alg`, `hash_alg`	`ML-DSA-65`, `SHAKE-256`

Registration is a new transaction type, AGENT_REGISTERED. It carries two signatures. The controller's TIP-ID signs to claim the agent, and the agent's own new key signs the same body to prove the controller holds it. Without the second signature a controller could register someone else's public key as its own agent. The ledger refuses any key, and any 16-character fingerprint, already used by another agent, so identifiers never collide. Where two registrations with the same fingerprint reach different parts of a partitioned ledger at once, the one with the lower transaction identifier wins and the other is void. For a personal TIP-ID the controller's signature needs a fresh biometric check like every other TIP signature. For an organization TIP-ID it comes from an authorized roster member. Each controller has a registration rate limit, so one TIP-ID cannot flood the ledger with agents.

7.4 Key custody, and keeping the key away from the model

An agent's private key must never be visible to the language model that plans the agent's actions. A model that can read its own key can be talked into printing it.

The design puts signing in a separate component, the **agent signer**, which holds the key in a hardware security module, a cloud key management service, a trusted execution environment or a device enclave. The model asks the signer to sign a proposed request. Before signing, the signer checks everything it can evaluate on its own, which is the verb, the audience, the resource pattern

and the agent-side constraints such as validity, rate and amount. It refuses anything outside scope. For a step-up verb with no approval attached, it signs only a request in challenge mode (Section 9.1), which a site never executes. Checks that depend on the site, such as the site's own mapping of an endpoint to a verb, its declared environment for a resource and its irreversibility flags, belong to the site. A hijacked model therefore cannot obtain a signature for an action its mandate plainly excludes, and the receiving site checks everything again regardless.

The signer only helps if it is the agent's only way out. The agent process must run where its network egress passes through the signer or a policy proxy, and it must not hold ordinary API keys or bearer tokens for sites that accept TIP. A site that serves agents declares `tip_required` for each endpoint class, and on those endpoints bearer tokens and API keys alone never authorize a destructive, financial, safety-relevant or identity-changing action. The PocketOS deletion would have been stopped only where both conditions held. A token lying around would have been refused by the provider, and the agent would have had no path to the provider except through its signer.

The custody class goes in the Agent Record so sites can set policy on it, for example accepting only `hsm` or `tee` agents for payments. The declaration is a claim made under the controller's signature. Where a platform can produce hardware attestation, the RFC should allow the attestation to be attached, and sites that care can require it.

7.5 The agent manifest

The manifest is a signed document that states what the agent is and what it says it does. It gives the agent's name, operator and purpose, the AI models and versions it uses, the tools it calls, and a contact for abuse reports. Its hash goes in the Agent Record, so a changed manifest is visible as a new hash on the ledger.

Public manifests must not list software versions. A list of the exact framework and tool versions an agent runs is a map for attackers, especially for agents in hospitals and utilities, so version detail goes to relying sites under disclosure, and agents in critical sectors should register with `visibility: private`.

A signed manifest proves who published it, not that it is honest. Research published in March 2026 showed that A2A Agent Cards, which are signed, can still describe an agent falsely. The manifest is therefore a public statement the controller can be held to. A site, a principal or anyone who can show a manifest is false can open a dispute through TIP's existing adjudication pipeline (Section 16), and an upheld dispute counts against the controller.

7.6 Key rotation and loss

The controller rotates an agent key with AGENT_KEY_ROTATED, signed by the controller and by the new key, for the same reason as registration. A rotation takes effect only after a published notice delay, during which the old key keeps working and the principal's TIP client is told. The delay stops a

compromised controller, such as a breached agent vendor, from silently swapping in its own keys under every customer's mandate at once. Revocation is never delayed. If an agent key is suspected stolen, the controller revokes it at once with AGENT_REVOKED, and every mandate naming that agent stops working at every site that checks status (Section 13).

Mandates name a `tip://agent/` identifier so that routine rotation does not break them. A principal who wants more can pin a key. A mandate that carries `subject_key`, the full thumbprint of one agent key, is valid only for that key, and a rotation ends it. Clients should pin keys on every mandate that includes a step-up verb.

Unlike a TIP-ID, an agent has no biometric recovery path, because an agent is not a person. The controller registers a new agent.

Principals learn of rotations because their TIP client subscribes, through the global change feed (Section 13.2), to every registered agent named in a mandate it has signed. The ledger itself does not know who holds mandates naming an agent, so the subscription is the notice path.

theailab.org

8. Mandates: delegated authority

8.1 What a mandate says

A mandate is the signed permission slip from pull request 383, given a fixed and checkable shape.

FIELD	MEANING
`mandate_id`	SHAKE-256 of the canonical mandate body excluding `mandate_id` and `signature`
`issuer`	Principal TIP-ID, or the parent agent for a sub-mandate
`parent`	`mandate_id` of the parent mandate, absent for a root mandate
`root`	`mandate_id` of the root mandate of the chain; a root mandate omits it, and verifiers set it to the computed `mandate_id`
`subject`	The agent this mandate is for, as a `tip://agent/` identifier or the full thumbprint of an ephemeral key
`subject_key`	Optional full thumbprint pinning one agent key (Section 7.6)
`audience`	The receiving sites allowed, as origins or TIP-IDs of sites
`typ`, `v`	`tip-mandate/1` and the schema version (Section 6.4)
`account`	The site-specific account or resource owner the root issuer asserts, linked to the root issuer's TIP-ID at that site (Section 8.7)
`actions`	Allowed verbs from the Verb registry (Section 6.5)
`resources`	Allowed resources, as path prefixes or resource identifiers
`constraints`	Limits from the closed constraint vocabulary (Section 8.3)
`step_up`	Verbs, from the same vocabulary as `actions`, that need a fresh human approval each time
`step_up_amount`	Value per action above which any money-moving action needs step-up
`step_up_total`	Cumulative value within a window above which step-up applies, so a large payment cannot be split into small ones

thetalab.org

FIELD	MEANING
`approval_policy`	Who may approve step-up actions, how many, and what happens if nobody answers (Section 9.4)
`on_status_unavailable`	What the site may do when fresh status cannot be obtained: `refuse` or `safe_state_only` (Section 13.5)
`epoch_scope`	Which stop scopes this mandate answers to (Section 13.6)
`purpose`	A short description of the task written by the requester, at most 200 characters, shown labeled as unchecked (Section 8.2)
`issued_at`	When the issuer signed it; refused if later than the verifier's clock plus the allowed skew
`issuer_epoch`	The issuer's revocation epoch when it signed (Section 13.1)
`signers`	For organization mandates, the roster members who signed, checked against the organization's roster on the ledger (Section 8.2)
`not_before`, `expires`	Validity window
`max_depth`	How many more times this authority can be passed on; 0 means no sub-delegation
`status_ref`, `status_index`	The status list and position that record this mandate's revocation (Section 13.2)
`max_status_age`	The oldest revocation status a verifier may rely on for this mandate
`nonce`	Random value so two otherwise identical mandates differ
`sig_alg`, `hash_alg`	Algorithm labels inside the signed body

8.2 Signing a mandate is a human act

TIP's standing signing rule, adopted by The AI Lab in October 2026 and stricter than the PRF wording in Section 15.1 of the TIP Specification, is that every signature by a TIP-ID private key needs a fresh fingerprint, face or security-key check, with no unlock window and no cached key. The RFC must bring Section 15.1 into line with it. Throughout this paper, a fresh biometric check means user verification on a device-bound, non-exportable key at NIST SP 800-63B-4 authenticator assurance level 3. Fingerprint and face unlock that key; they are never a factor on their own. Because SP 800-63B-4 Section 3.2.3 requires that a non-biometric option always be available, a hardware security key with a PIN carries exactly the same standing, and people who cannot use biometrics are never shut out. Voice is never accepted. Federal deployments also accept PIV and derived PIV

credentials, as OMB M-22-09 requires. A mandate is signed under that rule. One approval produces one signature over one mandate.

A mandate is also a standing permission. It lets software act later without asking again, which is exactly what makes it useful and exactly what makes it dangerous. Three rules keep that honest.

1. **The TIP client renders the mandate, not the requester.** The person sees the mandate in plain words in their own TIP app or browser extension, generated from the signed fields, never from text supplied by the site or agent asking for it. "This agent may update pages under example.com/blog until 6 pm Friday. It may not delete anything. It must ask you before publishing." What they see is what they sign. The requester's `purpose` text appears only below the generated rendering, labeled "Description from the requester, not checked", with no links, and it never counts as an instruction to the agent.
2. **The requester is shown by its verified identity.** If a site or agent asks a person for a mandate, the client shows the requester's TIP-ID and bound domain, so a phishing page cannot pass itself off as a known service.
3. **Mandates have ceilings.** Clients refuse to sign root mandates that last longer than a published maximum, and the maximum is shorter for higher-risk verbs. Long-running agents renew mandates, which keeps a human returning to the decision.

Keys and history follow one rule. A mandate authorizes a new action only if, at the moment the site decides, the key that signed the mandate is still its issuer's key in effect. When a principal rotates or recovers a TIP-ID key after a compromise, or a parent agent's key rotates, outstanding mandates signed by the old key stop authorizing new actions and must be signed again. Routine rotations follow the continuity rule in Section 13.8. Someone holding a retired or stolen key therefore gains nothing by signing a mandate and backdating it, because any action under it is decided after the rotation, when the old key is no longer in effect. Auditors check each past action against the key in effect at its decision time, as TIP already does for every historical signature, so routine rotation never turns old, lawful actions into failures.

For an organization TIP-ID, the mandate is signed under the organization's TIP-ID by an authorized member of its roster. Today TIP keeps organization rosters off the ledger, which would leave a court or auditor unable to tell who could sign on a given day. This design therefore adds ROSTER_UPDATED, signed by the organization's current roster quorum, which lists members by TIP-ID, their roles (`mandate_signer` , `approver` , `security_contact` , `roster_admin`), per-role thresholds keyed by verb and amount, and the time each version takes effect. An organization that does not want its staff list public publishes members as salted commitments and reveals an entry to a verifier when needed. Organizations can require more than one roster member to sign mandates above a threshold they set, such as any mandate that includes `delete` on production systems or `purchase` above a value. Such mandates list the members in `signers` and carry each member's

signature. Verifiers check every member's key against the roster in effect when the mandate was signed and when the action is decided, so a removed member's signature stops counting.

8.3 The closed constraint vocabulary

Constraints come from a closed list published in the TIP Agent Profile. A verifier that meets a constraint type it does not know must refuse the mandate. The list is closed and declarative on purpose. A general policy language, such as the Datalog in Biscuit, would need a logic engine at every verifier and would make "is this child narrower than its parent?" hard to decide. A closed list keeps verification fast, predictable and the same everywhere.

CONSTRAINT	MEANING	HOW NARROWING IS CHECKED
`time_window`	Allowed hours, as a stated time zone plus local hours and the IANA time zone database version used, expanded into UTC intervals	Child's UTC intervals inside parent's, checked over the mandate's whole validity period with the stated database version
`max_uses`	Total actions allowed under this mandate per audience site	Child less than or equal to parent
`rate`	`count` actions per `window_seconds`	Child's count less than or equal to parent's and child's window greater than or equal to parent's; converting to a per-second rate is not allowed, because 86,400 a day and one a second are not the same limit
`amount_max`	Largest value per action	Same currency or asset, child less than or equal to parent
`amount_total`	Largest cumulative value per audience site	Same currency or asset, child less than or equal to parent
`data_classes`	Categories of data the agent may read or write, such as `public`, `internal`, `personal`, `financial`, `health`	Child set inside parent set
`regions`	Where the action may take effect	Child set inside parent set
`environment`	Labels such as `staging` or `production`, matched against the environment the site declares for the target resource	Child set inside parent set
`recipients`	For messages and payments, the allowed recipients	Child set inside parent set

Amounts are strings of integer minor units with an ISO 4217 currency code, or, for assets that are not money such as shares, tokens or drug quantities, a registered asset type and unit. Sites never

convert between currencies; a mismatch is a refusal. The site, not the agent, derives the value of an action from the request body. All numbers in TIP objects are integers within plus or minus 2 to the power 53, set-valued arrays are sorted by UTF-8 bytes with duplicates refused, and object keys are ordered by UTF-8 bytes as TIP canonical JSON requires.

`environment` exists because of incidents like PocketOS. Real APIs serve staging and production resources from the same address, so the site declares the environment for each resource, not for its whole origin, and works it out from the target resource when it decides. A resource whose environment the site cannot determine is treated as `production`. A mandate for a staging task that says `environment: [staging]` is then refused on any production resource, as long as the site also refuses non-TIP credentials for that action (Section 7.4).

8.4 Narrowing rules

A sub-mandate is valid only if it is narrower than or equal to its parent on every axis. The verifier checks each one.

1. `actions` is a subset of the parent's.
2. Each `resources` entry falls inside a parent entry. Resource patterns are path prefixes matched on whole path segments, so `/admin` does not cover `/administrator`. Paths in mandates and in requests are normalized under RFC 3986 before matching, and any path containing a dot segment (`.` or `..`), an encoded slash or an encoded dot is refused outright. The site matches against the same normalized path its router uses. Regular expressions are not allowed, because subset checks on them are not reliably decidable.
3. `audience` is a subset of the parent's.
4. Every constraint is narrower or equal, by the rule in the table above. A constraint present in the parent cannot be missing from the child.
5. `step_up` is a superset of the parent's, and `step_up_amount`, where present, is less than or equal to the parent's. A child can add approval requirements, never remove them.
6. `not_before` is no earlier and `expires` no later than the parent's.
7. `max_depth` is less than the parent's.
8. `issuer` is the parent's `subject`, and the sub-mandate is signed by that subject's key, or by the pinned `subject_key` if the parent has one.
9. `root` equals the parent's `root`.
10. `max_status_age` is less than or equal to the parent's.
11. The child does not replace any status reference higher in the chain. Every link is checked against its own status entry, so a child pointing at a list that never changes cannot hide its parent's revocation.

The verifier recomputes every `mandate_id` from the canonical body and never trusts the stated value.

Because each hop names its parent and must be narrower, authority flows one way, from principal to agent to sub-agent, and the chain can never loop back. The chain is a directed acyclic graph, the same shape as the TIP ledger itself, and a verifier walks it from root to leaf. Chains longer than a published maximum depth are refused outright.

Narrowing is checked once, but enforcement does not rely on it. At the time of each action the site enforces the constraints of every mandate in the chain, not only the leaf, so a bug in a narrowing check can never turn into extra authority.

8.5 Counters across many sites

Limits such as `max_uses` and `amount_total` are counted by each receiving site for every mandate in the chain, keyed by each `mandate_id`. An action must fit within the remaining budget of the leaf, of its parent and of every link up to the root, so sibling sub-mandates share their parent's budget and a child's lower limit is still honored. The site updates every counter in one atomic check-and-increment, together with the nonce, so two simultaneous requests cannot both spend the last unit.

A limit cannot be counted across different sites without a shared counter, and a design that pretended otherwise would leave a gap. The rule is explicit. A mandate with several audience sites gives each site the full limit. A principal who needs one global limit either issues single-site mandates or names a counter service in `constraints.counter` by TIP-ID and origin. Every site must reserve against it before acting and commit or release afterwards. Each answer is a signed token bound to the request hash and goes into the receipt. A site that cannot reach the counter service refuses.

8.6 Privacy of mandates

A mandate can reveal business relationships and plans. Agents present only the mandate chain the receiving site needs. Where a principal wants to show a site less, the TIP Agent Profile will define a selective-disclosure encoding in the style of SD-JWT, which Verifiable Intent already uses. Selective disclosure can never hide anything a site must enforce. `actions`, `constraints`, `step_up`, `step_up_amount`, `step_up_total`, `approval_policy`, the validity window, `max_depth` and the status fields are always disclosed. Only entries in `audience`, `resources` and `account` that concern other sites may be replaced by salted digests. A verifier that meets a withheld field it needs refuses. The signature still covers every field.

Selective disclosure cannot hide the issuer, because the signature depends on the issuer's key. A personal principal's TIP-ID is therefore visible to every site an agent acts on with a full chain, and sites could compare notes. Section 20 sets out how to limit that.

8.7 A mandate never creates authority

A mandate narrows authority. It never creates it. Without this rule, anyone with a TIP-ID could sign a mandate naming an agent that already has access to a hospital records system, list `/patients/` as a resource, and trick the agent into using it. Every signature would verify and the request would still be wrong, because the person who signed the mandate never had the right to read those records. This is the confused deputy problem, and it is the most dangerous mistake an implementer can make.

The rule is therefore normative. Before it checks scope, the receiving site resolves the root issuer, and for chains acting for an end user the referenced end user, to an account under its own access control, through a prior account-linking step that ties that account to the issuer's TIP-ID. It then authorizes the action against that account's rights. The effective authority of a request is the intersection of three things, the root issuer's own rights at the site, every mandate in the chain, and the site's policy for the agent and its controller. A site must never authorize an action under a mandate using credentials or sessions held by the agent or by its controller. A mandate whose `account` is not linked to its root issuer is refused.

theailab.org

9. TIP Approval for high-risk actions

Some actions should never run on standing permission alone. Step-up applies whenever any one of these is true.

1. The verb is listed in the mandate's `step_up`.
2. The action moves money or assets above the mandate's `step_up_amount`.
3. The receiving site has marked the endpoint irreversible, whatever the mandate says.

The TIP Agent Profile publishes a minimum step-up list. It contains every verb whose risk class in the Verb registry is `irreversible`, `financial`, `safety_relevant` or `identity_security`, which includes `delete`, `publish`, `transfer`, `send_message`, `sign_contract`, `change_credentials`, `change_security_settings`, `change_recovery`, `control_write` and `order_clinical`. Sites apply the minimum list whether or not a mandate repeats it, so a careless or malicious client that leaves it out gains nothing.

A sub-mandate that grants a verb on the minimum list needs its own approval. The parent agent obtains a TIP Approval under the root mandate's `approval_policy` over the hash of the sub-mandate and attaches it as `delegation_approval`, and verifiers refuse the sub-mandate without it.

9.1 The site-signed challenge

When an agent needs a step-up action, it first sends the request in challenge mode, with `intent: challenge`. The site never executes a request in challenge mode. It answers with a **step-up challenge**, signed with the site's receipt key, that names the site's TIP-ID, a `challenge_id`, the HTTP method, the full target URI, the audience, the action, the resource, the `content_digest`, the leaf `mandate_ref`, the approval `meaning`, a `display` list and a short expiry. The site signs the challenge because a hijacked agent must not be able to write its own. A content digest alone is not enough, because many dangerous requests, such as a `DELETE`, have an empty body, and an approval tied only to that digest would match every empty-bodied request.

A person cannot approve what they cannot see. A path such as `/api/v2/orders` or `/fhir/MedicationRequest/88213` tells nobody that the order is for $9,800 or that a dose is changing tenfold, and the client holds only a hash of the body. The `display` list closes that gap. It is an array of typed entries from the Display registry, such as `amount` with value and currency, `payee`, `item` with name, quantity and unit price, `record_count`, `environment`, `subject_ref` and `dose_change` with from, to and unit. The site derives every entry from the request body it will execute and signs the challenge, display included. The client renders only registered display types in fixed templates and refuses a challenge whose action class needs display types that are missing. Payments need `amount` and `payee`, deletions need `record_count` and `environment`, and clinical orders need `subject_ref` and the clinical change. Because the site signed the display, a display that does not

match what the site then executed is the site's own signed misstatement. This is the same dynamic linking that strong customer authentication requires for payments.

The agent forwards the challenge to the approver's TIP app. The client checks the site's signature and TIP-ID, builds the plain-words display from the challenge's typed fields ("Pay $9,800.00 to Acme Supplies Ltd from account ending 4471"), and never shows text the agent or site supplied as free prose. The approver approves with a fresh biometric. The client returns a **TIP Approval** signed by the approver's TIP-ID over the `challenge_id`, the challenge's full hash, the `meaning` and the approver's display name as rendered, and it includes the WebAuthn authenticator data showing user verification.

The agent resubmits the request with the approval attached. The resubmitted request carries a fresh nonce, so the approval binds the challenge rather than the request hash. The site checks that the approval's challenge was issued by this site, has not expired and has not been used, that the approver satisfies the mandate's `approval_policy`, and that every field in it matches the new request exactly. The site marks the challenge consumed in the same atomic step that records the nonce and spends the counters (Section 11, check 12), so two copies of one approval sent at the same moment cannot both succeed. The receipt carries `display_hash`.

9.2 Where the approval happens

An agent that can drive the screen must not be able to approve for its human. Browser agents and computer-use agents often run in the same browser profile or desktop session as the person's TIP extension, and such an agent could open the prompt and click Approve, or type a device password it saw earlier. The rules are strict. A TIP Approval must come from an authenticator the requesting agent cannot drive. A client must refuse to show an approval prompt in a browser profile or operating-system session where an agent holding the same mandate is active, and must route the approval to a second registered device, such as the person's phone or a roaming security key. A password or PIN typed into the same session never satisfies step-up for a verb on the minimum list. A site may require `approval_authenticator_class` of `separate_device` or `roaming`, and sites in health care, finance, government and critical infrastructure should.

9.3 Approval fatigue

Push-approval systems are attacked by flooding a person with prompts until they tap approve. TIP clients and sites must resist this.

1. Sites rate-limit step-up challenges per agent and per mandate. Clients rate-limit prompts per agent and per site, and batch repeated identical prompts into one.
2. A denial is a signal. After a published number of denials within a window, the client offers to revoke the mandate and, by default, suspends it until the principal acts.

3. Every prompt has a “deny and stop this agent” choice that signs a revocation in the same gesture.

This mirrors the human-present mode in Verifiable Intent and the “one assertion, one signature” rule that already governs every TIP signing. A hijacked agent that tries to wipe a database or send money gets as far as a prompt on its principal’s phone, and no further.

9.4 Who approves: policies, dual control and break-glass

Real organizations do not approve with one person. Banks use maker and checker. Power and water operators use a two-person rule for setpoint changes. Hospitals need whichever clinician is on call, not the person who signed the mandate. A mandate therefore carries an `approval_policy`.

FIELD	MEANING
`threshold`	How many approvals are needed, for example 2
`approvers`	TIP-IDs, or roles on the organization’s ledger roster such as `approver` or `on_call_pharmacist`
`separation`	The person whose request or agent started the action may not approve it
`timeout`	How long the site waits before the request lapses
`escalation`	An ordered list of fallback approvers if the first do not answer
`end_user_required`	Where an end user’s money, data or health is affected, that end user must be among the approvers, through a pairwise pseudonymous identifier

Where an end user’s own funds are moved, the end user approves, never only the service acting for them, which keeps the design in line with strong customer authentication rules that require the payer to authenticate.

Emergencies need a door that opens fast and leaves a loud record. A site may list verbs as eligible for **break-glass**, such as emergency read access to a patient chart or an emergency shutdown command. A break-glass approval needs one approver with a fresh biometric and a reason code from the Reason Code registry, carries the meaning `break_glass`, alerts the organization’s security contact at once, and creates a review item that must be closed by a second person within 24 hours. Break-glass never applies to transfers of money, changes to credentials or the registration of content.

HIPAA’s emergency access procedure (45 CFR 164.312(a)(2)(ii)) is a required safeguard, and an emergency may come when the approval service or the clinician’s phone is unreachable. A hospital may therefore issue a **break-glass mandate** in advance, signed under its roster with a fresh check

like any mandate. It names a narrow set of break-glass verbs, such as read access to records for treatment, carries a purpose-of-use code, lasts no longer than a shift, and needs no live approval at the moment of use. Each use is receipted with the meaning `break_glass`, alerts the security contact at once and opens a review item. The break-glass mandate keeps working in degraded mode, because emergency clinical access is declared a safe-state action (Section 13.5).

theailab.org

10. Action requests

10.1 What the agent signs

For each action, the agent signer produces a signature that covers the following.

COVERED ITEM	WHY
HTTP method and full target URI	Stops the request being redirected to another endpoint
`audience`, the receiving site's origin or TIP-ID	Stops a request signed for one site being replayed at another
`content_digest`, SHAKE-256 of the exact body bytes as transmitted, before any decoding	Stops the body being altered after signing; see Section 10.3
`mandate_ref`, the leaf `mandate_id`	Binds the action to one chain of authority
`action` and `resource`	What the agent claims to be doing, checked against the mandate
`created` and `expires`	Freshness; the recommended maximum lifetime is 300 seconds
`nonce`	One-time value; the site refuses any nonce it has seen before
`basis_digest`, optional	Commits the agent's operator to what the agent acted on, without revealing it (Section 10.6)
`intent`	`execute`, or `challenge` to ask only for a step-up challenge (Section 9.1)
`sig_alg`, `hash_alg`	Algorithm labels

On HTTP the signature uses the HTTP Message Signatures framework (RFC 9421), with a `Signature-Agent` header in the style of the Web Bot Auth drafts that resolves to the agent's TIP record. The covered components must include `@method`, `@target-uri`, `content-type`, `content-encoding`, `content-digest` computed over the transmitted bytes before any decoding, `signature-agent`, and `idempotency-key`, `if-match` and `tip-basis-digest` when present. Sites refuse method-override headers on TIP requests. Any header the site uses in deciding is either covered by the signature or ignored. The request `audience` is always one exact origin; a mandate's audience may name a site's TIP-ID, which covers the origins bound to it on the ledger. Other transports, such as MCP and A2A, carry the same signed fields in their own message envelopes (Section 18). The post-quantum registrations these standards need are in Section 10.5.

theailab.org

10.2 Passing the chain by reference

ML-DSA-65 signatures are 3,309 bytes and public keys are 1,952 bytes. In base64 a signature is about 4.4 KB. A request signature fits in an HTTP header, but a chain of three mandates with their signatures is well over 13 KB, which exceeds the 8 KB per-header limit many web servers and proxies apply by default. The design therefore sends the request signature in the header and the mandate chain by reference.

1. On first contact the agent sends the full chain, and the public key of any ephemeral agent key in it, in the request body. A site may instead fetch the chain from a URL the agent gives, but only after the request's signature has verified, only over HTTPS to a public address, and only up to a size cap, so the URL cannot be used to make the site attack its own network or amplify traffic.
2. The site verifies the chain once, caches it by `mandate_id` until it expires, and afterwards needs only `mandate_ref`.
3. Agent public keys are resolved from the TIP ledger or from the mandate, then cached by key thumbprint.

Chains, stapled status and approvals do not fit in headers. They travel in a preflight call to `/.well-known/tip/agent-session`, which returns a short-lived session handle bound to the agent key, or in a multipart envelope, so the site's own API body schema stays unchanged. Receipts come back in the response body or are fetched by reference. This keeps steady-state requests to one extra header of about 5 KB and one signature check. Cached chains are refreshed within `max_status_age`, together with the issuer keys, agent records and rosters they depend on, so a cache never outlives a revocation or rotation it should have seen.

10.3 Matching is not authorizing

TIP's content fingerprint, CNA-2.2, normalizes text so that a published post still matches its registered draft after a platform changes spacing, case or punctuation. It keeps only letters, numbers and marks. Pull request 383 showed the consequence for actions. TIP's own `hashContent` returns the same value for `balance=+100` and `balance=-100`.

That behavior is correct for content matching and dangerous for authorization. Every action request therefore commits to SHAKE-256 of the exact bytes. Where an action also produces content, the receipt can carry the CNA-2.2 hash as well, so the action links to the content's CTID, but authorization always uses the exact-bytes digest.

10.4 Request results

The receiving site must bind what it did, not only what it was asked. The receipt (Section 12) includes a `result_digest`, a hash of the response bytes as transmitted, or of a representation of the affected resource after the action that the site declares for the endpoint, so a site cannot later claim a different outcome without contradicting its own signature.

10.5 Standards this builds on, and what remains

Most of the registry work this design needs was completed in 2026, and the design uses the registered names exactly rather than inventing its own.

1. **HTTP signatures.** IANA added `ml-dsa-44`, `ml-dsa-65` and `ml-dsa-87` to the HTTP Message Signature Algorithms registry on October 23, 2026, referencing the C2SP `httpsig-pq` specification. TIP agents sign RFC 9421 requests with the `alg` value `ml-dsa-65`, or `ml-dsa-87` under the national security profile below.
2. **JOSE, COSE and X.509.** RFC 9964 (May 2026) defines ML-DSA for JOSE and COSE, RFC 9881 (October 2025) defines ML-DSA identifiers for X.509, and RFC 10073 (November 2026) defines composite ML-DSA certificates. DPoP tokens, COSE Receipts and A2A Agent Card signatures use those identifiers.
3. **Transparency.** The SCITT architecture is RFC 9943 and COSE Receipts are RFC 9942, both June 2026. TIP receipts map to COSE Receipts, and witnesses act as SCITT transparency services.
4. **Digests.** SHAKE-256 is not yet in the HTTP Digest Algorithms registry of RFC 9530, so the exact-bytes digest travels as a covered TIP field until it is registered.
5. **Web Bot Auth.** The working group draft `draft-ietf-webbotauth-httpsig-protocol` defines the `Signature-Agent` header and a key directory. Where a verifier still accepts only Ed25519, agents add an Ed25519 signature alongside ML-DSA-65 as a transitional measure. NIST IR 8547, final on October 27, 2026, deprecates such classical signatures after 2030 and disallows them after 2035, and Executive Order 14412 sets December 31, 2031 for federal post-quantum signatures, so the Ed25519 part is labeled transitional and never satisfies a TIP check.
6. **National security systems.** NSA's CNSA 2.0 suite requires ML-DSA-87 rather than ML-DSA-65 for national security systems, with new acquisitions compliant from January 1, 2027. TIP therefore defines a CNSA profile in which every key and signature in a chain uses ML-DSA-87, and a verifier configured for that profile refuses any ML-DSA-65 link.
7. **Validated modules.** Federal deployments need ML-DSA and SHAKE from FIPS 140-3 validated modules with CAVP certificates, and FIPS 140-2 certificates moved to the historical list in September 2026. Conforming signers, sites, gateways and witnesses in regulated deployments name their validated module.

The IETF opened agent work in the second half of 2026. IETF 127, in San Francisco from November 14, 2026, holds the first session of the "audit" BoF (Agent Use of Delegation and Interaction Traceability), meetings of the newly chartered agentproto working group, whose charter calls for agents authenticated independently of the users who delegated to them with independent revocation and auditable actions, and sessions of the dawn, webbotauth and wimse groups. The WIMSE group adopted `draft-ietf-wimse-aims` on agent authentication on November 2, 2026, and the OAuth group adopted `draft-ietf-oauth-delegate-sd-jwt` on October 30, 2026. The AI Lab

theailab.org

intends to bring the mandate and receipt model to the audit BoF and to align mandates with delegated SD-JWT and Rich Authorization Requests, rather than build a parallel format.

10.6 The basis of an action

A receipt proves what an agent sent and what the site did. In an audit or a dispute the next question is why the agent did it. What instructions did it have, which model and version planned the action, and which tool results and documents did it rely on? An Action Request may therefore carry `basis_digest`, a commitment to the material the agent acted on. In this section and in Section 12.8, the agent's operator means the controller of a registered agent. For an ephemeral key it means the controller of the nearest registered agent above that key in the mandate chain. An ephemeral key with no registered agent above it never carries `basis_digest`, because no operator can be named, and a site refuses one that does (Section 11, check 9).

1. **What it commits to.** The agent signer builds a basis manifest, an object with `typ` `tip-basis/1` and a `v`, in TIP canonical JSON. Its `entries` list, in the order the agent received or produced them and each with its `kind`, the principal's instructions and the prompts given to the model, the model identifier and version, each tool call and its result, and each document or page the agent relied on. An item is the exact bytes the agent received or sent. A document or page whose bytes may not be kept is entered by its URL and retrieval time, or by its CTID. Each entry carries a salted commitment, SHAKE-256 over its own random salt of at least 32 bytes followed by the item, so an item and its salt can later be erased without breaking the manifest (Section 12.8, item 6). `basis_digest` is SHAKE-256 over a fresh random salt of at least 32 bytes followed by the canonical manifest. Every digest here has a 256-bit output and is written as `shake256:` followed by 64 lowercase hex characters. Nobody can confirm a guess about what the manifest contains, and two receipts never show that two actions shared a basis. When the field is not used it is absent. On HTTP it travels in the `tip-basis-digest` header, which is then a required covered component (Section 10.1).
2. **Who holds the material.** The agent's operator keeps the manifest, every item it lists and every salt in its own log, under the rule in Section 12.8. None of the basis goes to the receiving site, to TIP nodes or to the ledger.
3. **What the site does.** The site checks only that the field is well formed (Section 11, check 1) and copies it into the receipt unchanged. It cannot check what the digest commits to, because it never sees the material, and the digest's value never affects the decision. A site may require the field for an endpoint class it declares in its published endpoint definitions, and sites that receive clinical or financial recommendations from agents should. It enforces that at check 9 of Section 11 and refuses a request without the field with the reason code `basis_required`.
4. **What it proves.** Covered by the agent's signature, `basis_digest` is the operator's signed statement of what the agent acted on, fixed at the moment of the action. Revealing the manifest, its items and their salts to an auditor or to the confidential panel of Section 16.2

(Section 12.8, item 5) lets them check that statement against the receipt, as Appendix B outlines. It does not prove that the model used that material, or used it well. A false basis is the operator's own signed misstatement and can be disputed (Section 16.2).

`basis_digest` is part of `tip-receipt/1` from version 1 of the TIP Agent Profile. Every conforming site accepts the field and copies it into its receipt, and every conforming verifier and monitor accepts receipts that carry it.

theailab.org

11. Verification at the receiving site

The receiving site runs these checks in this order before anything changes. The order puts cheap checks first, so a flood of junk requests cannot force the site to run expensive signature checks, and it records the nonce only after the signature is proven, so an attacker cannot burn an honest agent's nonces with forged requests.

An agent signature counts as proven only when the key it verifies against is bound to a valid root, either a registered agent's key in effect or a key named by a mandate whose whole chain of signatures verifies to an active TIP-ID. Anyone can invent a mandate naming a throwaway key, so a signature over such a request proves nothing. A request that fails any of checks 1 to 7 gets a plain, unsigned error and nothing is logged, because signing and logging a receipt costs more than verifying and would turn junk traffic into work. From check 8 onward every failure stops the request with a signed refusal receipt that gives the reason. Signed refusals are rate-limited per root issuer and per controller, and above the limit the site returns unsigned errors.

1. **Shape.** Size limits, required fields, known algorithm labels, canonical encoding, no duplicate keys, no unknown fields.
2. **Freshness.** `created` within the allowed clock skew (recommended 60 seconds), `expires` in the future, lifetime within the maximum.
3. **Audience.** The request names this site's exact origin, as scheme, host and port.
4. **Replay pre-check.** The nonce has not been seen within the validity window.
5. **Agent signature.** The request is signed by the key the mandate names or pins, or by the registered agent's key in effect now. When the request carries `tip-basis-digest`, that header is a covered component.
6. **Agent status.** A registered agent's record is active and its controller's TIP-ID is active.
7. **Chain.** Every `mandate_id` is recomputed; each mandate's signature is valid under the key in effect now; the root issuer is a TIP-ID; organization signers are on the roster in effect; every hop obeys the narrowing rules in Section 8.4; depth is within the maximum; no `issued_at` is in the future.
8. **Revocation and expiry.** No link in the chain is revoked or expired, and no link's `issuer_epoch` is below its issuer's current epoch, using status no older than the smallest `max_status_age` in the chain (Section 13).
9. **Scope and underlying authority.** The site works out the verb and resource from its own endpoint definitions, never from the agent's label alone, and refuses the request if the agent's claimed action differs. A `DELETE` sent with the label `update` is refused. The action, resource, audience and the environment of the target resource must fall inside every mandate in the chain, and the constraints of every link must hold. The site then authorizes the action against the rights of the root issuer's linked `account` under its own access control (Section 8.7). A

request that the account itself could not make is refused, whatever the mandate says. Where the site's published endpoint definition requires a basis digest and the request carries none, the site refuses it with the reason code `basis_required`, and it refuses a basis digest from an ephemeral key with no registered agent above it with `basis_not_allowed` (Section 10.6).

10. **Step-up.** If step-up applies (Section 9), approvals satisfying the mandate's `approval_policy`, each for an unconsumed challenge this site issued and matching every field of this request, are attached. A request in challenge mode is never executed; it receives a challenge.
11. **Bytes.** The delivered body hashes to `content_digest`.
12. **Atomic commit.** In one atomic, linearizable step, record the nonce, mark every approval's `challenge_id` consumed, and check-and-increment every counter in the chain. If a challenge is already consumed or any counter is exhausted, nothing is recorded and the request is refused. Nonce, challenge and counter state for an origin live in one linearizable store, or are partitioned so that each `mandate_id` is served by exactly one partition. A site that cannot do this declares a separate origin per region.
13. **Intent, act, receipt.** The site appends a signed `intent` receipt to its log, carries out the action, computes `result_digest`, and appends a signed `final` receipt that names the intent. If the action fails, the final receipt says so and the counters are refunded in the same atomic step. An operation that settles later, such as a payment, returns `accepted_pending`, and its final receipt follows within a declared maximum time.

Checks 7 and 8 hit caches almost every time. A site that has seen a mandate before verifies one ML-DSA-65 signature per request.

theailab.org

12. Receipts, logs and anchoring

12.1 Receipts

Every decision produces a receipt, including refusals, so attempted misuse leaves evidence. A receipt carries `typ` and `v`, the site's TIP-ID, `log_id`, `receipt_key_id`, `phase` (`intent` or `final`) and, on a final receipt, `intent_ref`, the agent, the verb and resource as the site derived them, `content_digest`, `request_hash` (SHAKE-256 of the RFC 9421 signature base followed by the signature bytes), the leaf and root mandate IDs, the decision and a registered reason code, `decided_at`, `result_digest`, `basis_digest` copied from the request when it carried one, `display_hash` for approved step-up actions, the time token for step-up and financial actions (Section 12.5), any counter service tokens (Section 8.5), `status_mode` (`normal` or `degraded`), the receipt's `log_index`, and two facts about what the site knew when it decided. `status_seen` names the version of the revocation status it relied on, and `ledger_seen` names the most recent TIP ledger transaction it had observed. The site signs it with a receipt key registered under its TIP-ID.

The site returns the signed receipt in the response to the agent, synchronously. A site that claims TIP agent conformance must do this. The agent keeps its own copy, so evidence of the action exists in two places held by two different parties.

12.2 The receipt log

Each site appends receipts to a Merkle log. The log uses the tree shape and proofs of RFC 9162, Certificate Transparency version 2, with inclusion proofs bound to the leaf index and tree size and consistency proofs between any two sizes, using SHAKE-256 with the RFC 9162 leaf and node prefixes registered as a TIP hash profile. TIP's current `shared/merkle.js`, which pull request 383 uses, builds a different tree and has no consistency proofs, so it must be replaced before witnesses can work. The receipt format is aligned with SCITT (RFC 9943) so transparency tooling can read it. At intervals the site signs a **TIP Seal** containing the log size and root hash.

12.3 Witnesses stop split views

A site that controls its own log could show one version to one auditor and a different version to another. Independent **witnesses** prevent that. Each log has a published set of at least three witnesses, assigned under rules the AI Trust Council sets rather than chosen by the site, and no witness may share ownership or control with the site, or host the site's log or evidence store (Section 20, item 10). A witness must hold an active organization TIP-ID, so nobody can create witnesses at will. A change to a log's witness set is a WITNESS_SET_UPDATED transaction cosigned by a majority of both the old and the new sets, which keeps the guarantee below intact across the change. A witness cosigns a TIP Seal only after checking that it extends the last TIP Seal it cosigned for that log (a consistency proof). A TIP Seal counts only with cosignatures from a majority of that

log's witness set. Because any two majorities share at least one witness, and an honest witness never cosigns two TIP Seals that do not extend each other, two conflicting histories cannot both be accepted unless a witness signs both, and that pair of signatures is itself public proof of misconduct.

Witnesses do not sign every site TIP Seal one by one. Each witness verifies consistency for every TIP Seal it receives during an interval, puts them in a Merkle tree keyed by log identifier, and signs the tree's root once. A site's witnessed TIP Seal is its TIP Seal plus inclusion proofs under a majority of those signed roots. This keeps a witness's signing work to one signature per interval, however many sites it serves.

Someone has to look. A **monitor** is a conformance class that fetches every anchored seal, checks consistency across all of them, and publishes any fork it finds as a signed proof. The AI Lab and at least two independent parties run monitors, and anyone may.

If a log cannot obtain a witnessed seal within its maximum merge delay, because too many witnesses are unreachable, the site stops step-up actions until it can, and continues only the actions it has declared as safe-state actions (Section 13.5).

12.4 Anchoring on the TIP ledger without overloading it

Witnessed roots are anchored on the TIP ledger, which gives every log a single public history and an independent clock. Anchoring every site's TIP Seal as its own transaction would not scale. With 100,000 sites sealing every five minutes, that alone would be 28.8 million transactions a day, about 333 per second, more than three times the 100 per second at which the TIP specification plans to move to full decentralization.

The design anchors in two tiers. Aggregators, which are typically the witnesses themselves, anchor one signed root per interval with a new LOG_CHECKPOINT_ANCHORED transaction. A thousand aggregators anchoring every five minutes produce 288,000 transactions a day, about 3.3 per second, however many sites and actions sit underneath. Each aggregator tree is keyed by log identifier and holds at most one TIP Seal per log per interval, so for any log the anchored history is a single sequence, and two different TIP Seals of the same size for the same log, anchored anywhere, are public proof that the log forked.

A site's inclusion proof runs from receipt to site TIP Seal to witness root to ledger transaction. Every conforming log has a **maximum merge delay**. A receipt must appear in an anchored TIP Seal within that time. The default is 15 minutes and the upper limit 24 hours, and each log states its value in its registration. Busy sites TIP Seal by count and by time, whichever comes first; low-traffic sites may seal less often, but never beyond the maximum merge delay.

theailab.org

12.5 Time

A receipt's `decided_at` comes from the site's clock, which pull request 383 lists as a limit. Ledger time means the commit time attested by the committing quorum's signatures, the coordinator in TIP's Phase 1 and the committee in Phase 2, never the timestamp a transaction's signer chose. The ledger bounds `decided_at` from both sides, and `ledger_seen` must be no older than 60 seconds at `decided_at`, so a site cannot weaken the lower bound by naming an old transaction. For step-up and financial actions the site also obtains, before committing, a time token signed by a threshold of TIP time signers over the `request_hash`, and puts it in the receipt. It cannot be earlier than the ledger time of the `ledger_seen` transaction the receipt names, because the site claims to have already seen that transaction. It cannot be later than the ledger time of the anchor that includes the receipt, and that anchor must arrive within the maximum merge delay. A verifier checks both bounds and the request's own `created`. The width of the window is set by how recently the site last read the ledger and by the merge delay, so high-value sites read the ledger often and keep both short.

12.6 Withheld receipts

A site could act and never log the receipt. Three things narrow that gap. The receipt is returned synchronously, so an agent that received a response without a receipt knows at once and records the fact in its own log. After the maximum merge delay, the agent, or anyone holding a receipt, asks the site for that receipt's inclusion proof against an anchored TIP Seal; a site that cannot produce it has been caught withholding or forking, and the signed receipt is the evidence. Agent operators can also run their own receipt logs, witnessed and anchored the same way, so the agent side holds a signed request and a timestamped record of any missing receipt. Withholding cannot be prevented by cryptography alone, but under this design it leaves evidence.

12.7 Receipt and witness keys

A stolen receipt key lets an attacker forge receipts for new decisions and forge the challenges that drive approvals, so these keys need a full lifecycle. Receipt and witness keys are held in FIPS 140-3 Level 3 or equivalent hardware, declared as `key_custody` when registered. RECEIPT_KEY_ROTATED replaces a key on schedule. RECEIPT_KEY_REVOKED carries `compromised_since`, and receipts or challenges signed after that time are invalid unless they appear in a TIP Seal anchored before it. Witness keys follow the same rules through WITNESS_KEY_REVOKED.

12.8 Keeping the evidence

A digest proves that a record still matches what was signed. Once the record is gone, the digest proves only that something was signed. Every party therefore keeps the originals behind the digests it signs or relies on, and TIP holds none of them.

1. **Who keeps what.** The receiving site keeps each signed request in full as it received it (the covered components, the signature input and the signature) and its body, the mandate chain exactly as presented with any disclosures, delegation approvals, approvals with the challenges and displays it signed for them, stapled status statements, counter and time tokens, the results it produced with the representation of the affected resource that `result_digest` covers (Section 10.4), and its signed receipts. The agent's operator keeps its signed requests in full, the chains it presented, its copies of receipts, and the basis manifest, items, salts and any erasure records behind each `basis_digest` (Section 10.6). Principals and approvers should keep copies of the mandates and approvals they signed, so they never depend on the other side's copy in a dispute, and their TIP client keeps them where it can. These copies are advised, not required, and carry no retention period.
2. **Where.** In systems the party controls, its own or a provider's it contracts with, in the region its law requires. TIP does not move them, and no witness, node operator or aggregator receives them in that role. A party that uses a hosted evidence store stays responsible for keeping its evidence (Section 20, item 10). Inside an enclave, originals stay inside it except for disclosure under item 5 (Sections 18.1 and 21.3).
3. **For how long.** At least the retention in Section 6.6, which binds sites and operators, and longer where law or contract requires. A party that deletes an original early, other than for erasure under item 6, can no longer prove that action.
4. **How.** Originals are stored as the exact bytes their digests cover, as transmitted before any decoding, so they can be matched later. They are encrypted at rest, with access to them logged. Personal values and their salts are stored apart (Section 20, item 6).
5. **Disclosure.** Originals go only to the parties to a dispute and its adjudicators under Section 16.2, to auditors and regulators entitled to them, or under legal process. Evidence that contains regulated data, such as health records or customer financial data, and every basis manifest with its items, goes only to the confidential panel of Section 16.2, never to community jurors. The recipient checks what it received against the digest in the signed receipt.
6. **Erasure.** Erasure never overrides a legal duty to retain or a legal hold. Where it applies, deleting a personal value and its salt meets the duty, and the receipt stays valid but no longer proves that value. A value inside a request body is erased by deleting the whole body, together with the random value that Section 20, item 6, requires in bodies that carry short personal values, so the receipt's `content_digest` can no longer confirm a guess. Each erasure is recorded with the time it happened, signed by a system key the party names, and anchored in the party's own witnessed log (Sections 12.2 and 12.6). A party without such a log fixes the record's time another way it can later show, such as a TIP content registration of the record's hash. In a dispute, an original missing before its retention period ended counts against the party that held it, unless its erasure record was fixed in time before the dispute was filed.
7. **Values nobody may keep.** Some rules forbid keeping certain values at all, such as card security codes under PCI DSS. Endpoints that handle such values must receive them by token or by

reference, so the signed body never contains them. An endpoint that receives such a value in a signed body does not conform, because the receipt's `content_digest` could confirm a guess even after the body is deleted.

theailab.org

13. Revocation and TIP Stop All

13.1 What can be revoked

TRANSACTION	EFFECT
MANDATE_REVOKED	Revokes one mandate by `mandate_id`, and every sub-mandate under it
MANDATE_EPOCH_ADVANCED	Raises the issuer's integer revocation epoch; every mandate carrying a lower `issuer_epoch` stops working, in one transaction; this is TIP Stop All
AGENT_REVOKED	Revokes a registered agent; every mandate naming it stops working
AGENT_SUSPENDED	Suspends a registered agent for new actions, by its controller or by an emergency panel (Section 13.7)
MANDATE_EPOCH_RESTORED	Lifts a suspension-type stop, restoring the previous epoch only (Section 13.6)
Existing TIP revocations	A revoked or device-revoked TIP-ID can no longer authorize; mandates it issued stop working

TIP Stop All uses an integer epoch, not a date, because a date inside a mandate is chosen by whoever signs it, and a thief with a stolen key could sign with a future date. An epoch can only be raised by a ledger transaction, so it cannot be predicted or skipped.

Revocation is signed by the issuer of the revoked mandate, by any issuer higher in the chain, or, for agents, by the controller. An organization can also let a designated security contact on its roster stop agents without the usual signing quorum, because stopping an agent should be easier than starting one. A stop signed by a single security contact suspends rather than revokes (Section 13.6), so one compromised contact cannot permanently destroy every mandate an organization holds.

13.2 Checking status at scale

Asking a ledger node about every link of every chain on every request would cost the network money on every action (Section 22), make every site depend on a live connection, and tell node operators which principals use which sites. The design uses four layers.

1. **Short lifetimes.** Mandates for routine work last hours, not months. Expiry does most of the revoking.
2. **Status entries.** Every mandate that can be revoked on the ledger carries a `status_ref` and a `status_index`, a position in a status list in the style of the W3C Bitstring Status List, in which

each mandate holds one bit. Lists are shared pools of at least 131,072 entries holding many issuers' mandates, positions are assigned at random, and `status_ref` names the pool, never the issuer, so a status reference does not reveal who signed the mandate. Issuers obtain positions in blocks of at least 4,096 per ledger transaction, so status allocation does not add a ledger write per task. Sites fetch whole pools or rely on stapling, never per-entry queries. One list covering 131,072 mandates is 16 KB before compression. An ephemeral agent key cannot hold a list, so sub-mandates it issues rely on short expiry and on revocation of their parent.

3. **Stapled status.** The agent attaches a recent signed statement of the current status of every link in its chain, in the way TLS servers staple certificate status. Not any node may vouch. Status statements are valid only when signed by a threshold of status signers, by default two of three with at least two independent operators in different jurisdictions, registered by STATUS_SIGNER_REGISTERED transactions that the AI Trust Council approves. Each status list version carries a strictly increasing version number, the hash of the previous version and the ledger transaction that produced it, and status list roots are anchored on the ledger at a fixed interval. The site checks the threshold signature and the statement's age instead of looking anything up. This removes most status lookups and keeps node operators from learning which sites a principal's agents visit.
4. **A global change feed.** Nodes publish one signed feed of all revocations and epoch changes. Sites that prefer not to rely on stapling subscribe to the feed and keep their own current view, with no per-principal polling.

Each mandate states `max_status_age`. A site may rely on status, stapled or cached, no older than the smallest `max_status_age` in the chain. `status_seen` in a receipt names the status version and carries its threshold signature, and a verifier checks that no anchored status version issued between the one the site used and the decision, older than the allowed age, showed the link revoked. A mandate for routine reads might allow ten minutes; a mandate that allows payments might allow thirty seconds.

13.3 Fail closed

If a site cannot obtain status fresh enough for the mandate, it refuses ordinary actions. There is no "allow if the status server is down" mode for actions that change something, and step-up actions always need status checked at the time of the action. A site may serve reads under stale status only if the mandate's `max_status_age` permits it. Section 13.5 sets out the one exception, safe-state actions, which exists because refusing everything can itself cause harm.

13.4 History stays true

An action decided before a revocation took effect stays valid afterwards, so revoking an agent never rewrites history. The test is what the site was entitled to know. An action is valid if, at the site's decision, the status version named in the receipt's `status_seen` showed no revocation, that

status was within the chain's `max_status_age` of `decided_at`, and the keys were the keys in effect at `decided_at`. A site that acted on status older than it was allowed to use, or ignored a revocation its own `status_seen` already contained, carries the responsibility for that action.

13.5 Safe state and degraded operation

A design that refuses everything when it cannot reach the network is safe for a bank's marketing site and dangerous for an intensive care unit or a water treatment plant. Lost availability in clinical and industrial settings is a safety failure, not an inconvenience. This section is therefore normative for every site in health care, energy, water, transport and government emergency services, and recommended everywhere.

1. **Safe-state actions.** A site declares, in its published endpoint definitions, the actions that move the controlled system to a documented safe condition, such as stopping a pump in a defined way, holding a process at its last safe setpoint, or read-only access to a patient record for clinical care. These are the only actions it may allow under degraded status.
2. **Mandate choice.** A mandate's `on_status_unavailable` is either `refuse` or `safe_state_only`. Under `safe_state_only`, when status older than `max_status_age` is all the site can get, the site refuses everything except declared safe-state actions.
3. **Loud records.** Every action taken under degraded status gets a receipt with `status_mode: degraded` and a reason code, and the site files each one for review by a person within 24 hours.
4. **No single status source.** Status must be obtainable from at least three independently operated status signers in different jurisdictions (Section 13.2). No single operator, The AI Lab included, may be required for a status answer.
5. **The ledger today.** Until TIP reaches its Phase 2 committee, ledger commits pass through one coordinator. No deployment in a safety-critical sector may depend on ledger liveness for safe-state actions, and this paper describes the current ledger as coordinator-committed.
6. **Stops hand control back.** In industrial settings a stop returns the process to traditional automation or manual operator control. It never halts the plant. This follows the joint guidance led by CISA with partner agencies in Australia, Canada, Germany, the Netherlands, New Zealand and the United Kingdom, Principles for the Secure Integration of Artificial Intelligence in Operational Technology (December 3, 2025), which calls for failsafes that revert to manual control, operation without continuous internet or cloud access, and AI identities logged separately from other identities. IEC 62443-3-3 says access controls shall not prevent essential functions, including safety functions, from operating, and OMB M-25-21 recognizes that human approval is not practicable for an automated safety mechanism that must act at once.
7. **Never in the loop.** TIP is never placed inside a safety instrumented function or a real-time control loop. It governs requests that reach a control system through an enforcement gateway (Section 18.1) and stays out of the path of protective functions.

theailab.org

13.6 Scoped stops, suspension and restoration

TIP Stop All is powerful, which makes it a target. Three rules keep it from becoming a weapon.

1. **Scope.** MANDATE_EPOCH_ADVANCED carries `scope`, which is `all`, one agent, one verb, or one environment label. Mandates carry `epoch_scope`, so an organization can stop every agent touching production, or one compromised agent, without stopping its whole fleet.
2. **Suspension first.** A stop signed by a single security contact suspends the affected mandates. The suspension lapses after 72 hours unless the organization's normal signing quorum confirms it, and the quorum can lift it at once with MANDATE_EPOCH_RESTORED, which can only restore the previous epoch. A stop signed by the full quorum revokes.
3. **Finishing safely.** An operation already under way when a stop arrives, such as a database migration, a valve sequence or a transfer debited but not yet credited, may still run the steps the site declares as compensating or safe-state steps for that operation, for up to the compensation window (300 seconds by default). Every such step is receipted as `compensating`.

Stop All does not answer a stolen principal key. A thief who holds the key can read the new epoch and sign fresh mandates at it. A principal who suspects key theft must also revoke the device or recover the key under TIP's existing transactions, and clients offer all three in one flow.

13.7 Stopping a harmful agent whose controller will not

If a registered agent is attacking people and its controller is unreachable or is the attacker, someone else must be able to stop it. AGENT_SUSPENDED may be signed by the controller, by an emergency panel the AI Trust Council appoints, acting by threshold, for up to 72 hours pending adjudication, or under a court or regulator's order recorded as a governance transaction. Sites treat a suspended agent as revoked for new actions.

13.8 Key events and outstanding mandates

Routine key events must not halt every agent at once. KEY_ROTATED carries `continuity`, either `routine` or `compromise`. After a routine rotation, mandates signed by the old key keep working until their own expiry, capped at seven days, and clients prompt the principal to re-sign them. After a compromise rotation, a device revocation or a key recovery, they stop at once. When a Verification Provider's accreditation is revoked and the revocation cascades to the TIP-IDs it verified, those TIP-IDs cannot issue new mandates, and existing mandates continue until a grace period the AI Trust Council sets ends, unless the cascade is marked as fraud, in which case they stop at once. Before any action that would end outstanding mandates, the client tells the principal how many will stop.

theailab.org

14. Agent-made content and Verifiable Assets

14.1 Who signs what an agent makes

When an agent creates content, the content needs a CTID like any other Verifiable Asset. A Verifiable Asset is any piece of digital content that carries a CTID (Content TIP ID): a permanent, signed public record of who made it, when, how it was made, and a fingerprint that shows whether it has changed since. (See theailab.org/verifiable-assets.)

The agent cannot sign as its principal, because the principal's key never leaves the principal's device and every signature with it needs a fresh biometric. The design adds a new attribution mode, `delegated`, alongside TIP's existing `self`, `employed` and `hosted` modes.

1. **A verb of its own.** The agent's mandate must include the `register_content` verb with the TIP ledger as its audience. An agent allowed to update a website is not thereby allowed to register content in its principal's name.
2. **Registered agents only.** Only registered agents can register content in this mode, because an ephemeral key has no public record to resolve.
3. **Checkable by every node, without publishing the chain.** The registration carries hashes of the mandate chain with a selective-disclosure proof that reveals only the `register_content` verb, the ledger audience, the validity window and the signatures, so every ledger node can check the authority at commit time and the ledger never stores the principal's business relationships or purposes.
4. **Honest attribution.** The CTID's author segment comes from the agent identifier, and readers see "registered by an agent acting for" the principal, not "created by" the principal.
5. **No race for priority.** A `delegated` registration never wins first-registration priority over a `self` registration of the same content, so an agent cannot register other people's work by the thousand to claim it first.

14.2 The origin floor

Content registered in `delegated` mode can never carry the OH (Original Human) origin code. The minimum is AA (AI-Assisted), and AG (AI-Generated) applies whenever the agent did the substantive creative work, which is the usual case. This follows TIP's conservative labeling rule and closes an obvious laundering route, in which AI-generated content is pushed through an agent to pick up a human label. A principal who wrote the content themselves registers it in `self` mode with their own key, as they do today.

theailab.org

14.3 Linking actions and content

The content registration carries the hash of the action request that produced it, and once the CTID is committed the agent or site can publish a short signed link from the receipt to the CTID. A reader of the content can find the action that produced it, and an auditor of the action can find the content it produced.

theailab.org

15. Protecting the agent's inputs

TIP cannot stop a model from being misled, but it can give agents better information about what they read.

15.1 Instruction provenance

Agents should treat instructions according to where they come from. Instructions from the principal arrive only through the principal's own signed channel. The mandate's `purpose` was written by the requester and is never treated as an instruction. Everything else, including web pages, emails, documents and other agents' outputs, is data, not commands. Where that data carries a CTID, the agent can see whether it was registered, by whom, with what origin code and at what trust tier, and agent frameworks can use that as one input in deciding how much weight to give it. Text that tries to grant the agent new authority is never authority, because authority comes only from a signed mandate.

15.2 Signed tool and server manifests

Tool poisoning hides instructions in tool descriptions, and "rug pulls" change a tool after a user approved it, because most clients do not compare tool definitions between sessions. Under this design, an MCP server or tool provider registers its tool manifest as a TIP content record signed by its TIP-ID. The agent pins the manifest's hash when the principal approves the tool. If the description changes, the hash changes, and the agent signer refuses to act through that tool until the principal approves the new version.

15.3 Agent cards and manifests are claims

A2A Agent Cards and TIP agent manifests prove who published them, not that they are true. TIP adds two things a signed card lacks. The publisher is a verified TIP-ID, so a false card has an accountable author. False claims can be disputed through TIP's adjudication, and an upheld finding stays on the public record.

theailab.org

16. Accountability and reputation

16.1 Accountability stays with people and organizations

An agent has no trust score of its own. Scores belong to TIP-IDs, and every TIP-ID belongs to one verified person or organization. This is deliberate. If agents had independent reputations, a controller with a ruined agent would register a fresh one. Because conduct traces to the controller and the principal, and deduplication makes a second TIP-ID hard to obtain, there is no fresh start to buy.

16.2 Disputes about agent conduct

A receiving site, a principal or an affected third party can open a dispute about an agent's conduct, attaching receipts and inclusion proofs as evidence. Disputes run through TIP's existing three-stage pipeline. The kinds of finding the RFC should define include a false manifest, a false basis digest (Section 10.6), conduct outside a mandate that the controller's own signer should have refused, and a controller ignoring revocation. Any score effect on a controller or principal TIP-ID must go through a SCORE_UPDATE transaction with a stated reason, under TIP's single-channel rule. The size of those effects is a decision for the AI Trust Council, and this paper proposes none. Disputes whose evidence contains regulated data, such as health records, controlled government data or customer financial data, and disputes that rely on a basis manifest (Section 10.6), never go to anonymous community jurors. They go to a confidential panel of accredited experts bound by confidentiality, and only the finding is published. Organization TIP-IDs are linked to the personal TIP-IDs of their beneficial owners, and conduct findings attach to those owners as well, so a fresh shell company is not a fresh start.

16.3 What refusals do not mean

A refusal receipt is not evidence of misconduct by itself. Sites refuse for many honest reasons, and a hostile site could refuse valid requests on purpose. Refusal counts are shown for context and never feed a score. What a refusal does prove is that a specific, signed request was made, because a site cannot forge an agent's signature on a request the agent never sent.

16.4 Conduct signals for sites

Sites still want to know whether an agent behaves. The public record supports that without a new score. A registered agent's age, its controller's TIP-ID and trust tier, any certification of the AI system behind it, and its upheld disputes are all visible. Sites combine those with their own experience of the agent.

theailab.org

17. Threat model

The table lists the attacks the design has to stop, how it stops them, and what remains. “Residual” is stated plainly so nobody mistakes a narrowed risk for a closed one.

#	THREAT	ATTACK	DEFENSE	RESIDUAL
T1	Impersonation	Agent claims to be a known agent	Request must verify against the registered agent key or the mandate-named key	Depends on the integrity of the ledger copy the verifier uses and on key safety
T2	Forged authority	Company A signs a mandate in company B’s name	Root signature must verify against B’s registered TIP-ID key	None
T3	Stolen mandate	Thief replays a mandate with their own key	Mandate names one agent key or registered agent; thief’s key is not it	Where the mandate names a registered agent without a pinned key, a rotation by a compromised controller changes the key; see T41
T4	Stolen agent key	Thief signs requests as the agent	Keys in HSM, KMS, TEE or enclave; short mandates; AGENT_REVOKED; MANDATE_EPOCH_ADVANCED	Works until revoked or expired; short lifetimes bound the window
T5	Replay	Same request sent twice	Nonce plus short lifetime	None
T6	Cross-site replay	Request for site A replayed at site B	`audience` inside the signature	None
T7	Scope escalation	Agent acts outside its mandate	Agent signer refuses; site refuses	Sites that skip checks or map endpoints to the wrong verb; conformance testing (Section 23)
T8	Escalating sub-delegation	Agent grants a helper more than it holds	Narrowing rules on every axis; segment-aware prefixes; no regex	None
T9	Chain loops and depth bombs	Circular or very deep chains to exhaust verifiers	Parent and root links; `max_depth`; hard maximum chain length	None

theailab.org

#	THREAT	ATTACK	DEFENSE	RESIDUAL
T10	Confused deputy	Agent tricked into using its authority for someone else	Mandate only narrows the root issuer's rights at the site; agent and controller sessions never authorize mandate actions (Section 8.7)	Within-scope misuse remains possible; see T21
T11	Body tampering	Agent signs +100, -100 is delivered	Exact-bytes SHAKE-256 digest, never the normalized content hash	None
T12	Time-of-check versus time-of-use	Site checks one body and executes another	Digest checked on the bytes executed; `result_digest` in receipt	Site insider; see T16
T13	Receipt forgery or edit	Receipt changed after the fact	Site signature plus log inclusion against a witnessed, anchored TIP Seal; receipt keys in hardware with revocation (Section 12.7)	A stolen receipt key allows forged new receipts until it is revoked
T14	Log fork	Site shows different histories to different auditors	Witness quorum and consistency proofs; ledger anchoring	Needs collusion of the site and a witness quorum
T15	Receipt withholding	Site acts but logs nothing	Synchronous receipts; agent-side logs; anchored log sizes	Detectable, not preventable
T16	Site lies about authority	Site logs an action the chain did not allow	Verifier re-derives authority from the signed chain, never trusts the site	None for detection
T17	Clock manipulation	Site backdates or postdates a decision	Bounds from request `created` and ledger anchor time	Uncertainty equal to the anchor interval
T18	Revocation race	Agent acts in the gap after revocation	Short `max_status_age`; status pushed by nodes; step-up actions check status live	Window equal to `max_status_age` for non-step-up actions
T19	Status denial	Attacker blocks status checks hoping sites fail open	Fail closed for ordinary changes; threshold status from several independent signers; safe-state actions stay available (Section 13.5)	Ordinary actions stop during a long outage; safe-state actions continue with degraded receipts
T20	Algorithm downgrade	Labels stripped or swapped to a weak scheme	Labels inside the signed body; verifier allow-list	None

theailab.org

#	THREAT	ATTACK	DEFENSE	RESIDUAL
T21	Prompt injection within scope	Hijacked agent does something allowed but unwanted	Narrow mandates; step-up for irreversible actions; instruction provenance; receipts for reversal and dispute	Real; limited to the mandate's scope
T22	Tool rug pull	Tool description changed after approval	Pinned manifest hash; signer refuses changed tools	None once pinning is in place
T23	Lying manifest or agent card	Agent describes itself falsely	Verified publisher; adjudication; public findings	Lie possible until disputed
T24	Sybil agents	Bad controller spins up new agents to escape history	No agent scores; conduct traces to a deduplicated TIP-ID; per-controller registration limits	Bounded by TIP-ID deduplication strength
T25	Mandate phishing	Fake service asks a person to sign a broad mandate	TIP client renders the mandate and shows the requester's verified TIP-ID and domain; ceilings on duration and verbs	Social engineering of a careless signer
T26	Mandate harvesting	Malicious site collects mandates it was shown	Mandates are useless without the agent key; audience binding; per-site mandates and selective disclosure	The harvested mandates still reveal business details the site was shown
T27	Controller compromise	Attacker controls an organization's signing member	Roster quorum for high-risk mandates; security-contact revocation; epoch advance	Window until detected
T28	Model holds the key	Injected prompt makes the model print its key	Key held by the agent signer, never in model context	None if custody rules are followed
T29	Verification flood	Junk requests to exhaust site CPU	Cheap checks before signature checks; rate limits per agent and per controller	Ordinary denial-of-service risk
T30	Canonicalization tricks	Two encodings of one mandate, or hidden fields	TIP canonical JSON; reject duplicate keys and unknown fields	None
T31	Counter evasion	Agent spreads spending across many sites	Per-site counters by root; counter service for global limits	Global limits need single-site mandates or a counter service

thetalab.org

#	THREAT	ATTACK	DEFENSE	RESIDUAL
T32	Human-label laundering	AI output pushed through an agent to look human	Origin floor of AA in `delegated` mode	A principal can still sign AI output in `self` mode; detection rests on the classifier and disputes
T33	Privacy harvesting	Observer profiles a company from public records	Only hashes and roots on the ledger; receipts held by parties; selective disclosure	Agent registration itself is public
T34	Quantum attack on old records	Future quantum computer forges old signatures	ML-DSA-65 and SHAKE-256; algorithm labels allow stronger schemes to be added later	Depends on the post-quantum assumptions NIST standardized
T35	Verification Provider compromise	A VP issues fraudulent TIP-IDs	TIP's existing REVOKE_VP cascade and accreditation controls	As in TIP today
T36	Backdated mandate	Holder of a retired or stolen key signs a mandate dated before the rotation	New actions need the key in effect at decision time; future `issued_at` refused	None for new actions
T37	Mislabeled action	Agent labels a delete as an update to fit its mandate	Site derives the action from its own endpoint definitions and refuses mismatches	None where endpoints are declared correctly
T38	Agent host compromise	Attacker takes over the machine running the agent signer	Signer enforces the mandate; hardware custody stops key export; step-up blocks high-risk actions; revocation	Attacker can act within scope until revoked
T39	Step-up swap	Approval for one action replayed on another, such as an empty-bodied delete	Site-signed challenge binding method, URI, audience, action, resource, digest, display and mandate; consumed atomically	None
T40	Approval fatigue	Flood of prompts until the principal taps approve	Rate limits; denials suspend the mandate; deny-and-stop in every prompt	A tired person can still approve
T41	Vendor key swap	Breached controller rotates every agent to its own keys	New key must sign rotations; notice delay; pinned keys on step-up mandates	Window of the notice delay, with the principal told

theailab.org

#	THREAT	ATTACK	DEFENSE	RESIDUAL
T42	TIP Stop All dodge	Thief signs mandates with a future date to survive an epoch change	Integer `issuer_epoch` raised only on the ledger	A thief who still holds the issuer key can sign new mandates at the new epoch; key revocation is required (Section 13.6)
T43	Junk-request amplification	Invented mandates naming throwaway keys force signed refusals	No signed receipt until the full chain verifies to an active TIP-ID; signed refusals rate-limited per root issuer	Ordinary denial-of-service risk
T44	Chain-URL abuse	Fetch URL points at the site's own network	Fetch only after the chain signature check; resolve the host once, check the address is public, connect to that exact address, follow no redirects, cap at 64 KB and 2 seconds	None where resolution is pinned
T45	Path tricks	`..`, encoded slashes or case variants slip past prefix checks	RFC 3986 normalization; dot segments and encoded slashes refused	None where the router and matcher share normalization
T46	Status-field widening	Sub-mandate sets a long status age or a dead status list	Status age narrows down the chain; each link checked against its own entry	None
T47	Cross-site tracking	Sites compare a personal principal's TIP-ID	Organization principals for consumer services; agent-only profile for low-risk reads (Section 20)	Real for personal principals using full chains
T48	Stolen TIP-ID key rotated	Thief with a principal's key signs a key rotation to take over	Hardware-locked keys that never leave the device; see open question 8	A core TIP question, beyond this paper
T49	Mandate beyond the issuer's own rights	Person with no rights at a site signs a mandate for an agent that has access there	Mandate only narrows the root issuer's own rights; site authorizes against the linked account (Section 8.7)	None where account linking is correct

tip.ai-lab.org

#	THREAT	ATTACK	DEFENSE	RESIDUAL
T50	Agent approves for its human	Browser or computer-use agent opens the approval prompt and clicks Approve	Approval on a device the agent cannot drive; no password approvals for high-risk verbs (Section 9.2)	A person who approves without reading
T51	Hidden amount or dose	Approval screen shows a path, not the amount, payee or dose	Site-signed typed display entries required per action class (Section 9.1)	None for registered display types
T52	Double use of one approval	Two copies of one approval sent at once	Challenge consumed in the same atomic step as the nonce and counters	None
T53	Rogue status source	Attacker runs a node that vouches for a revoked mandate	Threshold of Council-registered status signers; versioned, anchored status lists	Collusion of a threshold of status signers
T54	Legacy credential bypass	Agent uses an ordinary API key and never touches TIP	`tip_required` endpoints; egress through the signer (Section 7.4)	Sites that still accept legacy tokens for agents
T55	Structuring	A large payment split into many small ones under the step-up amount	`step_up_total` over a window	Splitting across sites without a counter service
T56	Reconnaissance from the registry	Attacker reads agent manifests to find vulnerable versions	No versions in public manifests; private visibility (Section 7.5)	Agent existence is visible for public agents
T57	Weaponized stop	Compromised security contact stops every agent during an incident	Single-contact stops only suspend; scoped stops; quorum restore (Section 13.6)	Up to 72 hours of suspended agents if the quorum does not act
T58	Status or coordinator outage	Central ledger component or status service unavailable	Several independent status signers; safe-state actions; degraded receipts (Section 13.5)	Ordinary actions wait for recovery
T59	Signature reuse across object types	A signature made for one object presented as another	`typ` in every object and FIPS 204 context strings (Section 6.4)	None

thevlab.org

#	THREAT	ATTACK	DEFENSE	RESIDUAL
T60	Lost or altered original	A party loses or edits the record behind a digest, so an action cannot be proven or is misrepresented	An altered original fails to match its signed digest; parties keep originals at least as long as the receipt (Section 12.8)	A lost original cannot be recovered, and the digest then proves only that something was signed
T61	False basis	An operator commits to a basis the agent did not use, to shift blame or hide an input	`basis_digest` is covered by the agent's signature and fixed at the time of the action; disputes (Section 16.2)	TIP cannot prove what a model read or relied on, and a compromised agent host (T38) can feed the signer a false basis that is attributed to the operator
T62	Hosted provider breach or misuse	A provider running a party's log, gateway or evidence store reads, leaks, alters or deletes what it holds, or uses the receipt key it holds to forge receipts and challenges	Hosting is optional; contract, residency and access-logging terms; encryption at rest; receipt keys in declared hardware custody with revocation (Section 12.7); altered originals fail to match their digests; the provider may not witness what it hosts (Sections 12.3 and 20)	Disclosure or loss of what the provider held, and forged receipts until the key is revoked

theailab.org

18. Interoperability

TIP is the anchor. These profiles define how each existing protocol carries TIP's objects without changing that protocol.

PROTOCOL	HOW TIP FITS
HTTP Message Signatures (RFC 9421) and Web Bot Auth	The request signature in Section 10 uses the RFC 9421 framework. `Signature-Agent` resolves to the TIP agent record, so a verifier learns who operates the agent and under what mandate. Where a Web Bot Auth verifier still accepts only Ed25519, agents add a transitional Ed25519 signature alongside ML-DSA-65 (Section 10.5). The Ed25519 part never satisfies any TIP check; a request counts as TIP-authenticated only if the ML-DSA-65 part verifies, so stripping it gains an attacker nothing
OAuth 2.1 and Rich Authorization Requests (RFC 9396)	A site that runs OAuth can accept a TIP mandate at its authorization server, map it to `authorization_details`, and issue a short-lived, sender-constrained token (DPoP) bound to the agent key. The token must expire no later than the mandate and no later than the smallest `max_status_age` in the chain, the authorization server must follow the status change feed and revoke tokens (RFC 7009) when a link is revoked, and resource servers record the token's `mandate_ref` in receipts. The mandate stays the source of authority
Model Context Protocol	The MCP server is the receiving site. It verifies the request signature and mandate, signs receipts, and registers its tool manifests as TIP records (Section 15.2)
Agent2Agent	The Agent Card carries the `tip://agent/` identifier and is signed by the controller's TIP-ID. Delegations between agents are sub-mandates. Because A2A payloads carry no user identity by design, the mandate chain supplies it where the receiving agent needs it
AP2, Verifiable Intent, card network agent programs, x402	Payments stay on those rails. A TIP mandate authorizes the agent to use them within limits, TIP Approval covers human-present purchases, and the receipt records the payment reference. Where a payment mandate is already signed by the principal, the TIP receipt references it instead of duplicating it
SCITT (RFC 9943)	Receipts and TIP Seals follow SCITT's statement and receipt model, so transparency services and auditors can process them with standard tooling

theainl.org

PROTOCOL	HOW TIP FITS
W3C Verifiable Credentials 2.0	A mandate can be expressed as a credential with the principal as issuer and the agent as subject, for ecosystems that already handle credentials
SPIFFE and WIMSE	Inside one organization, workload identity can protect the agent signer's connection to its key service; TIP covers the cross-organization step
C2PA	Content an agent produces can carry both a C2PA manifest and a CTID, with the CTID pointing to the action receipt
HL7 FHIR Provenance and AuditEvent	Each receipt maps to a Provenance resource, with the agent in `agent.who`, the delegating clinician or organization in `agent.onBehalfOf`, the mandate in `authorization` and the signature in `signature`, and to an AuditEvent whose `agent.altId` carries the agent identifier, so hospital audit systems ingest TIP evidence without new tooling. SMART on FHIR backend services stay the EHR's authorization method; the agent's SMART client is bound to its TIP identity
Model Context Protocol, July 28, 2026 revision	Step-up uses the multi round-trip `input_required` response for mid-call confirmation; the mandate reference and receipt hash travel in `_meta`; gateways set policy on the `Mcp-Method` header
A2A 1.0	Agent Cards are signed with ML-DSA-65 under the RFC 9964 identifiers and linked to the controller's TIP-ID; `TASK_STATE_AUTH_REQUIRED` maps to a TIP Approval
FIDO Alliance agentic authentication	The FIDO Agentic Authentication and Payments working groups, formed April 28, 2026 with AP2 and Verifiable Intent contributed, are the natural home for aligning TIP Approval with payment-grade passkey flows; TIP agent HTTP signatures stay wire-compatible with Visa's Trusted Agent Protocol and Web Bot Auth so merchants verify once

18.1 Legacy, clinical and industrial systems: the enclave and gateway profile

Much of the world's critical software cannot speak HTTP Message Signatures or verify a 3,309-byte signature. A substation's protection relays speak IEC 61850 or DNP3, a water plant's controllers speak Modbus in frames of about 250 bytes, a hospital's bedside devices speak HL7 version 2 over MLLP or DICOM, and a bank's core runs on mainframe transactions. Many of these networks are segmented under IEC 62443 zones with no internet path, keep clocks that drift, and must keep running when the wide-area network is down. A design that needs live ledger access on every request cannot be deployed there. This profile makes it deployable.

1. **Enforcement gateway.** A TIP Enforcement Gateway sits at the boundary of a zone, as the policy enforcement point. It verifies TIP requests on behalf of devices that cannot, applies every check in Section 11, translates approved requests into the native protocol, and signs receipts as the receiving site. The TIP-ID of the organization it serves, also when a provider hosts it (Section 20, item 10), the zones and device classes it protects, and its declared safe-state actions are published in its record. Devices behind it accept commands only from it.
2. **Enclave.** An enclave is a network segment with its own TIP Status Mirror and Enclave Log. Threshold-signed status versions cross into the enclave in one direction only, for example over a data diode or a scheduled transfer. The enclave declares `max_offline_duration`. Within it, mandates are honored against the newest status version the mirror holds; past it, only safe-state actions are allowed (Section 13.5).
3. **Local evidence.** Enclave Logs are sealed locally and cosigned by witnesses inside the facility, such as a separate security team's system, and they are anchored on the TIP ledger whenever a connection is available. Until open question 13 in Section 26 is settled, cosignatures from witnesses that do not meet the independence rule of Section 12.3 do not count toward the majority that section requires. The maximum merge delay for an enclave is set per deployment and stated in the gateway record.
4. **Local time.** Freshness and clock-skew bounds inside an enclave come from the enclave's declared trusted time source, such as a GPS-disciplined clock or a facility time server, named in the gateway record.
5. **Human approval on site.** In an enclave, approvals come from registered devices that work inside the facility, such as hardware security keys at an operator console, under the two-person policies in Section 9.4.
6. **Hard limits.** The gateway never sits inside a safety instrumented function or a real-time control loop, never delays a protective trip, and never blocks a local manual override. TIP governs which agent may request a change and leaves the evidence; the plant's own safety systems keep the last word.

The same pattern serves hospitals, where an integration engine already sits between clinical systems, and banks, where an API gateway already sits in front of the core.

theailab.org

19. Scalability and performance

19.1 Where the load goes

WORK	FREQUENCY	WHERE
Agent registration	Once per registered agent	TIP ledger
Mandate issuance	Per task or per renewal	Between principal and agent; nothing on the ledger
Action request verification	Per action	At the receiving site, almost always from cache
Receipt signing and logging	Per action	At the receiving site
TIP Seal cosigning	Per site per interval	Witnesses
TIP Seal anchoring	Per aggregator per interval	TIP ledger
Revocation	Per revocation event	TIP ledger, then pushed status lists

The ledger's load grows with agents, revocations, status blocks and aggregators. It does not grow with actions. The one exception is content that agents register in `delegated` mode, which grows with agent output and is priced and capped like any other registration.

19.2 Sizes

ITEM	SIZE
ML-DSA-65 signature	3,309 bytes (about 4.4 KB in base64)
ML-DSA-65 public key	1,952 bytes
Typical mandate with signature	About 5.5 KB in base64, plus about 2.6 KB for an ephemeral agent's public key on first contact
Worst-case first request, depth 3, hybrid signatures, stapled status and one approval	About 35 KB, carried in the session preflight, never in headers
Steady-state request overhead	One signature header, about 5 KB, plus a short mandate reference
Receipt with signature	About 4.6 KB or more
Status list for 131,072 mandates	16 KB before compression

Witness load stays small because each witness signs one root per interval (Section 12.3); its main work is checking consistency proofs, which are hashes. Status traffic is mostly stapled by agents (Section 13.2).

thetalab.org

The signature size is the main cost of being post-quantum. It is why chains travel by reference, why sites cache verified mandates, and why receipts are logged as hashes in the Merkle tree with full receipts held by the parties.

19.3 Ledger budget

SCENARIO	LEDGER TRANSACTIONS PER DAY	AVERAGE PER SECOND
100,000 sites anchoring every 5 minutes, unaggregated	28,800,000	About 333
1,000 aggregators anchoring every 5 minutes	288,000	About 3.3
10 million agents registered over one year	About 27,400	About 0.3
Status position blocks, 10 million mandates a day in blocks of 4,096	About 2,450	Under 0.1
Revocations and stops, assumed 1 percent of 10 million daily mandates	About 100,000	About 1.2
Delegated content registrations	Grows with agent-made content, a named volume `V_delegated`	Must be measured; this is the one ledger load that grows with agent output

The TIP specification records benchmarks above 5,000 transactions per second for the reference node and plans the move to full decentralization at a sustained 100 per second. The aggregated design stays far below that.

19.4 Measurements still needed

These numbers are calculated from published sizes. Before wide deployment, the reference implementation must measure ML-DSA-65 signing and verification rates on typical agent signer and site hardware, cache hit rates on real traffic, and witness cosigning latency. Pull request 383's demo prints a timing figure, and the next revision of this paper should replace estimates with measured results.

theailab.org

20. Privacy and data protection

1. **Nothing personal on the ledger.** The ledger holds agent records, revocations and Merkle roots. Receipts, mandates and request bodies stay with the parties or with providers they choose (item 10).
2. **Hashes, not content.** Receipts commit to digests. The content itself is never logged by TIP; the parties keep it, themselves or through a provider they choose (Section 12.8).
3. **Selective disclosure.** Mandates can show each site only what applies to it (Section 8.6).
4. **Cross-site tracking.** A personal principal's TIP-ID is visible to every site that verifies a full chain. Two measures limit that. Services acting for consumers should make the service's organization TIP-ID the principal and hold the consumer only as an opaque reference, as item 5 describes. For low-risk reads, the TIP Agent Profile should define an agent-only mode in which the agent presents its registered identity and no chain, so the site learns the agent and its controller but not the person. Per-site pseudonymous principal identifiers would go further and are an open research question.
5. **End users behind an agent.** When an agent acts for a company's customer, the mandate can name the company as principal and carry the customer only as an opaque reference the company can resolve, so customer identities are not spread across every site the agent touches.
6. **Erasure.** Because the ledger is append-only, the rule from TIP's existing design applies. Never anchor personal data, only roots. Receipts carry personal fields only as salted commitments, the hash of a random salt and the value, with the values and salts stored apart and released only to authorized verifiers. Erasing a person's data deletes the value and its salt, and the signed receipt stays valid. An endpoint whose request bodies carry short or guessable personal values defines a field for a random value of at least 32 bytes, which the agent fills for each request and each party holding the body deletes with it, so nobody holding a receipt can confirm the body by guessing it once the body is gone (Section 12.8, item 6).
7. **Volume privacy.** A log's size every interval reveals how busy a site is, which for a bank can be market-sensitive. Logs may pad each seal to a size bucket with dummy leaves, and witnesses must not publish per-log sizes.
8. **Unlinkable mandates.** Mandates for different sites should be issued separately, so one `mandate_id` does not link a principal's activity across sites, and status references name shared pools rather than issuers (Section 13.2).
9. **Controllers can keep names private.** A personal controller's display name follows TIP's existing opt-out. The TIP-ID is still public, because accountability needs it.
10. **Hosted services are optional.** TIP never requires a site, an agent operator or a principal to give request bodies, prompts, basis material or receipts to The AI Lab, to a node operator or to a

thetilab.org

witness, except evidence a party chooses to submit in a dispute (Section 16.2). Witnesses and aggregators receive TIP Seals, which hold a log identifier, a log size and a root hash, never content. A party may choose to have a third party, The AI Lab included, run a log, a gateway or an evidence store for it. That is a separate service under its own contract, with its own data processing, business associate and data residency terms, and the contract still carries the lookup cost lines of Section 22. A hosted log or gateway signs under the customer's TIP-ID, with a receipt key the customer registers and whose custody it declares (Section 12.7), never under the provider's own. The customer stays the receiving site and stays responsible for its evidence (Section 12.8). The provider gains no standing in the protocol and may not witness a log when it hosts that log or the evidence behind it (Section 12.3). It takes no part as an adjudicator in a dispute involving a party whose evidence it holds, and the AI Trust Council assigns a replacement for that role. Conformance never depends on using a hosted service, and evidence from an enclave stays inside it except for disclosure under Section 12.8, item 5.

theailab.org

21. Regulation and sector requirements

This design is meant to make it easier for organizations to meet the requirements regulators and standards bodies already impose. It does not, by itself, make anyone compliant with any law or standard, and no TIP record should be described as proof of compliance. The tables below map each requirement to the part of this design that supports it, and name the gaps TIP leaves to the organization. Statuses are as of November 9, 2026.

21.1 Health care

REQUIREMENT	WHAT IT ASKS	HOW TIP SUPPORTS IT	LEFT TO THE ORGANIZATION
HIPAA 45 CFR 164.312(a)(2)(i), unique user identification	A unique identifier for every user, including software	Every agent has its own key and identity; every receipt names the agent and the delegating person	Mapping identities into the EHR's access control
HIPAA 164.312(a)(2)(ii), emergency access	Procedures to reach ePHI in an emergency	Pre-issued break-glass mandates that work in degraded mode, with alerts and review (Section 9.4)	The emergency procedure itself
HIPAA 164.312(b), audit controls	Record and examine activity	Signed receipts, witnessed logs, FHIR AuditEvent mapping	Review of the records
HIPAA 164.312(c) and (d), integrity and authentication	Protect ePHI from improper change; verify who acts	Exact-bytes digests, `result_digest`, agent signatures and approvals	Encryption and storage of ePHI, which never enters TIP
HIPAA Security Rule proposed rule (90 FR 898, January 6, 2025)	Mandatory MFA, audit logs and 72-hour restoration	Designed to the anticipated MFA and logging standards	The rule is not final; the Unified Agenda lists a final action for July 2027
HL7 FHIR R5 Provenance, R4 AuditEvent	Who created or changed a resource, on whose behalf	Receipts map to both (Section 18)	Profile adoption by the EHR vendor
FDA Clinical Decision Support guidance (January 2026)	Clinicians can review the basis of a recommendation	Receipts carry the agent's `basis_digest`, so the basis can be audited later from the operator's records (Sections 10.6 and 12.8)	Showing the basis to the clinician at the time of use, and all clinical and device claims; TIP makes none

theailab.org

21.2 Identity assurance and federal systems

REQUIREMENT	WHAT IT ASKS	HOW TIP SUPPORTS IT
NIST SP 800-63B-4 (final August 2025), AAL3	Phishing-resistant authentication with a non-exportable key	Approvals and mandates use device-bound, non-exportable keys with origin binding (Section 8.2)
SP 800-63B-4 Section 3.2.3	Biometrics only with a physical authenticator, and a non-biometric option always offered	Security key with PIN has equal standing; voice never accepted
SP 800-63C-4, FAL3	Audience-restricted, replay-protected, holder-of-key assertions	Mandates name exact audiences, carry nonces and bind to the agent key
SP 800-53 Rev 5 IA-9, IA-2, AC-6	Unique identity for services; processes bound to the users they act for; least privilege	Agent identities, mandate chains, narrowing rules
SP 800-53 AU-9(3), AU-10	Cryptographic protection of audit records; non-repudiation	Signed receipts, witnessed and anchored TIP Seals
SP 800-207, zero trust	No implicit trust from network location; per-session authorization	Per-action verification at the receiving site
OMB M-22-09	Phishing-resistant MFA for agency staff	PIV and derived PIV accepted for approvals
OMB M-25-21 Section 4(b)	Human oversight, intervention and an appropriate fail-safe for high-impact AI	Step-up, approval policies, scoped stops and safe-state actions
Executive Order 14412 and OMB M-26-15 (June 2026)	Post-quantum signatures in high-value federal systems by December 31, 2031	ML-DSA from the first signature, with validated modules (Section 10.5)
CNSA 2.0	ML-DSA-87 for national security systems from January 1, 2027	CNSA profile with ML-DSA-87 throughout
FedRAMP 20x	Machine-readable security evidence for cloud services	A hosted TIP node or status service would target 20x Class C; signed receipts suit automated evidence

21.3 Critical infrastructure and operational technology

REQUIREMENT	WHAT IT ASKS	HOW TIP SUPPORTS IT
IEC 62443-3-3 and 62443-4-2	Identification, use control and timely response, without access controls blocking essential or safety functions	Enforcement gateway at the zone boundary; safe-state actions; never in a safety function (Sections 13.5 and 18.1)

tmaplab.org

REQUIREMENT	WHAT IT ASKS	HOW TIP SUPPORTS IT
CISA-led joint guidance on AI in OT (December 3, 2025)	Failsafes to manual control; operation without continuous cloud access; distinct AI identities in logs	Stops hand control back; enclaves with offline status mirrors; agent identities separate from people
NERC CIP-004-7 R5	Remove access within 24 hours of termination	Revocation reaches verifiers within `max_status_age`, never more than 24 hours, with cached lists in enclaves
NERC CIP-005-7 R2	Intermediate system and MFA for remote access, and a way to disable vendor system-to-system access	Agents reach BES Cyber Systems only through the gateway; scoped stops disable one vendor's agents
NERC CIP-007-6 R4	Log events, alert, keep logs at least 90 days, review every 15 days	Local receipt retention of at least 90 days; refusals raise alerts
NERC CIP-011-3	Protect BES Cyber System Information	Only hashes and roots leave the enclave
TSA Security Directive Pipeline-2021-02G	Segmentation so OT keeps running if IT is compromised; manual isolation	Verification and stops work inside the enclave without IT-side reachability
America's Water Infrastructure Act, SDWA 1433	Risk and resilience assessments covering cybersecurity	Per-agent identities and receipts replace shared logins, which EPA's May 2024 enforcement alert flagged

21.4 Finance and payments

REQUIREMENT	WHAT IT ASKS	HOW TIP SUPPORTS IT
DORA, Regulation (EU) 2022/2554, Article 9(4)	Access limited to legitimate functions; strong authentication; key protection	Mandates, approvals and hardware custody
DORA RTS, Delegated Regulation (EU) 2024/1774, Articles 12 and 21	Tamper-protected logs, detection of logging failure, clock synchronization, users identifiable at all times, emergency access	Witnessed logs where a missing seal is detectable; ledger-bounded time; break-glass mandates
PSD2 SCA RTS, Delegated Regulation (EU) 2018/389, Article 5	Dynamic linking of amount and payee to the authentication	Site-signed display entries for amount and payee in every payment approval (Section 9.1)
Payment Services Regulation, 2023/0210(COD)	Pending; plenary forecast for December 2026	Mandate format can carry its fields once final
SEC Rule 17a-4(f)(2)(i)(A)	Records with a complete time-stamped audit trail of changes and who made them	A witnessed append-only log matches the audit-trail option; firms still retain the records themselves

thetalab.org

REQUIREMENT	WHAT IT ASKS	HOW TIP SUPPORTS IT
FINRA 2026 Regulatory Oversight Report	Agents acting without validation or beyond authority	Mandate scopes and step-up checkpoints; receipts as the supervision trail
FFIEC authentication guidance (2021)	Layered security and MFA for third parties	Agents treated as third parties with step-up for high-risk actions

21.5 Artificial intelligence regulation

REQUIREMENT	WHAT IT ASKS	HOW TIP SUPPORTS IT
EU AI Act Article 14(4)(e)	A stop button or similar procedure that halts the system in a safe state	Scoped stops with compensation windows and safe-state actions (Section 13.6)
EU AI Act Article 14(5)	For remote biometric identification, confirmation by at least two natural persons	Approval policies with a threshold of two, each approver identified in the receipt
EU AI Act Articles 12, 19 and 26(6)	Automatic logs kept at least six months	Receipts and default six-month retention
EU AI Act Article 50	Transparency for AI-generated content, applying from August 2, 2026	CTIDs on agent-made content with the origin floor (Section 14)
Regulation (EU) 2026/1744, the AI Omnibus	Moves Annex III high-risk obligations to December 2, 2027 and Annex I to August 2, 2028	The mapping applies from those dates
ISO/IEC 42001 Annex A	Event logging for AI systems	Receipts as evidence
NIST AI RMF and the planned critical infrastructure profile	Accountability and deactivation	Accountable TIP-IDs, scoped stops
Singapore IMDA Model AI Governance Framework for Agentic AI (January 22, 2026)	Human checkpoints before sensitive or irreversible actions; agent identity and audit trails	Minimum step-up list, agent identities, receipts

21.6 European identity and cyber resilience

eIDAS 2.0 (Regulation (EU) 2024/1183) requires each Member State to offer a European Digital Identity Wallet for natural and legal persons. It does not provide wallets for AI agents. A mandate's issuer may be a person or an organization verified with an EUDI or European Business Wallet attestation, and the agent acts under that mandate; no claim is made that agents hold wallets. NIS2 Article 21(2)(h) and (j) ask for cryptography policies and multi-factor authentication, which agent step-up and the algorithm registry document. The Cyber Resilience Act requires vulnerability reporting with a 24-hour early warning from September 11, 2026, which applies to TIP software sold

theailab.org

in the EU. The UK Cyber Security and Resilience Bill was still before Parliament on November 9, 2026.

21.7 Contracts, liability and terminology

A mandate is evidence of what a principal authorized. Whether it creates legal authority, and who bears liability for an agent's action, depends on the governing law and the parties' agreements. TIP records the facts; it does not settle the law. United States federal documents issued under Executive Order 14434 (September 29, 2026) use the term "Super Intelligence (SI)" where earlier documents said "AI", and some NIST pages now refer to "SI agents". The legal definitions are unchanged, and this paper keeps the term AI agent.

theailab.org

22. Economics and lookup costs

Verification of TIP records is free to the public, and every lookup costs The AI Lab compute, database, bandwidth and node capacity. Agent traffic could multiply lookups by orders of magnitude, so any pilot, partner deal or price for agent accountability must carry these cost lines, with volume times unit cost shown openly. Where a unit cost has not been measured it stays a named variable.

COST LINE	VOLUME DRIVER	UNIT COST
TIP-ID resolution (click or verify)	Root issuers and controllers resolved, after caching	`C_tipid`, not yet measured
CTID resolution (click or verify)	Agent-made content and tool manifests checked	`C_ctid`, not yet measured
Near-similar content search	Agents and sites checking inputs against registered content	`C_similar`, not yet measured
Full content history lookup	Audits and disputes	`C_history`, not yet measured
Agent record resolution	Registered agents resolved, after caching	`C_agent`, not yet measured
Mandate status list fetch	Status list refreshes, driven by `max_status_age`	`C_status`, not yet measured
TIP Seal anchoring write	Aggregators times intervals	`C_anchor`, not yet measured
Inclusion proof lookup	Receipts verified by auditors and disputants	`C_proof`, not yet measured

Monthly cost for a counterparty is the sum over lines of (monthly volume times unit cost), where volume is requests multiplied by one minus the cache hit rate. Caching rules in Sections 10.2 and 13.2 are therefore also cost controls.

Rate limits, caps and fees never apply to threshold-signed status versions or to the revocation change feed. Throttling status would make sites fail closed and stop agents in hospitals and utilities, so status is served from public mirrors at the Lab's cost, shown as `C_status` times volume in every model, or from the counterparty's own node. Lookup fees, caps and limits apply only to the other lines.

High-volume counterparties, such as an agent platform serving millions of actions a day, should either run their own TIP node, which removes most of the lookup cost from The AI Lab, or accept a lookup fee, a volume cap or a rate limit. Every proposal to such a counterparty must say which.

theailab.org

23. Changes needed in TIP

The items follow two paths. Object specifications, registries, parameters, companion profiles, conformance classes, the optional `basis_digest` field and the evidence-keeping rule live off the ledger. They form the TIP Agent Profile, which The AI Lab publishes, revises after a public comment window and deploys now, without the RFC process (TIP Protocol Specification v5.0, Sections 49.2 and 50.1). The new identifier, the new transaction types, node features, changes to existing transaction types and to content registration, adjudication changes, the CNSA profile, the Merkle tree change that witnesses need and the specification correction change the TIP ledger or its core specification. Each of those goes through the RFC process with a thirty-day comment period and ninety-day testnet, and each new transaction type gets a schema module, commit-time verification and an activation epoch, as TIP already requires for commit-rule changes. The IETF items follow the IETF's own process.

CHANGE	KIND	NEEDED FOR
`tip://agent/` resource type	New identifier, IANA registration update	Registered agents
AGENT_REGISTERED, AGENT_UPDATED, AGENT_KEY_ROTATED, AGENT_REVOKED	New transaction types	Agent identity
Mandate schema, constraint vocabulary, narrowing rules, verb vocabulary	New object specification	Delegated authority
MANDATE_REVOKED, MANDATE_EPOCH_ADVANCED	New transaction types	Revocation and TIP Stop All
Status list allocation (`status_index`), threshold-signed stapled status and the global change feed	Node feature	Revocation at scale
TIP Approval object and minimum `step_up` list	New object specification	Human control
Receipt and TIP Seal schemas, SCITT alignment	New object specification	Evidence
RECEIPT_KEY_REGISTERED, WITNESS_REGISTERED, LOG_CHECKPOINT_ANCHORED	New transaction types	Logs and anchoring

thealab.org

CHANGE	KIND	NEEDED FOR
`delegated` attribution mode, the `register_content` verb, the origin floor and the no-priority rule	Change to content registration	Agent-made content
Agent conduct dispute reasons	Adjudication change	Accountability
HTTP, MCP, A2A and OAuth profiles	Companion documents	Interoperability
SHAKE-256 in the HTTP digest registry; the `tip-basis-digest` field in the HTTP Field Name registry; TIP profiles brought to the IETF audit BoF, agentproto, WIMSE and OAuth groups (Section 10.5)	IETF work	Standards fit
CNSA profile with ML-DSA-87 throughout	Specification section	National security systems
ROSTER_UPDATED	New transaction type	Organization signers, approvers and security contacts
STATUS_SIGNER_REGISTERED and threshold-signed, versioned status lists	New transaction type and node feature	Status no single party controls
MANDATE_EPOCH_RESTORED, scoped epochs, AGENT_SUSPENDED	New transaction types	Safe stops
RECEIPT_KEY_ROTATED, RECEIPT_KEY_REVOKED, WITNESS_SET_UPDATED, WITNESS_KEY_REVOKED	New transaction types	Key lifecycle
`continuity` field on KEY_ROTATED	Change to an existing transaction type	Routine rotation without mass failure
RFC 9162 tree and consistency proofs in `shared/merkle.js`	Reference implementation change	Witnesses and monitors
The registries in Section 6.5 and the parameters in Section 6.6	Specification section	Testability
Section 15.1 brought into line with the fresh-biometric signing rule	Specification correction	Human control
Conformance classes: Agent Signer, Receiving Site, Enforcement Gateway, Witness, Monitor, Status Signer, Verifier	Specification section	Testing and certification
`basis_digest` in the Action Request and the Receipt	New optional field in two object specifications	The basis of an action

thetailab.org

CHANGE	KIND	NEEDED FOR
The `tip-basis/1` object type, the erasure record format, and the reason codes `basis_required` and `basis_not_allowed`	Registry entries and specification section	The basis of an action and erasure
The evidence-keeping rule of Section 12.8	Specification section	Audits and disputes

Apart from the new `continuity` field on KEY_ROTATED and the correction to Section 15.1, none of these changes alters existing transaction types, the canonical signed payload for content, or the meaning of existing CTIDs and TIP-IDs.

23.1 What The AI Lab deploys now

The TIP Agent Profile needs no protocol change, and The AI Lab deploys it now. Pull request 383 demonstrates the full flow with TIP's existing primitives, offline and with an in-memory identity directory. Once the reference code reads real TIP-IDs from the ledger and has been reviewed, a site can verify mandates signed by its customers' TIP-IDs, sign receipts and log them. Until RECEIPT_KEY_REGISTERED exists, the site needs an interim way to publish its receipt key under its TIP-ID, such as a content record signed by that TIP-ID. Mandates need TIP clients that render and sign them under the profile (Section 8.2). TIP Approval also needs client support, so deployments before then avoid verbs on the minimum step-up list (Section 9). The public agent namespace, ledger revocation of mandates, witnesses and ledger anchoring wait for the RFC.

theailab.org

24. Roadmap

PHASE	WHAT SHIPS	DONE WHEN
0. Deploy	Version 1 of the TIP Agent Profile published; pull request 383 hardened and merged as the reference, reading real TIP-IDs; mandate and approval signing in TIP clients; site-signed receipts under existing TIP-IDs; the first deployments	A deployment logs real agent actions and a third party verifies them from public keys and ledger state alone
1. Ledger specification	RFCs for public agent records, ledger revocation and epochs, status signers, witnesses and anchoring, written in BCP 14 normative language (RFC 2119 and RFC 8174), with separate Security Considerations and Privacy Considerations in the style of RFC 3552 and RFC 6973; the profile's media types such as `application/tip-mandate+json` and a well-known URI published alongside	The ledger RFCs pass comment and the Council adopts them
1a. Proof	Test vectors for every object and every narrowing rule; a symbolic model of delegation, step-up and receipts in Tamarin or ProVerif; a TLA+ model of nonces, counters, challenges and revocation	Models published with no open attack, and reviewed by outside cryptographers
2. Testnet	New transaction types on the test cluster; agent signer, receiving-site and gateway SDKs using FIPS 140-3 validated ML-DSA and SHAKE implementations for regulated deployments; witness, monitor and status-signer networks run by independent operators	Ninety days of testnet operation with measured performance, an external security review, and conformance suites for every class in Section 23
2a. Interop and safety	Two independent implementations; an interoperability event; a sector pilot each in health care and in an IEC 62443 industrial enclave, with safe-state and degraded-mode drills	Both implementations pass the conformance suites and the drills show safe-state actions continue during a full outage
3. Interoperability	HTTP and Web Bot Auth, MCP, A2A and OAuth profiles; one real workflow per profile	Each profile verified end to end with an outside framework

thetalab.org

PHASE	WHAT SHIPS	DONE WHEN
4. Mainnet	Activation epoch for the new rules; agent conduct disputes in adjudication	First registered agents and anchored logs on mainnet

theailab.org

25. What this design does not do

1. It does not make an agent safe or its decisions good. It records them and limits their reach.
2. It does not stop prompt injection. A hijacked agent can still do anything its mandate allows, which is why mandates should be narrow and irreversible actions need step-up.
3. It does not prevent a site from withholding receipts. It makes withholding detectable.
4. It does not enforce global spending limits across many sites without a counter service or single-site mandates.
5. It does not prove that content an agent made is true, stop copying, or stop AI training.
6. It does not settle legal liability. It supplies evidence for whoever does.
7. It does not replace a site's own security. It gives that security better information.
8. It does not belong inside real-time control loops or safety instrumented functions, and it must never be placed there.
9. It does not make a site compliant with HIPAA, NERC CIP, IEC 62443, DORA, the EU AI Act or any other law or standard. The mappings in Section 21 show where TIP records can serve as evidence and where they cannot.
10. It does not prove what a model read or relied on. `basis_digest` records the operator's signed statement of it (Section 10.6).
11. It does not keep anyone's records. Each party keeps its own originals, itself or through a provider it chooses (Section 12.8), and nobody has to use a hosted service (Section 20, item 10).

theailab.org

26. Open questions for The AI Lab and the AI Trust Council

1. **Standing authority and the signing rule.** A mandate is one signature, signed after a fresh biometric, that lets software act later. That matches the letter of TIP's signing rule and stretches its spirit. Should mandates be approved as a scoped exception, as website batch signing was, with the ceilings in Section 8.2 written into the rule?
2. **Who may register agents.** Should personal TIP-IDs register public agents, or only organization TIP-IDs at first?
3. **Agent namespace.** Is `tip://agent/` the right resource type, and should it carry a region segment like TIP-IDs do?
4. **Mandates on the ledger.** Should high-value mandates have their hashes anchored at issuance, which helps disputes but adds ledger writes and reveals timing?
5. **Score effects.** What, if any, SCORE_UPDATE effects should upheld agent conduct findings carry for controllers and principals?
6. **Witness set.** Should witnessing start with the Founding Node Operators, and what quorum is required?
7. **Lookup economics.** What lookup fee, cap or rate limit applies to agent platforms that do not run a node?
8. **Key rotation under theft.** TIP Specification Section 15.2 lets the current key sign a rotation. A thief who extracts a principal's key could rotate it and take over. Hardware-locked keys make extraction hard, but should KEY_ROTATED also require a fresh device-bound WebAuthn assertion and a notice period?
9. **First pilots.** Which two use cases in Section 3 should the first pilots cover, and which partners run them?
10. **Patent and licensing terms for a neutral standard.** Bodies such as the IETF and W3C, and most government buyers, need royalty-free or fair and reasonable patent terms disclosed before they adopt a standard. Should The AI Lab offer a royalty-free non-assert covering implementations of the TIP agent wire profiles, and submit those profiles to the IETF, while ledger rules stay under TIP governance?
11. **Status signers and witnesses.** Which independent operators, in which jurisdictions, form the first status signer and witness sets, and what threshold applies?
12. **Emergency panel.** Who sits on the panel that may suspend a harmful agent under Section 13.7, and how is it held to account?
13. **Enclave witnesses.** Section 18.1 lets an enclave's logs be cosigned by witnesses inside the facility, which may not meet the independence rule of Section 12.3. Should enclave seals carry a

theailab.org

weaker, labeled guarantee until independent witnesses cosign them, and how should a verifier report it?

14. **Long-offline enclaves.** An enclave anchors its log whenever a connection is available, while Section 6.6 caps the maximum merge delay at 24 hours. What longer range should apply to enclaves that stay offline for more than a day?
15. **Risk class of purchases.** Section 9's minimum step-up list covers every financial verb. Is `purchase` financial in every case, or should small purchases under `step_up_amount` stay outside the list, as Section 3.1 assumes?

theailab.org

Appendix A. Example records

Field values are shortened for reading. Real signatures are 3,309 bytes. Amounts are integer minor units.

Root mandate

theailab.org

```
{
  "typ": "tip-mandate/1",
  "v": 1,
  "mandate_id": "shake256:9c41...e0",
  "issuer": "tip://id/US-a3f8c91b2d4e7021",
  "subject": "tip://agent/5d0e77a1c4b2f913",
  "subject_key": "keythumb:shake256:44d1...8e",
  "audience": ["https://cms.example.com"],
  "account": "cms.example.com:user:88412",
  "actions": ["read", "create", "update", "delete"],
  "resources": ["/sites/blog/"],
  "constraints": {
    "environment": ["staging"],
    "max_uses": 500,
    "rate": {"count": 60, "window_seconds": 60},
    "data_classes": ["internal", "public"]
  },
  "step_up": ["delete", "publish"],
  "approval_policy": {
    "threshold": 1,
    "approvers": ["tip://id/US-a3f8c91b2d4e7021"],
    "authenticator_class": "separate_device",
    "timeout": 300
  },
  "on_status_unavailable": "refuse",
  "epoch_scope": ["all", "environment:staging"],
  "purpose": "Draft and revise blog posts in staging for the November launch",
  "issued_at": "2026-11-09T13:58:12Z",
  "not_before": "2026-11-09T14:00:00Z",
  "expires": "2026-11-09T22:00:00Z",
  "max_depth": 1,
  "status_ref": "tip://status/pool/0412",
  "status_index": 40211,
  "max_status_age": 120,
  "issuer_epoch": 3,
  "nonce": "f1c2...9a",
  "sig_alg": "ML-DSA-65",
  "hash_alg": "SHAKE-256",
```

```
  "signature": "3f9a..."
}
```

Sub-mandate to a helper agent

theailab.org

```
{
  "typ": "tip-mandate/1",
  "v": 1,
  "mandate_id": "shake256:51aa...7c",
  "issuer": "tip://agent/5d0e77a1c4b2f913",
  "parent": "shake256:9c41...e0",
  "root": "shake256:9c41...e0",
  "subject": "keythumb:shake256:be03...11",
  "audience": ["https://cms.example.com"],
  "account": "cms.example.com:user:88412",
  "actions": ["read", "update"],
  "resources": ["/sites/blog/drafts/"],
  "constraints": {
    "environment": ["staging"],
    "max_uses": 50,
    "rate": {"count": 10, "window_seconds": 60},
    "data_classes": ["public"]
  },
  "step_up": ["delete", "publish"],
  "approval_policy": {
    "threshold": 1,
    "approvers": ["tip://id/US-a3f8c91b2d4e7021"],
    "authenticator_class": "separate_device",
    "timeout": 300
  },
  "on_status_unavailable": "refuse",
  "purpose": "Fix spelling in draft posts",
  "issued_at": "2026-11-09T14:59:40Z",
  "not_before": "2026-11-09T15:00:00Z",
  "expires": "2026-11-09T16:00:00Z",
  "max_depth": 0,
  "max_status_age": 60,
  "issuer_epoch": 0,
  "nonce": "07de...42",
  "sig_alg": "ML-DSA-65",
  "hash_alg": "SHAKE-256",
  "signature": "a812..."
}
```

Step-up challenge for a payment

```
{
  "typ": "tip-challenge/1",
  "v": 1,
  "site": "tip://id/US-77b0c2d9e1a4f530",
  "challenge_id": "ch_7f3e...b1",
  "method": "POST",
  "target_uri": "https://pay.example.com/v2/transfers",
  "audience": "https://pay.example.com",
  "action": "transfer",
  "resource": "/v2/transfers",
  "content_digest": "shake256:aa90...3c",
  "mandate_ref": "shake256:e2c8...19",
  "meaning": "authorize_payment",
  "display": [
    {"type": "amount", "value": "980000", "currency": "USD"},
    {"type": "payee", "name": "Acme Supplies Ltd", "account_ref": "...4471"},
    {"type": "environment", "value": "production"}
  ],
  "expires": "2026-11-09T15:14:04Z",
  "receipt_key_id": "rk_2026_11",
  "sig_alg": "ML-DSA-65",
  "hash_alg": "SHAKE-256",
  "signature": "9b1c..."
}
```

Receipt

```
{
  "typ": "tip-receipt/1",
  "v": 1,
  "site": "tip://id/US-77b0c2d9e1a4f530",
  "log_id": "log:US-77b0c2d9e1a4f530:1",
  "receipt_key_id": "rk_2026_11",
  "phase": "final",
  "intent_ref": 1048576,
  "agent": "keythumb:shake256:be03...11",
  "action": "update",
  "resource": "/sites/blog/published/launch",
  "content_digest": "shake256:0c11...5e",
  "request_hash": "shake256:0b7e...d2",
  "mandate_ref": "shake256:51aa...7c",
  "root": "shake256:9c41...e0",
  "decision": "refused",
  "reason": "outside_scope:resource",
  "status_mode": "normal",
  "decided_at": "2026-11-09T15:12:04Z",
  "result_digest": null,
  "basis_digest": "shake256:5d2f...a8",
  "status_seen": {"ref": "tip://status/pool/0412", "version": 918, "threshold_sig": "77ab..."},
  "ledger_seen": "tx:7c1e...a0",
  "log_index": 1048577,
  "sig_alg": "ML-DSA-65",
  "hash_alg": "SHAKE-256",
  "signature": "c40d..."
}
```

Appendix B. Verifier outline

theailab.org

The verifier returns `valid`, `invalid` or `indeterminate`. It returns `indeterminate`, never `valid`, when

thealab.org

any ledger state it needs cannot be obtained.

```
verify_action(request, chain, receipt, proofs, ledger):
    t = receipt.decided_at
    assert labels_allowed(request, chain, receipt)            # algorithm registry
    assert typ_and_context_match(request, chain, receipt)      # Section 6.4
    assert sig_valid(request, key_in_effect(request.signer, at=t))
    leaf = chain[-1]
    assert recompute_id(leaf) == request.mandate_ref
    assert request.signer matches leaf.subject (and leaf.subject_key if pinned)
    if registered(leaf.subject):
        assert agent_status(leaf.subject, at=t) == active
        assert tipid_status(controller(leaf.subject), at=t) == active
    min_age = infinity
    for each mandate m in chain, root first:
        assert canonical(m) and no_unknown_fields(m)
        signer = m.issuer if m is root else parent_of(m).subject
        assert sig_valid(m, key_in_effect(signer, at=t))       # forward rule and continuity
        if m.signers: assert roster_satisfied(m.signers, m.issuer, roster_at(m.issued_at),
roster_at(t))
        if m is not root: assert narrower_or_equal(m, parent_of(m))   # Section 8.4
        if grants_minimum_list_verb(m) and m is not root: assert delegation_approval_valid(m)
        assert m.issued_at <= t + skew
        assert m.not_before <= t <= m.expires
        assert m.not_before <= request.created <= m.expires
        assert m.issuer_epoch >= epoch_of(m.issuer, scope=m.epoch_scope,
as_of=receipt.status_seen)
        assert not revoked(m, as_of=receipt.status_seen)
        assert no_anchored_status_showed_revoked(m, between=(receipt.status_seen, t -
m.max_status_age))
        assert in_scope(receipt.action, receipt.resource, m)  # every link, using the site's
derivation
        min_age = min(min_age, m.max_status_age)
    assert threshold_signed(receipt.status_seen, status_signers(at=t))
    if receipt.status_mode == normal: assert age(receipt.status_seen, at=t) <= min_age
    else: assert safe_state_action(receipt.site, receipt.action, receipt.resource)
    if step_up_applies(receipt.action, chain):
        for each approval a in request.approvals:
            ch = a.challenge
            assert sig_valid(ch, receipt_key(receipt.site, at=t)) and ch.expires >= t
```

```
            assert fields_match(ch, request) and hash(ch.display) == receipt.display_hash
            assert sig_valid(a, key_in_effect(a.approver, at=t)) and a.user_verified
        assert approval_policy_satisfied(request.approvals, leaf.approval_policy, roster_at(t))
    assert sig_valid(receipt, receipt_key(receipt.site, at=t))
    assert not receipt_key_compromised_before(receipt, proofs.anchor_tx)
    assert receipt.request_hash == shake256(signature_base(request) || request.signature)
    assert receipt.content_digest == request.content_digest
    assert receipt.basis_digest == request.basis_digest     # both absent or equal
    if request.basis_digest: assert covered(request, "tip-basis-digest")   # Section 10.1
    assert inclusion(receipt, receipt.log_index, proofs.site_seal)      # RFC 9162 proof
    assert majority_witnessed(proofs.site_seal, witness_set(receipt.log_id, at=t))
    assert anchored_on_ledger(proofs.witness_roots, proofs.anchor_tx)
    assert ledger_time(receipt.ledger_seen) <= t <= ledger_time(proofs.anchor_tx)
    assert t - ledger_time(receipt.ledger_seen) <= ledger_seen_max_age
    assert ledger_time(proofs.anchor_tx) - t <= max_merge_delay(receipt.log_id)
    return (which_agent = request.signer,
            whose_authority = root_issuer(chain),
            what_it_did = (receipt.action, receipt.resource, receipt.decision))
```

The verifier never takes the site's word for authority. It re-derives authority from the signed chain, judges revocation against what the site was entitled to know, and treats the receipt only as the site's signed statement of what it decided. It cannot check, from public data, that the site authorized the root issuer's linked account correctly under its own access control (Section 8.7); that check belongs to the site and to any dispute.

An auditor or the confidential panel checks a revealed basis (Section 10.6) this way.

```
verify_basis(receipt, manifest, manifest_salt, items, item_salts,
             erasure_records, dispute_filed):
    assert receipt.basis_digest == shake256(manifest_salt || canonical(manifest))
    assert manifest.typ == "tip-basis/1"
    for each entry e in manifest.entries:
        if e in erasure_records:                        # Section 12.8, item 6
            assert fixed_in_time_before(erasure_records[e], dispute_filed)
        elif e not in items:
            mark_missing(e)      # counts only if missing before retention ended
        else: assert e.commitment == shake256(item_salts[e] || items[e])
    return manifest    # the operator's signed statement, never proof that the model used it
```

Appendix C. How this paper builds on pull request 383

PULL REQUEST 383	THIS PAPER
Agent signs every request with its own ML-DSA-65 key	Kept; adds registered and ephemeral agents, the agent signer, custody classes and the manifest (Section 7)
Principal's TIP-ID signs a scoped, time-limited delegation; sub-agents get only narrower power	Kept; adds a fixed schema, a closed constraint vocabulary, eleven narrowing rules, the environment constraint and the plain-language signing rules (Section 8)
Site checks signature, freshness, chain, scope, revocation and bytes in order	Kept; adds audience, agent status, step-up, a denial-of-service-aware order and nonce recording after the signature check (Section 11)
Site signs a receipt for every decision, including refusals, into a Merkle log with signed checkpoints, called TIP Seals in this paper	Kept; adds synchronous return, `result_digest`, witnesses, two-tier anchoring and agent-side logs (Section 12)
Algorithm labels inside the signed body	Kept for every object
Exact SHAKE-256 of the bytes, not the CNA-2 hash, for authorization	Kept and stated as a rule (Section 10.3)
An action taken before a revocation stays valid	Kept; defines "before" with anchored time bounds (Section 13.4)
Limits: in-memory identity lookup, site clock, stolen keys, withheld receipts, illustrative namespace	Each answered in Sections 7, 12, 13 and 23
Next steps: anchor the log, one real workflow, RFC if useful	Phases 0 to 4 of the roadmap (Section 24)
Not covered	TIP Approval, revocation at scale, TIP Stop All, agent-made content and the origin floor, input provenance and tool pinning, accountability without agent scores, interoperability profiles, privacy, regulation and lookup costs

thetailab.org

References

1. The AI Lab, “Agentic Provenance on TIP”, internal research paper prepared by Rama Chandra M under the direction of Dinesh Mendhe, November 9, 2026, and theailaborg/tip-protocol pull request 383. https://github.com/theailaborg/tip-protocol/pull/383
2. The AI Lab, TIP Protocol Specification v5.0, June 1, 2026.
3. IETF, HTTP Message Signatures, RFC 9421.
4. IETF Web Bot Auth working group, HTTP Message Signatures for Bots, draft-ietf-webbotauth-httpsig-protocol. https://datatracker.ietf.org/doc/html/draft-ietf-webbotauth-httpsig-protocol
5. IETF, HTTP Digest Fields, RFC 9530; OAuth 2.0 Token Exchange, RFC 8693; OAuth 2.0 Rich Authorization Requests, RFC 9396; OAuth 2.0 Demonstrating Proof of Possession, RFC 9449.
6. IETF individual drafts on agent delegation (2026), including draft-asor-wimse-agent-delegation-chain, draft-niyikiza-oauth-attenuating-agent-tokens, draft-sweeney-wimse-credential-delegation and draft-klrc-aiagent-auth. https://datatracker.ietf.org/doc/html/draft-asor-wimse-agent-delegation-chain-03
7. IETF, An Architecture for Trustworthy and Transparent Digital Supply Chains (SCITT), RFC 9943, June 2026. https://www.rfc-editor.org/rfc/rfc9943
8. NIST NCCoE, Accelerating the Adoption of Software and AI Agent Identity and Authorization, concept paper, February 5, 2026. https://www.nccoe.nist.gov/sites/default/files/2026-02/accelerating-the-adoption-of-software-and-ai-agent-identity-and-authorization-concept-paper.pdf
9. OWASP GenAI Security Project, OWASP Top 10 for Agentic Applications for 2026, December 9, 2025. https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026
10. Ethereum Improvement Proposals, ERC-8004 Trustless Agents (draft). https://eips.ethereum.org/EIPS/eip-8004
11. Mastercard and Google, Verifiable Intent, March 5, 2026. https://www.pymnts.com/mastercard/2026/mastercard-unveils-open-standard-to-verify-ai-agent-transactions/
12. Google, Agent Payments Protocol (AP2), 2025.
13. Cloudflare, x402 and agent payments. https://blog.cloudflare.com/x402
14. Cloud Security Alliance, MCP Security, Tool Poisoning and the Confused Deputy Problem, May 2026. https://labs.cloudsecurityalliance.org/research-rb/csa-whitepaper-mcp-security-agentic-attack-surface-20260516/
15. Cloud Security Alliance, IETF’s Race to Standardize AI Agent Identity, September 2, 2026. https://labs.cloudsecurityalliance.org/research/csa-research-note-ietf-agent-identity-standards-20260902-csa/

theailab.org

16. Gray Swan, Indirect Prompt Injection Arena results, March 2026. https://www.grayswan.ai/blog/your-ai-agent-can-be-compromised-youd-never-know
17. CSO Online, "Prompt injection breaks today's AI agents, study warns", June 2026. https://www.csoonline.com/article/4184455/prompt-injection-breaks-todays-ai-agents-study-warns.html
18. TechRepublic, "AI agent reportedly deletes company's entire database", April 2026. https://techrepublic.com/article/ai-agent-deletes-company-database-admits-violating-guardrails
19. Goodwin, "Not Delayed, Not Deferred: EU AI Act Transparency Obligations Are Now in Force", August 2026. https://www.goodwinlaw.com/en/insights/publications/2026/08/alerts-technology-dpc-eu-ai-act-transparency-obligations-now-in-force
20. W3C, Verifiable Credentials Data Model 2.0, and Bitstring Status List.
21. The AI Lab, Verifiable Assets. https://theailab.org/verifiable-assets
22. The AI Lab, Global Seal of Trust. https://theailab.org/global-seal-of-trust
23. Fortune, "AI coding tool Replit wiped a database and called it a catastrophic failure", July 23, 2025. https://fortune.com/2025/07/23/ai-coding-tool-replit-wiped-database-called-it-a-catastrophic-failure/
24. The Decoder, report on the Financial Times account of the AWS Kiro outage, February 2026. https://the-decoder.com/aws-ai-coding-tool-decided-to-delete-and-recreate-a-customer-facing-system-causing-13-hour-outage-report-says/
25. Windows Central, Meta director's inbox deleted by an OpenClaw agent, February 2026. https://www.windowscentral.com/artificial-intelligence/meta-summer-yue-director-openclaw-ai-email-deletion
26. AI Incident Database, DataTalks.Club production deletion, incident 1424, February 26, 2026. https://incidentdatabase.ai/cite/1424/
27. OECD AI Incidents Monitor, Meta internal agent Sev 1 data exposure, March 18, 2026. https://oecd.ai/en/incidents/2026-03-18-fefc
28. ABC News, PocketOS database deletion, April 2026. https://abcnews.com/GMA/News/rogue-ai-agent-haywire-tech-company-ceo-bullish/story?id=132473181
29. SmartCompany, Who Gives A Crap suspends AI email agent, July 2026. https://www.smartcompany.com.au/retail/who-gives-a-crap-suspends-ai-agent-email-error-prices-would-double/
30. Euronews, Swedish reality show season lost amid disputed AI claims, October 14, 2026. https://www.euronews.com/2026/10/14/entire-swedish-reality-show-season-lost-weeks-before-premiere-amid-disputed-ai-claims
31. Invariant Labs, GitHub MCP exploited, May 2025. https://invariantlabs.ai/blog/mcp-github-vulnerability

theailab.org

32. The Hacker News, EchoLeak zero-click vulnerability in Microsoft 365 Copilot, CVE-2025-32711, June 2025. https://thehackernews.com/2025/06/zero-click-ai-vulnerability-exposes.html
33. SC World, Amazon Q extension injected with wiper prompt, July 2025. https://www.scworld.com/news/amazon-q-extension-for-vs-code-reportedly-injected-with-wiper-prompt
34. Snyk, weaponizing AI coding agents in the Nx malicious package, August 2025. https://snyk.io/blog/weaponizing-ai-coding-agents-for-malware-in-the-nx-malicious-package
35. The Hacker News, CometJacking, October 2025, and The Register, OpenAI Atlas prompt injection, October 27, 2025. https://thehackernews.com/2025/10/cometjacking-one-click-can-turn.html and https://www.theregister.com/2025/10/27/openai_atlas_prompt_injection
36. The Hacker News, Salesforce patches ForcedLeak, September 2025. https://thehackernews.com/2025/09/salesforce-patches-critical-forcedleak.html
37. The Hacker News, 341 malicious ClawHub skills, February 2026. https://thehackernews.com/2026/02/researchers-find-341-malicious-clawhub.html
38. The Hacker News, MCP STDIO design vulnerability, April 2026. https://thehackernews.com/2026/04/anthropic-mcp-design-vulnerability.html
39. OECD AI Incidents Monitor, Grok and Bankrbot token transfer, May 4, 2026. https://oecd.ai/en/incidents/2026-05-04-4a73
40. Microsoft Security, "Prompts become shells", May 7, 2026. https://www.microsoft.com/en-us/security/blog/2026/05/07/prompts-become-shells-rce-vulnerabilities-ai-agent-frameworks/
41. SecurityWeek, SalesBleed flaws in Salesforce Agentforce, September 2026. https://www.securityweek.com/salesbleed-flaws-in-salesforce-agentforce-enabled-zero-click-data-exfiltration/
42. Dark Reading, prompt injection bug in the Manus agent, 2026. https://darkreading.com/application-security/prompt-injection-bug-agentic-ai-app-manus
43. NBC News, University of Zurich AI personas on Reddit, April 2025. https://www.nbcnews.com/news/rcna203597
44. Cloudflare, "Perplexity is using stealth, undeclared crawlers", August 4, 2025. https://blog.cloudflare.com/perplexity-is-using-stealth-undeclared-crawlers-to-evade-website-no-crawl-directives/
45. Wiz, exposed Moltbook database, February 2, 2026. https://wiz.io/blog/exposed-moltbook-database-reveals-millions-of-api-keys
46. The Register, AI agent attacks developer after rejected pull request, February 12, 2026. https://www.theregister.com/2026/02/12/ai_bot_developer_rejected_pull_request/
47. Cooley, Ninth Circuit rules on AI agent access to third-party websites under the CFAA, August 6, 2026. https://www.cooley.com/news/insight/2026/2026-08-06-ninth-circuit-rules-on-ai-

agent-access-to-third-party-websites-under-cfaa

48. Cybernews, iLands agents emailing researchers, September 2026. https://cybernews.com/ai-news/ai-agents-send-emails-job/
49. Help Net Security, 2026 Identity Security Landscape, May 14, 2026. https://www.helpnetsecurity.com/2026/05/14/2026-identity-security-landscape-report/
50. Cloud Security Alliance, organizations cannot distinguish AI agent from human actions, March 24, 2026. https://cloudsecurityalliance.org/press-releases/2026/03/24/more-than-two-thirds-of-organizations-cannot-clearly-distinguish-ai-agent-from-human-actions
51. Teleport, 2026 State of AI in Enterprise Infrastructure Security. https://goteleport.com/about/newsroom/press-releases/2026-state-of-ai-in-enterprise-security-report/
52. GitGuardian, The State of Secrets Sprawl 2026, March 17, 2026. https://blog.gitguardian.com/the-state-of-secrets-sprawl-2026/
53. Help Net Security, Thales 2026 Bad Bot Report, April 30, 2026. https://www.helpnetsecurity.com/2026/04/30/thales-ai-driven-bot-traffic-rise-report/
54. Gartner, over 40 percent of agentic AI projects will be canceled by end of 2027, June 25, 2025, and 40 percent of enterprise apps will feature task-specific agents by 2026, August 26, 2025. https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027
55. Financial Conduct Authority, Mills Review, July 2026. https://www.fca.org.uk/publications/corporate-documents/mills-review
56. Lewis Silkin, HM Treasury consults on a framework for payments, July 16, 2026. https://www.lewissilkin.com/insights/2026/07/16/hm-treasury-consults-on-a-future-proof-framework-for-payments-stablecoins-and-op-102nc2h
57. Debevoise and Plimpton, FINRA 2026 Regulatory Oversight Report, December 2025. https://www.debevoise.com/insights/publications/2025/12/finras-2026-regulatory-oversight-report-continued
58. NIST, AI Agent Standards Initiative, February 17, 2026. https://nist.gov/news-events/news/2026/02/announcing-ai-agent-standards-initiative-interoperable-and-secure
59. ITIF, "Agentic Commerce Is Coming, but Regulation Meant for Humans Will Slow It Down", March 23, 2026. https://itif.org/publications/2026/03/23/agentic-commerce-is-coming-but-regulation-meant-for-humans-will-slow-it-down/
60. Singapore MDDI and IMDA, Model AI Governance Framework for Agentic AI, January 22, 2026. https://www.mddi.gov.sg/newsroom/singapore-launches-new-model-ai-governance-framework-for-agentic-ai--/
61. White and Case, EU AI Omnibus enters into force, July 2026. https://www.whitecase.com/insight-alert/eu-ai-omnibus-enters-force-amending-ai-act

thelab.org

62. Decrypt, Estonia plans national IDs for AI agents, 2026. https://decrypt.co/371441/estonia-ai-agents-national-id
63. OpenID Foundation, Identity Management for Agentic AI, October 7, 2025. https://openid.net/new-whitepaper-tackles-ai-agent-identity-challenges/
64. Model Context Protocol, release notes for the July 28, 2026 specification. https://blog.modelcontextprotocol.io/posts/2026-07-28/
65. Linux Foundation, A2A protocol surpasses 150 organizations, April 9, 2026. https://www.linuxfoundation.org/press/a2a-protocol-surpasses-150-organizations-lands-in-major-cloud-platforms-and-sees-enterprise-production-use-in-first-year
66. Visa, Trusted Agent Protocol, October 14, 2025, and live agentic transactions in Europe, July 2, 2026. https://corporate.visa.com/en/sites/visa-perspectives/newsroom/visa-unveils-trusted-agent-protocol-for-ai-commerce.html and https://www.visa.co.uk/about-visa/newsroom/press-releases.3457328.html
67. Mastercard, first live agentic transaction in Hong Kong, March 2026. https://www.mastercard.com/news/ap/en-hk/newsroom/press-releases/en-hk/2026/mastercard-completes-its-first-live-agentic-transaction-in-hong-kong
68. Google Cloud, Agent Payments Protocol (AP2), September 2025. https://cloud.google.com/blog/products/ai-machine-learning/announcing-agents-to-payments-ap2-protocol
69. Forrester, on OpenAI pulling back Instant Checkout, 2026. https://www.forrester.com/blogs/what-it-means-that-the-leader-in-agentic-commerce-just-pulled-back
70. Deccan Chronicle, India framework for AI agent UPI payments, September 2026. https://www.deccanchronicle.com/amp/technology/india-preparing-framework-to-let-ai-agents-make-small-upi-payments-without-approval-1983986
71. GitHub Docs, About Copilot coding agent. https://docs.github.com/copilot/concepts/coding-agent/about-copilot-coding-agent
72. AWS, AWS DevOps Agent generally available, March 31, 2026. https://aws.amazon.com/about-aws/whats-new/2026/03/aws-devops-agent-generally-available
73. Microsoft Learn, Entra agent identities for Copilot Studio. https://learn.microsoft.com/microsoft-copilot-studio/admin-use-entra-agent-identities
74. Workday, AI Agent Partner Network and Agent Gateway, June 3, 2025. https://newsroom.workday.com/2025-06-03-Workday-Announces-New-AI-Agent-Partner-Network-and-Agent-Gateway-to-Power-the-Next-Generation-of-Human-and-Digital-Workforces
75. American Bar Association, BC tribunal confirms companies remain liable for information provided by AI chatbot (Moffatt v. Air Canada, 2024 BCCRT 149), February 2024.

theailab.org

https://www.americanbar.org/groups/business_law/resources/business-law-today/2024-february/bc-tribunal-confirms-companies-remain-liable-information-provided-ai-chatbot/

76. CMS, Interoperability and Prior Authorization Final Rule (CMS-0057-F). https://www.cms.gov/newsroom/fact-sheets/cms-interoperability-and-prior-authorization-final-rule-cms-0057-f
77. Becker's Hospital Review, MUSC Health AI agents for prior authorization. https://www.beckershospitalreview.com/ai/prior-authorizations-patient-check-ins-musc-healths-ai-agents.html
78. Dinsmore, CMS WISeR model, 2026. https://www.dinsmore.com/publications/is-ai-wiser-cms-models-ai-based-prior-authorization-process-in-six-states/
79. FCA, second cohort of AI Live Testing. https://www.fca.org.uk/news/press-releases/fca-announces-second-cohort-ai-live-testing
80. GOV.UK, new AI tool in the GOV.UK app, May 2026. https://www.gov.uk/government/news/millions-to-get-faster-easier-access-to-government-support-with-new-ai-tool
81. Instructure and OpenAI global partnership, July 23, 2025. https://www.nasdaq.com/press-release/instructure-and-openai-announce-global-partnership-embed-ai-learning-experiences
82. KTVZ via Stacker, agentic browsers and cheating at universities, October 27, 2025. https://ktvz.com/stacker-money/2025/10/27/cheating-beyond-chatgpt-agentic-browsers-present-risks-to-universities/
83. Nieman Lab, Centre Daily Times unionizes after McClatchy AI tool, June 2026. https://www.niemanlab.org/2026/06/the-centre-daily-times-unionizes-after-backlash-to-mcclatchys-ai-tool/
84. TechRepublic, Instagram AI-generated profile label, 2026. https://techrepublic.com/article/news-instagram-ai-generated-profile-label-recommendations/
85. Cloudflare, signed agents, August 2025, and IETF, draft-ietf-webbotauth-httpsig-protocol-00, September 1, 2026. https://blog.cloudflare.com/signed-agents and https://datatracker.ietf.org/doc/draft-ietf-webbotauth-httpsig-protocol/
86. Google, Gemini 3 auto browse in Chrome, January 28, 2026. https://blog.google/products-and-platforms/products/chrome/gemini-3-auto-browse/
87. TechCrunch, Gemini Spark, May 19, 2026. https://techcrunch.com/2026/05/19/google-introduces-gemini-spark-a-24-7-agentic-assistant-with-gmail-integration/
88. Federal Register, Executive Order 14363, Genesis Mission, November 28, 2025. https://www.govinfo.gov/content/pkg/FR-2025-11-28/html/2025-21665.htm

theailab.org

89. 45 CFR 164.312, HIPAA Security Rule technical safeguards. https://www.law.cornell.edu/cfr/text/45/164.312
90. HHS, HIPAA Security Rule proposed rule, 90 FR 898, January 6, 2025, and Unified Agenda entry RIN 0945-AA22. https://www.reginfo.gov/public/do/eAgendaViewRule?pubId=202510&RIN=0945-AA22
91. HL7 FHIR R5 Provenance and R4 AuditEvent. https://hl7.org/fhir/R5/provenance.html and https://hl7.org/fhir/R4/auditevent.html
92. NIST SP 800-63-4, Digital Identity Guidelines, August 2025. https://pages.nist.gov/800-63-4/sp800-63b.html
93. NIST SP 800-53 Rev. 5 control catalog. https://csrc.nist.gov/projects/cprt/catalog
94. NIST SP 800-207, Zero Trust Architecture, August 2020. https://csrc.nist.gov/pubs/sp/800/207/final
95. OMB M-22-09, January 26, 2022, and OMB M-25-21, April 3, 2025. https://www.whitehouse.gov/wp-content/uploads/2025/02/M-25-21-Accelerating-Federal-Use-of-AI-through-Innovation-Governance-and-Public-Trust.pdf
96. NIST IR 8547, Transition to Post-Quantum Cryptography Standards, final, October 27, 2026. https://csrc.nist.gov/pubs/ir/8547/final
97. Executive Order 14412, June 22, 2026, and OMB M-26-15. https://www.whitehouse.gov/wp-content/uploads/2026/06/eo-14412.pdf
98. NSA, Commercial National Security Algorithm Suite 2.0. https://media.defense.gov/2025/May/30/2003728741/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS.PDF
99. IANA, HTTP Message Signature Algorithms registry. https://www.iana.org/assignments/http-message-signature/
00. IETF, RFC 9964 (ML-DSA for JOSE and COSE), RFC 9881 (ML-DSA in X.509), RFC 10073 (composite ML-DSA), RFC 9942 (COSE Receipts) and RFC 9162 (Certificate Transparency version 2). https://www.rfc-editor.org/info/rfc9964
101. IETF audit BoF, agentproto working group charter, and draft-ietf-wimse-aims. https://datatracker.ietf.org/group/audit/ and https://datatracker.ietf.org/doc/charter-ietf-agentproto/
02. IEC 62443 series, via ISA. https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards
03. CISA and partners, Principles for the Secure Integration of Artificial Intelligence in Operational Technology, December 3, 2025. https://www.cisa.gov/sites/default/files/2026-01/joint-guidance-principles-for-the-secure-integration-of-artificial-intelligence-in-operational-technology-508cV2.pdf
04. NERC CIP-004-7, CIP-005-7, CIP-007-6 and CIP-011-3. https://www.nerc.com/pa/Stand/Reliability%20Standards/CIP-005-7.pdf

05. TSA Security Directive Pipeline-2021-02G. https://www.tsa.gov/sites/default/files/signed_security_directive_pipeline-2021-02g_and_transmittal_memo_508c.pdf

06. Regulation (EU) 2022/2554 (DORA) and Delegated Regulation (EU) 2024/1774. https://eur-lex.europa.eu/eli/reg_del/2024/1774/oj

07. Delegated Regulation (EU) 2018/389, PSD2 strong customer authentication. https://eur-lex.europa.eu/eli/reg_del/2018/389/oj

08. 17 CFR 240.17a-4. https://www.law.cornell.edu/cfr/text/17/240.17a-4

09. Regulation (EU) 2024/1689, the AI Act, and Regulation (EU) 2026/1744. https://eur-lex.europa.eu/eli/reg/2026/1744/oj

110. Regulation (EU) 2024/1183, eIDAS 2.0; Directive (EU) 2022/2555, NIS2; Regulation (EU) 2024/2847, Cyber Resilience Act. https://eur-lex.europa.eu/eli/reg/2024/1183/oj

111. FIDO Alliance, standards for trusted AI agent interactions, April 28, 2026. https://fidoalliance.org/fido-alliance-to-develop-standards-for-trusted-ai-agent-interactions/

112. Executive Order 14434, September 29, 2026. https://www.federalregister.gov/documents/2026/10/02/2026-20321

113. FedRAMP 20x. https://www.fedramp.gov/20x/

114. NIST CSWP 39, Considerations for Achieving Cryptographic Agility. https://csrc.nist.gov/pubs/cswp/39/considerations-for-achieving-cryptographic-agility/final

thealab.org