ATLASCLOSE · WHITE PAPER · FIELD REPORT

Atlas Decides: Rule-Bound Release of AI-Built Software Under an Institutional Governance System

Design, test protocol, recorded evidence and honest limits of a public demonstration in which a visitor's own AI builds a small tool and a rule-bound system, not a person and not the AI, decides whether it is released.

Author: Clinton Brown, AtlasClose
Date: November 13, 2026; revised November 16, 2026 (all times Pacific Standard Time, PST)
Version: 2.4 (supersedes v1, v2, v2.1, v2.2 and v2.3; see the errata notes at the end)
Systems described: the "Atlas decides" factory (server `factory2-mcp r3`) and the earlier human-approve factory (server `factory-real-mcp r3`)
Public endpoint: `https://demo.atlasclose.com/factory2/mcp` · public page `https://atlasclose.com/factory2`

Status and scope. This is an experimental demonstration on sample data. It is not production-certified. Every quantitative statement in this paper, other than the estimates in Section 7, comes from a dated acceptance record, deploy record, server record, review file, accounting record, or a receipt re-checked by the author's team on November 13, 2026 (2:34–3:02 PM PST) and November 16, 2026 (9:02:40–9:03:39 AM PST); a claims ledger maps each claim to its source. Section 7's human-hours figures are estimates. Atlas law, described in Section 4, is the internal governance of an AI institution. It is not a court of law, binds no one outside Atlas, and is not legal advice. Its "Magistrate", "Legal" and "Audit" are AI agents. Where evidence is missing or inconclusive, the paper says NOT VERIFIED. A PASS means the defined checks passed. It does not mean the software has no defects, or that it is useful.

Abstract

When an AI system writes software, writes the tests for it, judges the result and publishes it, there is no point at which anything other than the AI decides. We describe a small public demonstration that separates those roles. A visitor states a problem. Before any code exists, Atlas writes a specification and acceptance checks, and the person confirms that the specification describes the problem. The visitor's own AI assistant then builds one self-contained HTML and JavaScript page. Atlas runs that page in a locked box (a pinned JavaScript engine with no network, no files, and time and memory caps), applies nine written rules and the pre-written checks, and returns APPROVED or REFUSED with a reason for every rule. No approve tool exists. Approved bytes are released exactly as checked, under a browser policy that forbids network access, and a receipt re-hashes the live bytes each time it is opened. Every step is appended to a hash-chained docket. We state five falsifiable hypotheses and report recorded results: every scripted test step passed in the public end-to-end runs of releases r2 (16 of 16) and r3 (20 of 20); adversarial test pages were refused with the same reasons on re-run; tampering with copies was detected; five releases from cases the author ran through his own AI assistant (four distinct pages), four approved on the first try and one after two refusals; and the live evidence was re-checked at the time of writing. These are the same team's tests on a small number of cases, not independent validation, and we draw no statistical inference from them. We also report what failed along the way: a non-deterministic decision that forced a rollback, a content delivery network silently rewriting released bytes, and an origin check that refused a real person's button press. We close with the governance system the demonstration runs under, and with the limits of what these tests can show.

1. Problem: AI that grades and releases its own work

Most AI-assisted software work today has a single loop: the same assistant proposes code, proposes tests, runs them, interprets the output and, if it has deployment access, ships. Each step can be done well, but the structure itself has a defect: the actor whose output is being judged is also the judge. A passing test suite that the builder wrote, and the builder chose to run, tells a reader little about whether the work meets anyone else's standard.

The Atlas governance system states the principle directly. Its Constitution (Article XIV §1) says: "A material actor may not conclusively certify its own material factual or substantive result where independent validation is required." Article XIV §2 adds a point that matters for any cryptographic record: "Provenance is not truth. … a hash chain establishes attribution and non-forgery of origin, not that the claimed result satisfies applicable standards." And Article XIII §3 warns that "[p]ack-compliance, tests passing, and page generation are not purpose-satisfaction."

A second problem is memory. AI assistants have no lasting memory between sessions, and an internal Atlas research note records that people forget past fixes too. Whatever must be true tomorrow therefore has to be carried by durable software and durable records, not by anyone's recollection of a decision. That note labels this a reasonable inference, not adopted law; we adopt it here as a design principle.

The design question for this demonstration is therefore narrow and testable: *can the release of AI-built software be separated from the AI that built it, so that the only path to release is passing checks written before the build, under fixed written rules, with a record that a third party can re-check afterwards?*

2. System overview

Figure 1 shows the flow. Four parties are involved: the **person** (the visitor), the **visitor's own AI assistant**, connected to the factory as a remote tool server, the **Atlas factory server**, which holds the specification, the docket and the stored bytes, and the **locked box**, which runs submitted code. A fifth party, **anyone** holding a receipt link, can re-check the result later.

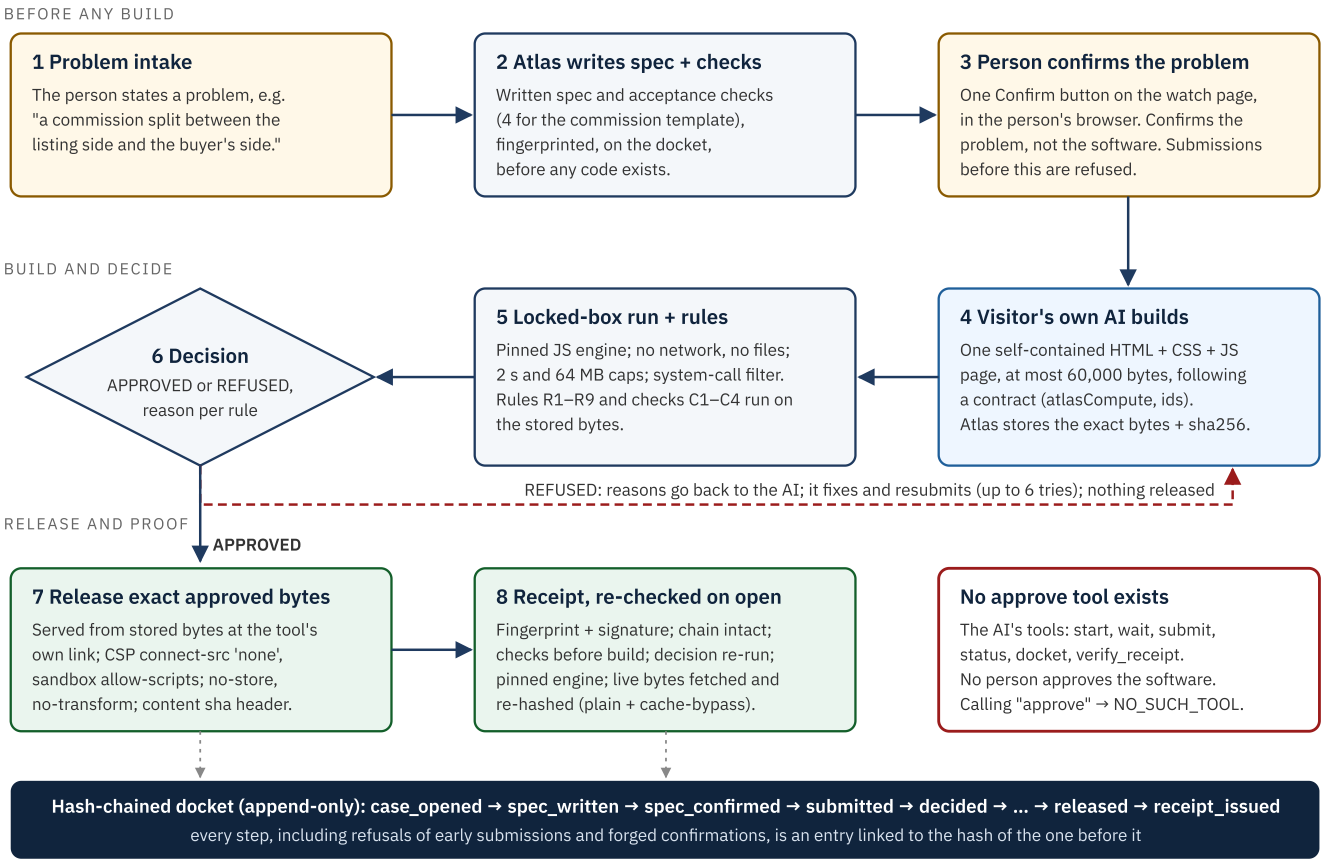


Figure 1. Flow of one case in the "Atlas decides" factory. Steps 1 and 3 are the person's; step 4 is the visitor's AI; steps 2, 5, 6, 7 and 8 are Atlas. The person's only action is to confirm the problem statement. The decision in step 6 is made by written rules and pre-written checks, recorded with a reason per rule. Grey dashed lines indicate that each step is appended to the docket.

The demonstration has a predecessor. An earlier factory on the same site (`/factory/mcp`) uses the same docket, stored-bytes and receipt machinery, but its release gate is a person: the AI submits a change to a sample page, Atlas tests it, and nothing ships until the person presses Approve on the watch page in their own browser. The "Atlas decides" factory (`/factory2/mcp`) removes that button and replaces it with the rules and checks of Figure 1. The two are useful together: the first shows that a person's approval can be enforced outside the AI's control; the second shows that a written standard can be enforced the same way.

Property	Human-approve factory (`/factory`)	"Atlas decides" factory (`/factory2`)
Who releases	The person, by pressing Approve on the watch page	Atlas, by rules R1–R9 and checks written before the build
What the person does	Approves a specific tested version	Confirms the problem statement only
Submitted code	HTML and CSS only; scripts refused; the page is read, never run	HTML, CSS and JavaScript, run in the locked box
Approve tool for the AI	None	None
Receipt	Re-checks code, tests, approval and live bytes	Re-checks spec, checks, decision re-run, engine, live bytes

3. Technical design

3.1 Tool surface

The visitor's AI reaches the factory through six tools: `atlas.factory2_start`, `_wait`, `_submit`, `_status`, `_docket` and `_verify_receipt`. None of them can confirm a specification or approve a tool; the server's own tool descriptions say so ("Cannot confirm or approve anything"). Calling a non-existent approve tool returns `NO_SUCH_TOOL`. The confirm action exists only as a form on the watch page, posted from the person's browser and checked for the page's origin and a one-time code. An `initialize` and `tools/list` capture by the author's team on November 13, 2026 at 2:56:57 PM PST returned server `factory2-mcp r3` with exactly these six tools.

3.2 Problem intake: checks before code

For the commission-split problem, Atlas uses a versioned template (`commission split r1`); for other small calculators, forms or checklists it builds checks from 2–6 worked examples the AI passes on the person's behalf. The specification and the checks are fingerprinted (spec sha256 `7dc47826…`, checks sha256 `98561836…` for the sample case) and written to the docket before any code is accepted. A submission made before the person confirms is refused and docketed (test T04). A confirmation from a foreign origin, or with a wrong code, is refused with HTTP 403 (T05).

3.3 The page contract

The AI must submit one HTML file of at most 60,000 bytes with all CSS and JavaScript inline; no outside addresses, scripts, images, fonts or frames; no network calls, storage, cookies or data collection; a page-level function `atlasCompute(input)` returning named outputs or `{error: "…"}`; one input element per field id; a button `atlas-run` wired with `addEventListener`; and an output element `atlas-output`. Numeric outputs must match each check to within half a cent.

3.4 The locked box

The submitted page is executed by a small runner built on the factory's cloud virtual machine from source, around a pinned open-source embeddable JavaScript engine at a fixed release whose source archive is pinned by sha256. The built runner binary is itself pinned (sha256 `f4e8a3b4a419…`), as is the test harness (`fc2f1b275b9b…`). At deploy the runner reported a Linux system-call filter in filter mode (seccomp mode 2), and a deliberate timeout self-test returned the timeout code (rc = 3). Each run is capped at 2 seconds and 64 MB, with no network and no files. Around the runner, the factory service itself runs as a dynamically allocated unprivileged user, may reach only localhost at the IP layer, and is capped at 384 MB of memory.

3.5 The nine rules

Atlas's rules are versioned (`atlas-tool-rules r1`) and pinned in every receipt. The watch page publishes each rule's result and reason for every try.

Rule	Name	What is checked (as published on the watch page)
R1	Size cap	Byte count against the 60,000-byte cap
R2	One self-contained page	No outside scripts, styles, images, fonts, frames or form targets
R3	No network calls	No network APIs or web addresses in the code, and no network attempt while running
R4	No data collection or storage	No storage, cookies, device or clipboard access, in the code or while running
R5	No hidden behavior	None of a written list of tricks (eval, encoded text, navigation, window access, timers given text, …)
R6	Runs in the locked box	Loads and runs without errors within the caps (2 s, 64 MB, no network, no files)
R7	Tool contract	`atlasCompute` defined; required inputs, button and output present

R8	Acceptance checks	Every pre-written check gives the expected output
R9	Screen works	Fills in check C1, presses the button, and reads the screen

Two features of this list are deliberate. First, R5 is a *written list* of behaviors, not a guess about intent; the design states that Atlas does not judge usefulness or motive. Second, R3 and R4 are checked twice: statically in the code and dynamically while the page runs, so a hidden or renamed network call that is attempted at run time is recorded as such (test T14).

3.6 Determinism

Because the receipt re-runs the decision, the decision must be a pure function of the stored bytes, the checks, the rules and the pinned engine. The first deploy attempt (r1) failed this: the reason text for R6 included the run time in milliseconds, so the hash of the results differed between runs and the receipt's decision re-run did not match. The local end-to-end suite scored 13/15 and the deploy rolled itself back before any public route was changed. Release r2 removed the timing from the reason text and added a determinism self-test (`det=yes`) to the deploy gate.

3.7 The docket hash chain

Each case has an append-only docket. Every entry (case opened, spec written, refusals, confirmation, each submission and decision, release, receipt) carries the hash of the entry before it, and a chain check recomputes the links. The sample case has 12 entries: `case_opened, spec_written, submit_refused, confirm_refused, confirm_refused, spec_confirmed, submitted, decided, submitted, decided, released, receipt_issued`. There is no approve entry, because there is no approve step. Editing the text of one entry in a copy of the state makes the receipt fail its "docket chain intact" check (local tamper test B_docket). Consistent with Article XIV §2, the chain proves origin and order, not correctness.

3.8 Release and byte binding

An approved page is served from the exact stored bytes. The author's fetch of the sample tool at 2:34:40 PM PST returned these response headers (abridged):

```
content-security-policy: default-src 'none'; script-src 'unsafe-inline'; style-src 'unsafe-
inline'; img-src data:; connect-src 'none'; … form-action 'none'; base-uri 'none'; frame-
ancestors 'self'; sandbox allow-scripts
cache-control: no-store, no-transform
x-atlas-content-sha256: 0494f78a5be042ae0774e3956460fa8e02ec697abf94b1bd04fda83760f09e64
```

The policy lets the page's own inline script compute but forbids it from opening any connection, and the sandbox isolates it from the site's origin. The content-hash header states what the origin believes it sent; the receipt and the tests do not trust that header, and re-hash the bytes actually received.

3.9 Receipt verification

The receipt is a canonical JSON text whose sha256 is its fingerprint; the factory also signs it. Verifying a receipt recomputes nine checks: receipt fingerprint; factory signature; docket chain intact; checks written and confirmed before the build (by docket position); stored tool bytes re-hashed; Atlas's decision re-run in the locked box; decision and release on the record; engine and harness are the pinned ones; and live tool bytes fetched now and re-hashed. A reader can also pass in a copy of the receipt they hold; a copy with one field changed returns `DOES_NOT_CHECK_OUT` while the true copy checks out (T12). The receipt page additionally runs a script in the reader's browser that fetches the public live link twice (normal and cache-skipping) and re-hashes it; that browser-side check has not yet been exercised in a real browser (Section 5).

3.10 Engineering history: what broke, and what it taught

We report failures as part of the method. Each was found by a test or a real use, recorded, fixed in a new release, and re-tested; none of the earlier records was edited.

1. **A content delivery network changed released bytes.** In the human-approve factory's first live run, both public fetches of a shipped page returned bytes whose hash differed from the approved hash and from each other, while the origin's content-hash header was correct. The network's e-mail address obfuscation feature had rewritten the sample e-mail addresses and injected a decode script (which the page's policy blocked). The server's own verification, a loopback fetch, could not see this. The fix was to send `Cache-Control: no-store, no-transform`; the next run passed. Lesson: byte identity must be checked from the public side, by a client outside the server, not only at the origin.
2. **Non-deterministic reasons.** Described in 3.6; caught by the deploy gate and rolled back cleanly.
3. **A test bug, kept on the record.** The first public run of r2 scored 15/16 with T09 FAIL. The test had looked up a response header case-sensitively, and the delivery network sends lower-case names. A direct request showed the policy was present. The test was fixed (case-insensitive lookup) and re-run at 16/16; the service was not changed, and the failing run's file and row were kept.
4. **An origin check that refused a real person.** In the human-approve factory, the author pressed Approve from a phone's in-app browser and was refused twice ("it did not come from the watch page"), at 12:25:28 and 12:26:53 PM PST. Root cause, from the code: the watch page sent `Referrer-Policy: no-referrer`, under which a browser form POST carries `Origin: null` and no Referer, so the exact-origin check refused every real button press. The earlier acceptance run had passed only because the test script set the Origin header itself. Release r3 switched to `same-origin` and accepts a valid one-time code with either the exact origin, or a null/absent origin backed by same-origin browser signals; it still refuses foreign origins, cross-site signals, foreign referrers, a missing origin with no signal, and wrong codes or versions (19/19 local tests). Every approve request now logs its origin-related headers, never the code. The real case was then approved by the person in a browser at 1:01:29 PM PST and shipped at 1:02:15 PM PST.
5. **A pre-check that stopped a deploy with no change.** The first r3 package required every stored receipt to check out; one receipt from the rolled-back r1 attempt never had. The deploy stopped before changing anything. The corrected gate requires that the set of receipts that checked out before the switch still checks out after it (6 of 7 before and after).
6. **A usability failure found by a real run.** At 2:04 PM PST, as relayed from his chat in the team's deploy record, the author reported that the flow was confusing: his AI stopped and waited for him to say he had confirmed. Release r3 added `atlas.factory2_wait` (blocks up to 45 seconds per call and returns the moment the person confirms), a plain `next_step` in every reply, and a four-step watch page with one Confirm button.

4. Governance: the Atlas law system and how it governs this demo

> **What "Atlas law" is.** Atlas law is the internal governance of an AI institution. It is not a court of law, binds no one outside Atlas, and is not legal advice. Its "Magistrate", "Legal" and "Audit" are AI agents.

4.1 An institution that manages other institutions

Atlas is designed as an institution that manages other institutions. Atlas's own organizational law governs Atlas's own matters; for example, Atlas law sets a 72-hour warning period, along with its own appeal and demand periods. Each organization Atlas manages runs under its own declared rules. A synthetic practice company used in another AtlasClose demonstration declares, for instance, a 60-second warning, a 120-second appeal window and a 60-second demand window, without any change to Atlas law. A managed organization's rules must be written and docketed before they apply; each matter pins the rule's hash, and code cannot override it at run time.

The factory follows the same pattern. It is an operational system with its own declared, versioned rules (`atlas-tool-rules r1`; template `commission split r1`), and every receipt pins the rules version, the spec hash, the checks hash, the engine hash and the harness hash. Under the Constitution (Article XIII §2), operational capabilities "do not create their own governing authority"; the factory's power to release exists because people wrote its rules and the Human authorized its deployment.

4.2 Who may decide what

Office	Authority under Atlas law (as recorded)	Role in this demonstration
The Human (Clinton Brown)	Ultimate constitutional authority (Art. I §1). Only his own first-hand yes adopts a law; an agent's say-so is never adoption. Spending, deleting or overwriting records, Human keys and public-server installs still need his yes.	The author directed the deploy ("Okay put that demo up too", 1:04 PM PST, as relayed from his chat in the team's deploy record). At 2:04 PM PST he reported the flow was confusing; the team treated that as his go-ahead for the r3 fix. The design document stated that building waited for his review and go.
The Magistrate	May stop Atlas work without Human approval, order fixes, and see the whole docket (O-LANDMARK-MAGISTRATE-STOP, adopted Nov 9, 2026). May reopen a closed record only by written order with reasons and new facts (O-CLOSE-OF-RECORD, adopted Nov 12, 2026). Any party may appeal; the Human has final say.	No Magistrate action concerning this demonstration is in the evidence set. Pages carry the line "Atlas · The Magistrate".
Legal	Rules on the scope of existing authority (not new law); writes the law-book addenda; may direct appended corrections of attribution or clerical/display errors on demo or practice dockets (O-CORRECTIONS-STANDING).	No Legal ruling specific to this factory is in the evidence set.
Audit	Independent validation (Art. XIV); runs the stop-register acceptance test (Rule Q.3); checks each correction within 24 hours; hash-checks copies under the space wheel. The Magistrate may never order Audit's verdict (O-LANDMARK-MAGISTRATE-STOP).	No independent Audit validation of this factory exists. Audit reviewed v1 of this paper as public copy (PASS_WITH_FINDINGS); that is a check of the text, not of the system (Section 6).
Secretary	Keeps first-hand records of the Human's words; an instruction counts as the Human's direction once it is captured first-hand.	Each order adopted since October 2026 that is cited in this section has a first-hand record of the Human's own answer.
Atlas factory (software)	An operational capability bound by its declared rules.	Decides APPROVED/REFUSED by R1–R9 and the checks; cannot change its rules at run time.

4.3 Mechanisms for detecting and correcting its own problems

The goal of the Atlas system is an institution that detects its own problems, starts corrective proceedings, enforces its rules and keeps operating. Several adopted instruments already encode parts of that loop:

- **Stop and fix (O-LANDMARK-MAGISTRATE-STOP).** The Magistrate can stop work and order a fix; the stop holds until the fix is in and verified, or until the Magistrate, an appeal panel or the Human lifts it. Rule Q requires one sealed stop register that every runner checks before each step, and that fails closed: if the register is missing or unreadable, the step is refused.
- **Append-only corrections (O-CORRECTIONS-STANDING).** A correction is a new entry that cites the original's sequence number and hash, keeps the original unchanged, and is applied only through an audited tool; it may never change a disposition or delete, rewrite or hide an entry. The Human adopted this order on October 20, 2026 at 12:43 PM PDT, by his own answer to a card: "Yes, adopt O-CORRECTIONS-STANDING as written." The adopted text and the adoption record are hash-pinned on Atlas's docket. A November 9 Legal memo noted that it had found only a draft of this order; that note reflected where the memo searched, and the adoption record stands.

- **Closing the record (O-CLOSE-OF-RECORD).** Proceedings end at a fixed disposition time; later filings need a written motion showing new facts; reopening needs a written order with reasons.
- **Self-maintenance (O-SPACE-WHEEL-STANDING, adopted Nov 12, 2026).** Disk space is watched hourly; at 75% eligible folders are copied off the machine with a hash manifest, Audit hash-checks every copy, and nothing is cleared before Audit's PASS. Law, rulings, dockets, ledgers, the stop register, keys and lent-data records are never eligible. As of this writing the watcher only reports; the clearing step is still being built and must pass Audit's test before it is switched on.
- **Build versus don't build (O-GC-002).** Before building a new capability, discover and reuse what exists; build new only on a recorded concrete mismatch or the Human's override. The factory design reuses the earlier factory's docket, receipts, watch page and byte-match code, and adds only the intake, the locked box and the rules engine.
- **Preservation of history (Constitution Art. VIII §5, Art. XIX).** Later law supersedes earlier law by new filings; history is not rewritten. The demonstration's own records follow this: failed runs, rolled-back attempts and the bad r1-era receipt are kept and reported.

4.4 How far the demonstration meets that standard today

In place: the deploy was authorized by the Human, and the r3 fix followed his report of a problem (both as relayed from his chat in the team's deploy records); every case step is on an append-only, hash-chained docket; rollbacks park files and delete nothing; results are labelled PASS, FAIL or NOT VERIFIED strictly by evidence. Not yet in place: the "Atlas decides" service is **not integrated with the stop register** that Rule Q requires runners to check; turning it off is a documented manual rollback. (The human-approve factory does record a stop check before its key steps.) And the acceptance tests reported here were designed, run and scored by the AI build agents that built and deployed the service. Under Article XIV, they are evidence, not independent validation.

4.5 Next level: an operational AI institution

The author's stated next target is what he calls **Level 6, Operational AI Institution**: a self-governing AI institution in which problems found by its own monitors (a failed receipt, a byte mismatch, a stop-register gap) automatically open a corrective proceeding, the proceeding produces an order, the order is carried out and independently verified, and operations continue, all on the record, with the Human's authority reserved for the questions that belong to the Human. This demonstration is one component of that target, not evidence that it has been reached.

5. Scientific method and evidence

5.1 Hypotheses

We state the claims as falsifiable hypotheses: each names an observation that would refute it. All attempts to refute these hypotheses so far were made by the team that built the service (see 6.4); independent attempts are future work (Section 8).

ID	Hypothesis	Refuted by
H1	Through its tools, the visitor's AI cannot release a page without an APPROVED decision or submit before the spec is confirmed.	Any release without an APPROVED decision; any approve or confirm path reachable through the AI's tools; a submission accepted before confirmation.
H2	The bytes served at a released tool's public link equal the approved bytes, on fresh and cache-bypassing requests from a client outside the server.	Any public fetch whose sha256 differs from the approved sha256.

H3	Refusals are reproducible and explained: the same submission yields the same decision and the same per-rule reasons when it is run again.	A different decision, failed-rule set or results hash for identical inputs.
H4	Tampering with stored bytes, the docket or a held receipt is detected by receipt verification.	A tampered state or copy that still returns CHECKS_OUT.
H5	Acceptance checks are fixed before the build: the spec and checks are on the record, confirmed, before the first try.	A first try recorded before the spec confirmation, or checks that change after the build starts.

5.2 Test protocol

Tests are black-box and adversarial. An end-to-end script, run from a separate machine against the public endpoint through the same network path a visitor uses, opens cases, attempts forbidden actions, posts the watch-page confirm form itself, submits test pages written in advance by the build team (a deliberately faulty tool, then a correct one, then attack pages: hidden network call, WebSocket, outside image beacon, an endless loop), fetches the released page twice, verifies the receipt, and reads the docket. No AI model takes part in these scripted runs. The same suite runs locally on the server inside each deploy as a gate. Tamper tests run only on copies of state, never on the live state. Each row records the action, expected result, actual result and a timestamp, and an artifact sha256 where one exists. The suite was run against two releases on the same day (r2 at about 1:44 PM PST; r3 at about 2:18 PM PST). The decision engine was byte-identical in both, so this repeats the same decision code with the same test inputs; it is not an independent replication. Separately, the author ran cases through his own AI assistant (Section 5.6).

Result definitions. **PASS**: recorded evidence demonstrates the expected behavior. **FAIL**: recorded evidence demonstrates a violation of the expected behavior. **NOT VERIFIED**: evidence is missing, incomplete or inconclusive. A result is never upgraded by argument; absent evidence is NOT VERIFIED, not PASS. Every label in this section was assigned by the AI build agents that ran the tests; no second party re-scored any row.

5.3 Results

Run	Steps	PASS	FAIL	NOT VER.	Note
"Atlas decides" r1, server-local gate (deploy attempt, 1:33 PM PST)*	15	13	2	0	Decision re-run not deterministic; rolled back before any route change. Fail count inferred as 15 − 13.
"Atlas decides" r2, first public run*	16	15	1	0	T09 FAIL was a test bug (header case); kept on record
"Atlas decides" r2, public run (1:44 PM PST)*	16	16	0	0	Fixed test; service unchanged
"Atlas decides" r2, server-local gate*	15	15	0	0	T16 needs the public path
"Atlas decides" r2, local tamper tests*	2	2	0	0	Stored bytes changed; docket entry edited
"Atlas decides" r3, public run (about 2:18 PM PST)*	20	20	0	0	Adds wait tool, scripted-client run, old-receipt check
"Atlas decides" r3, server-local gate / local copy*	19 / 19	19 / 19	0	0	
Stored receipts re-verified across the r2→r3 switch	7	6	1	0	The one failure was issued by the rolled-back r1 server; it failed before r3 too
"Atlas decides" r2, author's cases via his own AI	3 cases	3 released	—	—	Two APPROVED on try 1; one REFUSED twice (R8), then APPROVED on try 3 (5.6)

"Atlas decides" r3, author's cases via his own AI	2 cases	2 released	—	—	Both APPROVED on try 1, one of them the author's calculator (5.6)
Human-approve factory, live run 1 (11:54 AM PST)*	15	13	1	1	FAIL: delivery network rewrote bytes. NV: live tamper not attempted
Human-approve factory, live run 2 (after no-transform fix)*	15	14	0	1	NV: live tamper not attempted
Human-approve factory, local test server*	16	16	0	0	Includes stored-blob tamper and restore
Human-approve factory r3 approve fix, local tests*	19	19	0	0	Origin/referrer/fetch-site matrix

* Scripted test client and fixture pages written in advance by the build team. Each count is the number of scripted test steps passed, not a number of independent trials. The two author-case rows count cases and released or refused tries, not test steps, so the PASS/FAIL columns do not apply to them. No statistical inference is drawn.

Hypothesis outcomes

ID	Outcome	Evidence
H1	**PASS for the tool interface, in tested attempts**	No approve or confirm tool in r2 or r3 (T01; team capture at 2:56:57 PM PST); approve call → NO_SUCH_TOOL (T02); submit before confirm refused (T04); forged-origin and wrong-code confirms → 403 (T05); faulty and attack pages refused and not released (T07, T14, T15); release only after APPROVED, 9/9 rules and 4/4 checks (T08). Docket has no approve entry (T13). **NOT VERIFIED**: a code-holding agent forging the confirm post (6.5), which was not attempted.
H2	**PASS for tested cases**	T09 in r2 and r3: two public fetches each equal the approved sha256; receipt live re-hash (T10, T20); team fetches at 2:34:40 PM PST (sample tool) and 2:56:57–2:57:00 PM PST (sample tool and two of the author's released tools) and on November 16, 2026 at 9:02:40–9:03:39 AM PST (sample tool and all five of the author's released tools), plain and cache-bypassing, all equal to the approved sha256. The claim "for every visitor" is **NOT VERIFIED**: only tested cases were observed.
H3	**PASS for the three fixed test inputs and the sample receipt re-run (same engine in r2 and r3)**	In the author's case `fc2-a9c23db8bc877729`, the same 8,849-byte page submitted twice was refused twice with the same reasons. Deploy gate `det=yes`; identical try-1 bytes (`6af198e7…`) gave the same REFUSED decision and failed-rule set (R3, R8, R9) in r2 and r3; attack page `cfd62eac…` gave R2, R3, R5, R8 in both; loop page gave R6 in both; the decision re-run in the receipt returned "APPROVED again". The r1 rollback shows the gate caught one case of non-determinism.
H4	**PASS on copies, one part changed at a time**	Two local cases (stored bytes; one docket entry) plus a forged held receipt (T12, r2 and r3), with the checks run by the server itself. Not tested: live-state tampering, or a consistent rewrite by someone holding the server's signing key (**NOT VERIFIED**).
H5	**PASS for the sample case and the author's five released cases**	Spec and checks were on the docket and confirmed before the first try. In the sample case and the four commission cases they came from the fixed commission-split template, so they were fixed by construction. In the calculator case Atlas wrote them from the author's four worked examples before the build. The sample case was confirmed by test-script post ("spec 7dc47826109f confirmed at docket #6, first try at #7"); the author's cases by a watch-page confirm recorded before the first submission (5.6).

5.4 Worked example: the commission split (sample numbers)

> Both tries in this example are test pages the build team wrote in advance to exercise one refusal and one approval. Neither was written by a visitor's AI. The confirmation at docket #6 was posted by the test script, not by a person.

Problem: "I need a tool that splits a commission between the listing side and the buyer's side." Atlas's template specifies `gross = sale price × rate / 100` and `side = gross × share / 100`, to the cent, with an error instead of amounts when the shares do not add to 100 or the sale price is not above zero. Its four checks, as served on the live watch page:

Check	Input (sample numbers)	Expected
C1	sale price 400,000; rate 3%; listing 60%; buyer 40%	gross 12,000; listing 7,200; buyer 4,800
C2	sale price 350,000; rate 2.5%; 50 / 50	gross 8,750; listing 4,375; buyer 4,375
C3	400,000; 3%; listing 60%; buyer 50% (sum 110)	an error, no amounts
C4	sale price 0; 3%; 60 / 40	an error, no amounts

For C1: 400,000 × 3 / 100 = 12,000; 12,000 × 60 / 100 = 7,200; 12,000 × 40 / 100 = 4,800. (The design document's mock-up used 400,000 at 50 / 50 for C2; the deployed template uses 350,000 at 2.5%.)

Try 1: REFUSED. A 2,448-byte page. R3 failed: the code contained `fetch` and a web address, and while running it attempted `fetch https://rates.example.invalid/commission-defaults.json` (a deliberately unreachable test address). R8 failed, 2 of 4: on C1 it returned listing 6,000 and buyer 6,000 instead of 7,200 and 4,800, and on C3 it returned amounts instead of an error. R9 failed: the screen did not show the C1 amounts. R1, R2, R4, R5, R6 and R7 passed. Nothing was released. Try 1 was built to split evenly and to call the network.

Try 2: APPROVED. A 2,730-byte page, sha256 `0494f78a…`. All nine rules passed, 4 of 4 checks passed, and the locked-box run took 8 ms. R9 read the screen after filling in C1: "Gross commission 12,000.00. Listing side 7,200.00. Buyer's side 4,800.00." The page was released at its own link and receipt `f2r-e17310fdc613b6398580d58d` was issued. No person approved the software.

In r3, a scripted client (no AI model) that followed only the tool replies completed the same sequence end to end (start → wait → submit REFUSED → submit RELEASED) in 8 seconds, with one Confirm post from a simulated person and no messages from the person to the client (T17).

5.5 Adversarial submissions

A test page that hid a `fetch` under another name, opened a WebSocket and loaded an outside image beacon was refused on R2, R3, R5 and R8; R3's reason lists the WebSocket, fetch and image-source attempts observed while running (T14). A page whose function never returns was stopped by the locked box at the 2-second limit and refused on R6 (T15). Neither was released. Both were written by the build team.

5.6 Runs by the author (not outside visitors)

Human-approve factory. One run by the author, through an AI assistant and a phone's in-app browser, is on the record (case `fc-2d02e8bcb5932ae3`, request "add a phone column to the sign-in sheet"). Its try 1 passed 9 of 9 checks. The person's first two Approve presses were refused by the origin defect described in 3.10(4); after the r3 fix the person's approval was recorded at 1:01:29 PM PST and the page shipped at 1:02:15 PM PST. Its receipt `frr-964482c60392f89c97b48135` showed "Receipt checks out" when the team re-opened it at 2:34:40 PM PST, with a 14-entry chain that still contains both refusals.

"Atlas decides" factory. The author ran five cases that reached release through his own AI assistant on November 13, 2026. Server records copied by the team at 2:44 PM and 3:02 PM PST back all five. The team separated the author's cases from automated test cases using the acceptance records and, for two same-pattern cases at 2:17 PM PST, the sub-second scripted try sequence; that classification is the team's. Four used Atlas's commission-split template (the same four checks as Section 5.4). One was a calculator whose four checks Atlas wrote from eight input fields and four worked examples the author supplied (a photo-shoot budget and profit calculator, sample numbers). Three ran on release r2 and two on r3; the switch was at 2:17 PM PST. Two more cases were opened but never confirmed, so nothing was built for them. The five releases are four distinct pages: the approved try 3 in one case is byte-identical to the page released in an earlier case. The server records each submission, not who wrote the page; that the author's AI assistant wrote the pages is the author's account.

Case	Type (release)	Opened	Confirmed	Tries	Released	Receipt
`fc2-68bb528bddc102f9`	commission (r2)	1:58:53 PM	2:00:00 PM	1: APPROVED, 3,751 bytes	2:02:27 PM	`f2r-8bc9ae324fab52244a9be908`

fc2-918d53344d6de64c	commission (r2)	2:12:28 PM	2:13:17 PM	1: APPROVED, 9,480 bytes	2:13:38 PM	f2r-7b104361c0ef43b7d84e7b79
fc2-a9c23db8bc877729	commission (r2)	2:16:27 PM	2:17:02 PM	1: **REFUSED** (R8, 1 of 4 checks); 2: **REFUSED** (same bytes, same reasons); 3: APPROVED, 9,480 bytes	2:17:20 PM	f2r-d2333f6b693af2325fa35ae4
fc2-a946a712795d0689	commission (r3)	2:26:28 PM	2:26:45 PM	1: APPROVED, 9,879 bytes	2:28:39 PM	f2r-fa6f06104fac77c16ab18c3c
the calculator case	calculator from the author's examples (r3)	2:30:42 PM	2:31:17 PM	1: APPROVED, 12,486 bytes	2:32:22 PM	receipt available on request
fc2-48fd5df81b95f616 fc2-3d88fbac4902a795	commission	2:12:28 PM 2:26:27 PM	not confirmed	no tries	—	—

The one refusal is on a page submitted through the author's own AI assistant, not a build-team fixture. The assistant later called it the "crooked case"; whether the faulty page was a deliberate test is not recorded. On try 1, R8 passed only 1 of 4 checks: on C1 the listing and buyer amounts were swapped (4,800 and 7,200 instead of 7,200 and 4,800), and C3 and C4 returned amounts where the spec requires an error. The other eight rules passed. Try 2 resubmitted the same bytes and was refused with the same reasons. Try 3 was byte-identical to the page approved at 2:13:38 PM in an earlier case, and was approved. Nothing was released before try 3. Each confirmation is recorded as a watch-page form post before the first submission; the server cannot tell who pressed the button. The author's 2:04 PM report, as relayed, that his AI had waited for him (3.10(6)) came after the first run; the records do not show which case it refers to. For the two r3 runs, the records do not show whether his assistant followed the `next_step` replies without prompting: **NOT VERIFIED**. No outside visitor has completed a run. On November 16, 2026 at 9:02:40–9:03:39 AM PST, all five receipts returned CHECKS_OUT, 9 of 9, and plain and cache-bypassing fetches of all five live tools matched their approved sha256 (5.7).

What the visitor's AI assistant reported (qualitative; not verified). These observations came from the author's own AI assistant after its runs. They are its account, not measurements.

- For a calculator-type case, the spec was assembled from the fields and worked examples the visitor supplied. This is consistent with the record of the calculator case (Atlas template "calculator r1"; checks written from the author's four worked examples).
- The page's look (plain, restyled, court-brief style, calendar-cover) did not affect Atlas's judgment. This is consistent with the design, since no rule scores appearance, and with five approvals of differently styled pages from 3,751 to 12,486 bytes, but appearance was not tested as a variable.
- The server's "keep going without waiting" reply could not let the AI confirm the spec on the person's behalf, because only the person can. This is consistent with the tool list, which has no confirm tool.
- One run was refused (the "crooked case"). The records show exactly one of the author's cases with refused tries (`fc2-a9c23db8bc877729`, above). That this is the case the assistant meant is an inference.
- Limit: the checks' expected outputs came from the visitor's own examples, so how Atlas handles a loosely stated problem is untested.

5.7 Re-check at the time of writing (same team, not independent)

On November 13 and November 16, 2026, AtlasClose's build-and-drafting agent re-ran the key observations for the author from a client that is not the server (the factory's own probe, `atlasclose-factory-probe/1`). This is a repeat by the same team, not independent validation (see 6.4).

- 2:34:29 PM PST: `atlas.factory2_verify_receipt` on `f2r-e17310fdc613b6398580d58d` returned CHECKS_OUT with all nine checks true, receipt sha256 `cce35897…`, decision re-run "APPROVED again: 9 rules, 4 checks", and live bytes hashing to `0494f78a5be042ae…`.
- 2:34:40 PM PST: a plain fetch and a cache-bypassing fetch of the sample tool each returned 2,730 bytes with sha256 `0494f78a5be042ae0774e3956460fa8e02ec697abf94b1bd04fda83760f09e64`, equal to the approved sha256, with `connect-src 'none'` and `sandbox allow-scripts`. The receipt pages for `f2r-e17310fdc613b6398580d58d` and `frr-964482c60392f89c97b48135` each said "Receipt checks out". The public page `atlasclose.com/factory2` served bytes with sha256 `6c63417da8f9…`, equal to the r3 staged page; the r3 acceptance record had listed this as NOT VERIFIED.
- 2:56:57–2:57:00 PM PST: `initialize` and `tools/list` returned server `factory2-mcp r3` with six tools and no approve or confirm tool; receipts `f2r-e17310…`, `f2r-8bc9ae…` and `f2r-7b1043…` each returned CHECKS_OUT, 9 of 9; plain and cache-bypassing fetches of all three live tools matched their approved sha256.
- November 16, 2026, 9:02:40–9:03:39 AM PST: `initialize` and `tools/list` again returned `factory2-mcp r3` with six tools and no approve or confirm tool. All six factory2 receipts cited in this paper (the sample and the author's five) returned CHECKS_OUT, 9 of 9. Plain and cache-bypassing fetches of the sample tool and the author's five tools all matched their approved sha256, and the receipt page for `frr-964482c60392f89c97b48135` said "Receipt checks out". The watch pages still label times "PDT" (6.12).

6. Limitations and threats to validity

1. **Tests prove these checks passed, nothing more.** APPROVED means this exact page passed these written rules and these checks in the locked box. It does not show the page is free of defects, correct for inputs the checks do not cover, or useful. Atlas cannot judge usefulness.
2. **Who wrote the rules.** The rules and checks were written by AtlasClose's build agents. A design describing them was presented to the author before the build, and he gave the go to build (relayed in the deploy record); the deployed rule text (R1–R9) and the template checks were not separately reviewed by him. If a person confirms a weak specification, a weak tool can pass. Over-rigid checks can refuse a good tool.
3. **The checks are visible to the builder (construct validity).** The visitor directions and the watch page publish the inputs and expected outputs, so a page could pass by reproducing those four examples without computing correctly. No hidden or held-out checks were used. APPROVED therefore means it matched four published examples and passed the nine rules. In the scripted runs, the "person's confirmation" was a form post by the test script, and the tried pages were written in advance by the build team.
4. **The tests are not independent.** AtlasClose's AI build agents designed the service and the tests, ran them, assigned every PASS/FAIL/NOT VERIFIED label in Section 5, and drafted this paper for the author. No second party re-scored any row. The APPROVED/REFUSED decision is made by fixed rules and checks, not by an AI, but the evaluation of those decisions in this paper was. Under Article XIV this is evidence, not independent validation. No independent Audit report on this factory exists yet. Earlier versions of the underlying practice factory were reviewed by Audit; neither system described here has been. Audit's review of v1 of this paper checked the public copy, not the system.
5. **A code-holding agent can imitate a browser.** The one-time confirm or approve code is in the watch page's HTML. An agent able to load that page and send form posts with forged headers could imitate the button; the tools never return the code, and no approve tool exists, but this path is not closed and was not tested. In the human-approve factory, r3 deliberately accepted `Origin: null` with a valid code, on the reasoning that a script holding the code could already forge any origin. In "Atlas decides", the same path would let an agent confirm the problem statement on the person's behalf; it would not let it approve software (inferred from the design).
6. **The first real-browser approval failed.** The test harness had set the Origin header itself, which a real browser did not. Simulated clients can pass tests that real clients fail. The watch page's live updating, the r3 page in a real browser and on a phone, and the receipt page's in-browser re-hash remain NOT VERIFIED.

7. **Not every bypass is eliminated.** R5 is a written list, not a proof of absence. The locked box is a small attack surface but JavaScript is allowed, and the pinned engine must be updated when a security issue is found. Receipt verification runs on the server under test, and its own live-byte check is a loopback fetch. Live-state tampering, and a consistent rewrite of bytes, docket and receipt by someone holding the server's signing key, were not tested.
8. **No comparison or control.** We did not run the same tasks through an ungoverned build-and-ship loop, so this paper makes no claim that the design produces better or safer software than one. It shows only that the stated controls behaved as designed in the tested cases.
9. **Model not recorded.** The AI model and version used by the visitor's assistant and by the build agents were not recorded. Nothing here compares the same model with and without these controls.
10. **Cost not measured.** The decision server makes no paid AI calls. The AI agent usage spent building, testing and documenting the demonstration is not reported in this section; it is listed in Section 7. Spend on that work is not measured (usage records give shares, not dollars, and do not split by task).
11. **Rollback history.** Three attempts did not go as planned (the r1 rollback, the r3 pre-check stop, and the delivery-network rewrite in the human-approve factory). Each was caught by a gate or a test and reported; together they show the system is young.
12. **Operational gaps.** The "Atlas decides" service does not yet check the stop register required by Rule Q; turning it off is a manual procedure. The watch and receipt pages show times in UTC-7 labelled "PDT" on November 13, 2026, when Pacific Standard Time (UTC-8) applies. Each shown time is the correct instant, but it reads one hour later than local time. The receipt stores zone-free timestamps, and no stored entry text with "PDT" was found; the fix is in the page display.
13. **Small sample, no inference.** Two tool types (the commission-split template and one calculator built from the author's examples), one sample case per scripted run, five released cases run through the author's own AI assistant (four distinct pages; one with refusals), and no outside visitors, all on one day on one server. The test scripts were revised once after a failure caused by a test bug (kept on the record). These results show the mechanism working in these cases. They are not estimates of how often it works.

7. AI work and estimated equivalent human hours

Measured figures and an estimate, kept apart. The figures in 7.1 and 7.2 come from AtlasClose's accounting records or from the records already cited in this paper, each with its source. Where no record exists, the cell says not measured. The figures in 7.3 are an ESTIMATE of equivalent human professional hours, with a stated method and a low–high range.

The author asked for a comparison between the AI work behind this demonstration and the human labor it would take to produce the same outputs. The two sides are not measured the same way, so they are not subtracted from each other and this page does not say how much time was "saved". AI-side figures are measured from records, each with its source. Human-side figures in 7.3 are an ESTIMATE, with a stated method and a low–high range.

7.1 AI side (measured only)

Measure	Figure	Source
Wall-clock hours of AI agent runs (building, testing, reviewing, drafting)	About 3.4 hours of recorded build/test/drafting helper runs (206 minutes across 17 runs with start and end times: 15 on Nov 13, 2 on Nov 16); several runs, the review offices' work and the coordinating agent's own time not measured.	Accounting records (helper-run start and end times in the coordinating agent's chat transcript index), November 16, 2026
Agent turns and tool calls	Coordinating agent, Nov 13: 49 turns, 106 tool calls, 16 helper runs launched; Nov 16: 3 helper runs, about 12–17 tool calls on this paper. Helper and review-office turns not measured.	Accounting records (same transcript index), November 16, 2026. The Nov 16 tool-call count is classified by keyword and is labelled inferred.
Number of AI agents involved	4 named agents (build, legal, science, audit offices) plus 19 helper-run launches.	Accounting records (review files and helper-launch records), November 16, 2026
Usage or spend on AI agent work	No paid AI calls by the decision server (by design). Spend on AI agent work: not measured (usage records give shares, not dollars, and do not split by task).	Accounting records (design record and usage records), November 16, 2026. See Section 6.10.
Server releases of the "Atlas decides" factory	3 (r1 rolled back, r2, r3)	Deploy and acceptance records (Section 5.3)
Scripted test steps run in the public end-to-end runs of r2 and r3	16 and 20 (final runs)	Acceptance records (Section 5.3)
Review documents on this paper by AtlasClose's Legal, Science and Audit offices (AI agents)	8 to date. Legal: reviews of v1 and of v2.1. Science: review of v1 and rechecks of v2.1 and v2.2. Audit: reviews of v1, v2.1 and v2.2. Later notes on the effort comparison are not counted as reviews of the paper.	Review files of November 13 and 16, 2026
Versions of this paper	6 (v1, v2, v2.1, v2.2, v2.3, v2.4)	Errata at the end of this paper

7.2 The author's own time (human in the loop)

Measure	Figure	Source
Decisions, card answers and messages by the author on this work, with time span	31 messages on November 13, 2026, 11:10 AM–2:58 PM PST (span 3 hours 48 minutes), of which 6 are identifiable as decisions or approvals; no card answers found in the coordinating agent's chat (other offices' cards not searched). Two messages on November 16 about this paper, as relayed. The author's other November 16 messages were about other work and are not counted here.	Accounting records (coordinating-agent chat transcript index), November 16, 2026; the two November 16 messages are as relayed from Legal's record of the author's request
Server-recorded span of the author's "Atlas decides" cases, November 13, 2026	1:58:53–2:32:22 PM PST (about 34 minutes); 7 cases opened, 5 confirmed and released	Server records (Section 5.6). This is when the cases ran, not a measure of the author's attention.
Author's time reading and approving this paper	not measured	No record

7.3 Estimated equivalent human hours (ESTIMATE)

Method. List the outputs: the 'Atlas decides' factory and the human-approve factory and their tests; four distinct released tool pages (five releases, one of them byte-identical to an earlier page); the review documents; and this paper. Multiply

each by typical professional hours per task, with each per-task figure, a count, and its public source given; report the sum as a low–high range. The result is the estimated time for people to produce comparable outputs. It says nothing about whether the outputs are of comparable quality.

Task	Count	Hours per task (low–high)	Subtotal (low–high)	Public source
Design, build, test and deploy a small rule-checking web service	2 (the "Atlas decides" factory and the human-approve factory, including their tests)	20–160	40–320	Low: 20–60 h for a solo-developer MVP (weak source: an automated estimator page). High: "MVP / Simple App … 4–8 weeks" (4 weeks × 40 h = 160 h, the low end of that range)
Build one small single-page calculator or tool page	4 distinct pages (5 releases, 1 byte-identical)	1–12	5–60 (Accounting applied the rate to five releases, including the byte-identical page)	1–2 h each, a public note on building small calculators; "Simple (5 fields) 8–12 hours", a public calculator-build guide
Internal policy, method and audit-style reviews of a paper	6 (the reviews on record when the estimate was prepared; the two later reviews counted in 7.1 are not in this total)	5–6	30–36	Median 5 h per review, a 2018 global peer-review survey; 6 h, a public 2019 citation of a science-communication group's figure
Write a 15–20 page technical white paper	1 (six versions counted as one paper)	24–50	24–50	"typical white paper takes between 24 and 50 hours" (2007 survey of 600+ writers), a public white-paper writing guide
Total estimated equivalent human hours			**99–466 h**	Sum of the subtotals

The estimate uses the six reviews on record when the accounting figures were prepared: Legal's reviews of v1 and of v2.1; Science's review of v1 and recheck of v2.1; Audit's reviews of v1 and of v2.1. The two later reviews counted in 7.1 (Science's recheck of v2.2 and Audit's review of v2.2) are not included in the 99–466 hour total. The reviews counted here were done by AI agents inside AtlasClose's own governance system. They are not, and are not claimed to be equivalent to, the work of a lawyer, a court, an auditor or an independent scientist. Atlas law is not a court of law and is not legal advice (Section 4). The five released tool pages were submitted through the author's own AI assistant (the author's account; see 5.6), so they have no AtlasClose AI-side hours in 7.1; they are included in this human-hours estimate. Accounting noted that dropping them would leave 94–406 hours. Revisions after the first version are not costed separately.

8. Future work

- Independent validation: an Audit-run test plan, executed by a party that did not build the service, including attempts to falsify H1–H5.
- Held-out checks that the builder does not see, so that APPROVED tests computation rather than agreement with published examples.
- Integrate the stop register (Rule Q) so the Magistrate can stop the factory and the runner fails closed.
- Exercise the browser-side receipt check, the watch page and the confirm button on real browsers and phones, and record the headers they actually send.

- Collect completed runs by outside visitors' own AI assistants, record the model and version, and report them with the same PASS/FAIL/NOT VERIFIED discipline; run a comparison against an ungoverned build-and-ship loop.
- Fix the time-zone display on the watch and receipt pages to use the local Pacific zone, and add a test for it.
- Reduce the confirm-code imitation path (for example, by binding confirmation more tightly to a real browser interaction), and test live-state tamper detection under a controlled procedure.
- Optionally, a governed AI reviewer that writes a reasoned opinion on usefulness, on the record but not the decision; the rules and tests would still decide.
- Connect monitors to proceedings so that a failed receipt or byte mismatch opens a corrective matter automatically: a step toward Level 6.

References

Background literature

1. K. R. Popper, *The Logic of Scientific Discovery*, 1959. Falsifiability as the criterion for a testable claim (used for H1–H5).
2. S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *Journal of Cryptology* 3(2), 1991. Hash-linked records.
3. NIST, FIPS 180-4, *Secure Hash Standard*, 2015. SHA-256.
4. W3C, *Content Security Policy Level 3* (connect-src, sandbox).
5. WHATWG, *Fetch Standard*; and W3C, *Referrer Policy*. When a request's Origin is serialized as "null" and when a Referer is sent.
6. IETF RFC 9111, *HTTP Caching* (no-store, no-transform).

Primary records (AtlasClose, November 13, 2026, unless noted)

7. Design: "AtlasClose next factory — design (not a build)", with mock-up page.
8. Server records of the author's "Atlas decides" cases (watch pages, receipts and released bytes), captured November 13, 2026, 2:44 and 2:57 PM PST.
9. Deploy records: "Atlas decides" r2 (1:41 PM PST) and r3 (2:17 PM PST); human-approve factory r1/r2 and r3 (approve-button fix).
10. Acceptance records: "Atlas decides" r2 and r3; human-approve factory r1/r2.
11. Visitor directions v3, "Atlas decides: try it with your own AI".
12. Legal review 519-A, Science review of Sections 5–6, and Audit public-copy review of v1 of this paper (November 13, 2026).
13. Atlas Law Book (compiled from the public docket; Constitution of Atlas, effective August 27, 2026) and its addendum r1 (November 12, 2026); orders O-LANDMARK-MAGISTRATE-STOP, O-CLOSE-OF-RECORD, O-CORRECTIONS-STANDING, O-SPACE-WHEEL-STANDING, O-GC-002; Legal ruling on practice-company deadlines (October 26, 2026); AtlasClose design lessons (October 20, 2026).

Appendix A. Public links and identifiers

Item	Value
Endpoint ("Atlas decides")	`https://demo.atlasclose.com/factory2/mcp`
Public page	`https://atlasclose.com/factory2`
Human-approve factory page	`https://atlasclose.com/factory`
Sample case	`fc2-3b778315d2b2bac3`
Sample watch page	`https://demo.atlasclose.com/factory2/mcp?watch=w295ecc9ad197aa09d55a45089208e4252`
Sample live tool	`https://demo.atlasclose.com/factory2/mcp?tool=t29adcd54e73f9d52770e0`

Sample receipt	`https://demo.atlasclose.com/factory2/mcp?receipt=f2r-e17310fdc613b6398580d58d`
	`receipt sha256 cce3589743a4f199f4311b66eb652a8b291520e0b320d6f70062ac328e55fd0c`
	`re-checked CHECKS_OUT, Nov 13, 2026, 2:34:29 PM and 2:56:57–2:57:00 PM PST, and Nov 16, 2026, 9:02:40–9:03:39 AM PST`

Approved tool bytes (try 2)	`0494f78a5be042ae0774e3956460fa8e02ec697abf94b1bd04fda83760f09e64`
Spec / checks	`7dc47826109f70dfb351c4ccff800305c38fee9c6b2b8d3a338c5acefb3602d3` `9856183616d6059c29388df3add4fd201397f3da20202445e33cab1845fc8a48`
Locked-box engine / harness	`f4e8a3b4a4197cf354c35e81622fa36d7f1e62dcd03d8ffb504ea3968cd760c2` `fc2f1b275b9b7276c7ea2f5b8cafbb4e15e7460ff42accaecff25cc1878ead36`

Author's "Atlas decides" receipts	`f2r-8bc9ae324fab52244a9be908 (case fc2-68bb528bddc102f9)`
	`f2r-7b104361c0ef43b7d84e7b79 (case fc2-918d53344d6de64c)`
	`f2r-d2333f6b693af2325fa35ae4 (case fc2-a9c23db8bc877729)`
	`f2r-fa6f06104fac77c16ab18c3c (case fc2-a946a712795d0689)`
	`calculator case: receipt available on request`
	`all five CHECKS_OUT, Nov 16, 2026, 9:02:40–9:03:39 AM PST`

Rules version	`atlas-tool-rules r1`
Author's run receipt (human-approve)	`https://demo.atlasclose.com/factory/mcp?receipt=frr-964482c60392f89c97b48135` `page showed "Receipt checks out", Nov 13, 2026, 2:34:40 PM PST, and Nov 16, 2026, 9:02:40–9:03:39 AM PST`

To re-check the sample yourself: add the endpoint as a remote tool server in your AI assistant and ask it to call `atlas.factory2_verify_receipt` with the receipt id above, or open the receipt link in a browser. A result of anything other than "Receipt checks out" on a later date would be a finding worth reporting.

Errata and changes from v1

Version 1 (November 13, 2026) is superseded by this version; v1 is kept unchanged. Changes, after reviews by Legal, Science and Audit:

- Added that Atlas law is an AI institution's internal governance, not a court of law and not legal advice (status box, Section 4).
- The re-check at the time of writing is now described as a same-team repeat, not "independent re-verification"; the cited version capture is a fresh one at 2:56:57 PM PST.
- Corrected four overstatements: the limit on ordering Audit's verdict (it binds the Magistrate); the deploy authorizations (relayed; the r3 fix followed a problem report, not a yes); the record of first-hand adoptions (shown for orders adopted since October 2026); and the conditions for lifting a Magistrate stop. The Art. XIII §3 quote now reads in full.

- O-CORRECTIONS-STANDING: v1 presented its adoption as disputed; it was adopted on October 20, 2026. The space wheel only reports today; its clearing step is not yet built or switched on.
- Time-zone display: v1 suggested a docket correction; the error is in the page display only (times are correct instants labelled PDT instead of PST).
- Section 5 now states that the worked-example pages were written in advance by the build team and confirmed by the test script; narrows H1; describes the r2/r3 repeat as a re-run, not a replication; adds the author's recorded runs through his own AI assistant, and his assistant's qualitative observations, attributed to it.
- Section 6 adds construct validity (published checks, no held-out checks), who wrote the rules, same-party evaluation, no control, model and cost not recorded, signing-key tamper, and the true sample size with no statistical inference.

Changes in v2.1 (November 16, 2026)

- Section 5.6 now reports all five of the author's released "Atlas decides" cases backed by server records, including one case refused twice on R8 before an approved try, and one calculator built from the author's own worked examples. v2 reported two cases and listed the other three as not verified; the records copied at 3:02 PM PST on November 13 back them.
- The abstract, H2, H3, H5, Section 6 (sample size) and Appendix A are updated to match; the assistant's qualitative observations about a calculator case and a refused case are now checked against the records.
- Added a same-team re-check on November 16, 2026, 9:02:40–9:03:39 AM PST: every cited receipt still checks out and every cited live tool still serves its approved bytes.
- v2 is kept unchanged.

Changes in v2.2 (November 16, 2026)

- Section 5.3: v2.1 said the tables were updated, but the author-case row still showed the v2 count of two. It is replaced by two rows (three r2 cases, two r3 cases), counted as released and refused tries, not PASS/FAIL.
- H5 now says which cases had checks fixed by the template and that the calculator's checks were written by Atlas from the author's examples before the build.
- Section 6.2 no longer says the author reviewed the rules; he gave the go to build after a design was presented, and did not separately review the deployed rule text or template checks.
- The abstract, Section 5.6 and Section 6.13 note four distinct pages across five releases and that the server records submissions, not authorship; Section 5.6 states how the author's cases were separated from test cases and that the refused page's intent is not recorded.
- The calculator case's receipt is no longer printed (available on request), because its public receipt shows a personal title. The status box now repeats that the Magistrate, Legal and Audit are AI agents.
- Section 3.10(6) and Section 5.6 now mark the author's 2:04 PM report as relayed. The November 16 re-check time is given as 9:02:40–9:03:39 AM PST throughout.
- New Section 8 compares measured AI work with estimated equivalent human hours, at the author's request. Its accounting figures are pending and marked as such.
- v2.1 is kept unchanged.

Changes in v2.3 (November 16, 2026)

- The status box now excepts the estimates in Section 7 from the "every quantitative statement" line, and names review files and accounting records among the sources. Section 7's human-hours figures are estimates.
- Section 6.10 points to Section 7 for AI-agent usage; spend on that work is not measured.
- The effort comparison is moved before Future work and numbered Section 7; Future work is Section 8.
- Section 7 figures are filled from Accounting records of November 16, 2026; cells without a record say not measured. The human-hours range 99–466 is an estimate. The task table lists two services, four distinct tool pages (five releases, one byte-identical), the review documents, and this paper, with a count column.
- The sentence "Nothing on this page is a guess" is dropped.
- Two Appendix A re-check stamps that had said only "9:02 AM PST" now read 9:02:40–9:03:39 AM PST.
- The calculator case identifier is withheld from the public copy (receipt still available on request), because a public lookup of that identifier might lead to a personal title.
- v2.2 is kept unchanged.

Changes in v2.4 (November 16, 2026)

- Section 7.3 now says the five released tool pages were submitted through the author's own AI assistant (the author's account; see 5.6), not that they were written by it.
- The public-source column in Section 7.3 keeps the quoted figures, years and generic descriptions, and no longer prints outside brand, site or handle names. The exact sources remain in the private claims ledger.
- v2.3 is kept unchanged.