WIW, Reading Manual

Read-only · interpretation, not procedure (D-M1) · Block L, sub-sprint L.5

READING MANUAL

Generated from the same source as this page, so the two can never disagree.

canonically rigorous partial proof of concept, not a Tier 1 production system

NOT DEPLOYABLE (§C.16.2)

First reading, what you are looking at

1. You are looking at a read-only rendering layer. It shows state that the WIW system already produced; it never decides, never validates and never writes. Every response it reads carries `authoritative_for_wiw: false`.
2. It reads the public HTTP boundary, and only over GET. The routes it may read are declared as data in the manifest, and a test crosses each one against the real server, so a route named here that did not exist would fail the suite, not fail silently on screen.
3. The order of the sections is itself a claim. The four limits and the declared gaps come BEFORE the content: what the system does not prove is the first thing after the non-authority banner, so that no reader infers completeness from the layout.
4. Three words carry most of the reading. MET: evaluated and it holds. UNMET: evaluated and it does not hold. UNDETERMINABLE: not evaluated, because the producer does not exist, and it blocks exactly as a failure does.
5. The banner at the top is live: it is read from §C.16.2 on every load, not typed into the page. While it says NOT DEPLOYABLE, this is a canonically rigorous partial proof of concept demonstrating the thesis, not a Tier 1 production system.
6. This manual explains what a screen MEANS. It is not an operations runbook: it carries no operational step of any kind, and no procedure to carry out. The canonical frame is §D.56.1 — auditors inspect; they do not operate.

Panel by panel, what it proves, and what it does NOT

[surface]

[ui.manual.proves]	That the surface enumerates itself: every route the server answers is in the registry, and the registry is what the router is built from.
[ui.manual.does_not_prove]	That the surface is authorised for a given reader, §D.59.5 returns UNDETERMINABLE for every route, because RoleMayObserve and ScopeLegal have no producer.

[j1]

[ui.manual.proves]	That the identity MODEL of §8.1 is transcribed: classes, states, forbidden transitions and the receipt schema exist and are checkable.
[ui.manual.does_not_prove]	That any identity exists. No handler populates the identity store, so there is no WIW-ID to look up: the panel says so in `per_identity_lookup`.

[j2]

[ui.manual.proves]	That the authentication flow of §8.1.7 runs as law: the stages, the failure codes and the two temporal laws are executable.
[ui.manual.does_not_prove]	That anyone has authenticated. The executor exists and the producer does not; with no clock and no store the verdict is UNDETERMINABLE.

[j3]

[ui.manual.proves]	That authorization is a function of six named arguments (§8.1.8), and that each argument has a canonical producer named.
[ui.manual.does_not_prove]	That authorization is being computed. It also proves what AuthZ is NOT: not a PART 4 FinalDecision, not an EnforcementDirective.

[j4]

[ui.manual.proves]	That the gateway's authority boundary and the ten invariants INV-5D.1–10 are transcribed, with the ingress state machine.
[ui.manual.does_not_prove]	That execution ingress is running. The OA0 surface is NOT ingress: a declared subject mapping, held by the zero-mutation guard.

[j5]

[ui.manual.proves]	That the six dashboard families of §C.12.21 are declared, each with its indicators and its consistency model.
[ui.manual.does_not_prove]	That any number on them is live. There is no measurement layer wired to the runtime; a fabricated number would be the optimistic value §10.9 forbids.

[j6]

[ui.manual.proves]	That §8.12.H1.7 has a fixed list of isolation requirements, and the measured state of each one.
[ui.manual.does_not_prove]	That tenants are isolated in production terms: the partition axes that would carry it are named as absent.

[j7]

[ui.manual.proves]	That promotion has canonical stages and that PromotionAllowed is a conjunction of eight (§C.13.2), each with its producer named.

[ui.manual.does_not_prove]	That promotion could happen. All eight conjuncts are Tier C: none has a verdict producer, so the gate is UNDETERMINABLE by construction.

[j8]

[ui.manual.proves]	That disclosure and redaction have canonical classes and targets, and that provenance has explicit exposure prohibitions.
[ui.manual.does_not_prove]	That redaction is being applied to live data. This surface serves law text and schema inventory, never store records, measured, not asserted.

[j9]

[ui.manual.proves]	That SLO classes separate SERVICE health from EVIDENCE health, and that AuditClosure is a declared conjunction.
[ui.manual.does_not_prove]	That the system is healthy. No SLO is being measured; the panel shows the model and names the missing measurement layer.

[j10]

[ui.manual.proves]	That every persisted artifact family is inventoried by name, domain, contract_id and retention class: the audit trail of §C.12.16.
[ui.manual.does_not_prove]	It never proves anything about CONTENT: no record is ever served. The count is the server's, not a length the screen computed.

[k0]

[ui.manual.proves]	That ANNEX D never defines authority: the four authoritative axes it may never carry are declared, with the source of each domain's authority.
[ui.manual.does_not_prove]	It is not itself an authority statement: it is the boundary that keeps the operations layer from becoming one (#001/#426).

[k1_roles]

[ui.manual.proves]	That the nine RoleClass values of §D.50.3 are canonical, and which of them have a playbook.
[ui.manual.does_not_prove]	That any role is assigned to anyone. The registry is a model; there is no authenticated actor behind it.

[k1_actions]

[ui.manual.proves]	That the six action classes map to six approval classes with a declared floor, and that policy may tighten but never loosen it.
[ui.manual.does_not_prove]	That any approval has been granted. The mapping is the law; the approvals themselves have no producer.

[k1_envelope]

[ui.manual.proves]	That role x action-class is decided cell by cell by §D.51.3–§D.57.3, not by convention.
[ui.manual.does_not_prove]	That the envelope is being enforced at runtime, nothing consumes it on the execution path yet.

[k2]

[ui.manual.proves]	That HumanActionLegal (§D.50.5) is a conjunction of nine, and that the missing producer of each is named.
[ui.manual.does_not_prove]	That a human action would be legal. With producers missing, the predicate blocks, and blocking is the correct answer, not a defect.

[k3]

[ui.manual.proves]	That approval has a canonical state machine and five execution-time revalidations (§D.59.5A).
[ui.manual.does_not_prove]	That any approval flow has run. The machine is transcribed; no instance of it exists.

[k4]

[ui.manual.proves]	That the eight ConsoleClass values exist, mapped to roles, with the declared capabilities of each console.
[ui.manual.does_not_prove]	That every capability is served. Three of the seven OBSERVABILITY_CONSOLE capabilities are named as NOT served, by name.

[k5]

[ui.manual.proves]	That CommandVisible has four conjuncts and CommandExecutable five, with the provenance and tier of each.
[ui.manual.does_not_prove]	That any command is visible or executable, two of the four have no producer, so the predicate returns UNDETERMINABLE and blocks.

[k6_dual]

[ui.manual.proves]	That six triggers require dual control (§D.59.8), and that no surface may bypass them.
[ui.manual.does_not_prove]	That dual control is enforced anywhere today, there is no command surface for it to guard.

[k6_quorum]

[ui.manual.proves]	That quorum classes, timeout windows and deadlock resolutions are canonical, and that quorum authority is ANNEX E, not ANNEX D.
[ui.manual.does_not_prove]	That a quorum has ever been formed. The handoff is declared; the authority lives elsewhere.

[k6_viewers]

[ui.manual.proves]	That viewer classes are separated by §D.59.10, and which consumers were measured OUTSIDE that separation.
[ui.manual.does_not_prove]	That the separation is enforced: it is measured and declared, not applied by a running gate.

[k7]

[ui.manual.proves]	That §D.58.9 names five console receipts, and it shows the measured gap between the NAME and the schema.
[ui.manual.does_not_prove]	That the five receipts exist as artifacts: they are named by the canon and have no field schema and no producer.

[k8_audit]

[ui.manual.proves]	That §D.59.6 lists eight audit-viewer requirements, with the provenance and the met/unmet verdict of each.
[ui.manual.does_not_prove]	That an audit viewer is operating. The requirements are the subject; the viewer is not built.

[k8_plane]

[ui.manual.proves]	That §D.60 closes the human control plane as a conjunction, and it publishes the status of each conjunct.
[ui.manual.does_not_prove]	That the plane is closed. The panel shows the consequences the canon declares IF the closure is false, instead of hiding them.

[k8_inv]

[ui.manual.proves]	That ANNEX D has invariants which cut across sections, each carried with its section of origin.
[ui.manual.does_not_prove]	That they are enforced by a running gate, they are transcribed law, sealed as an ANNEX D artifact.

[c16]

[ui.manual.proves]	[!] That this system measures its own deployability under §C.16.2 and publishes the answer: NOT DEPLOYABLE, with the reason of each conjunct.
[ui.manual.does_not_prove]	It does NOT prove the system is broken. NOT DEPLOYABLE is the measurement that 'canonical proof of concept proving the thesis' and 'PRODUCTION-GRADE VALID' are different states, and only §C.16.2 is evaluated: §E.22.9 and §B.3.10 are declared and inert.

[e22]

[ui.manual.proves]	[!] That the public text is bound by a MECHANISM and not by discipline: the forbidden-claim list lives in `src/`, is served over this route, and a guard scans every site string against it in both languages (§10.14).
[ui.manual.does_not_prove]	That the text is free of overstatement in general. The list is ours and finite (§10.12): it catches the claims it names, and a claim nobody thought to name passes. It also evaluates neither §E.22.11 nor §E.15.2 — those are declared and inert, with the absent producer named.

[memory]

[ui.manual.proves]	That the system HOLDS memory records and graph nodes, that each one is sealed, and that a deterministic probe (§3.4.8) says whether the bytes on disk still match the write journal.
[ui.manual.does_not_prove]	[!] That the order means anything. The ordering is alphabetical by `giml_id`, declared, `GIMLScore` is undeterminable (O-LIM-05), so the list ORDERS, it does not RANK. And nothing here is authoritative: WIW-MOL, the only authority of memory (#033), does not exist in this system (O-LIM-10).

[command_receipts]

[ui.manual.proves]	That every button press left a receipt on disk, accepted AND refused, sealed, hash-chained, and still there after the process stops and comes back. §D.58.10 requires privileged actions to stay replay-auditable from console artifacts, and §D.59.5A requires the FAILED attempt to remain receipted.
[ui.manual.does_not_prove]	[!] That anything ran. `EXECUTABLE` means the REQUEST was lawful and produced a receipt, never that the target action happened. No function anywhere in the code carries out a target action (V-LIM-10), and that absence is deliberate: #390/§D.0.5 — ANNEX D never defines authority.

[approval_queue]

[ui.manual.proves]	That an approval has STATE and HISTORY: one person asks, the grant sits in the queue naming which binding of §D.59.4 it waits on, and another person binds themselves as approver. Each step is a new revision on disk, chained by hash, nothing is overwritten. And it proves the law bites: with one declared holder, an AP2 grant stops at UNDER_REVIEW because §D.59.8 requires two distinct people.
[ui.manual.does_not_prove]	[!] That an approved grant lets anything happen. `APPROVED` means a supervisor bound themselves to the request, never that the target action ran, and nothing ever reaches `EXECUTED`, because the five execution-time revalidations of §D.59.5A have no producer at all. A stored approval also authorises nothing on its own: §D.59.5A says a previously granted approval is not sufficient by itself, so a later request presenting none is refused even when the queue shows APPROVED.

[learning_pipeline]

[ui.manual.proves]	That local data really does become an abstract signal, and that the raw payload is REMOVED: the removal is verified by the post-condition of phi (§3.2.4), not asserted. It also proves that the signal STOPS: the export gate (VCB) has five conjunctions and only one of them has a producer today, so the verdict is UNDETERMINABLE and UNDETERMINABLE bars exactly like ILLEGAL.
[ui.manual.does_not_prove]	It does NOT prove that the signal is safe to export. Reconstructability and LeakageRisk have no producer, they belong to block U, and nothing here approximates them. It does not prove the scores are right either: three of the four LVS inputs and three of the five admission letters have no producer (blocks W and X). The letter mapping of §1.4.5 onto the artifact fields is ANCHORED INTERPRETATION (§10.9): declared, and contestable.

[memory_authority]

[ui.manual.proves]	That the system now declares, in running code, which organ may create authoritative memory, and which five kinds of state lawfully live outside it. It runs the five conjunctions of the authoritative-write gate (§1.H4) and the five of the promotion law (§1.3.11), and it refuses with the clause named. It also proves the two gates of §0.9A are taken as two separate inputs: passing the learning gate is not passing the memory gate.
[ui.manual.does_not_prove]	[!] It does NOT prove there is authoritative memory in this system. There is none. No record has crossed both gates: the authoritative-write gate has two conjunctions with no producer, and the promotion score of §1.3.11 has six coefficients with no value and six inputs with no NAME. It also does NOT close O-LIM-06 or O-LIM-10, which the block's own META promised, measured and refuted.

[containment]

[ui.manual.proves]	That once someone decides a contribution was poisoned, the system knows exactly how far to reach and what to block. Five reaches, from a single execution path to the whole network; five domains blocked for every contained subject, never a subset; and each block cites the clause that requires it. It also proves that reach is bounded both ways: nothing outside the declared scope is touched, and nothing inside it is missed. And nothing is erased: the lineage travels with every containment record.
[ui.manual.does_not_prove]	[!] It does NOT prove the system detects poisoning. It does not. All three detectors exist and return UNDETERMINABLE, because part of each formula's weight depends on quantities nobody measures yet (Bloco X). Every containment you see here began with a person deciding, or with an outside warning. Containment, not immunity. Two of the five domains, propagation and memory, also block paths that were already closed for other reasons.

[reconstruction]

[ui.manual.proves]	That the two quantities the export gate needs now answer. Reconstruction is scored over the five forms the corpus names (§9.1.3.H2 and §C.15.2, converging independently), weighted by the grades the founder set in D-U1, in the fixed-point form §5.E.12 prescribes. Every refusal names WHICH form caused it, HOW FAR past the line it fell, and WHICH VERSION decided \u2014 and the version seal means recalibration against real data is a new version, with the old verdict still reproducible byte for byte.
[ui.manual.does_not_prove]	[!] It does NOT prove the signal can cross. It cannot. The gate has five conditions and goes from one to three answered: the fourth is Bloco W, and the fifth depends on a layer in the next version of the product, outside the whole queue. It also does not prove the numbers are right: they are the founder's, not the corpus's, and they were set without a single real client signal. And the conservative regime refuses useful signal by design \u2014 the distribution shows how much, and the near-misses are the first to revisit.

[peers]

[ui.manual.proves]	That two separate machines exchange a real envelope over a real network, and that the receiving one verifies with its OWN criteria and refuses what does not match, naming the clause. It also proves the law that keeps the network in its place: transport is subordinate to execution, so a timeout says the MESSAGE did not arrive and never that the execution was invalid. And it makes a measured defect impossible rather than forbidden: node isolation was an environment-variable swap, declared non-reentrant; two processes do not share an environment.
[ui.manual.does_not_prove]	[!] It does NOT prove that one node sent intelligence and another LEARNED from it. A signal now crosses on its own and the receiving node ADMITS it, every condition of its admission law concludes, and nothing downstream takes an admitted signal and learns from it: the canonical path runs governance and treaty first, and the treaty layer can only produce BLOCK. It also does NOT prove the receiver knows WHO sent: it checks WHAT it received, never FROM WHOM. That is attestation, that is a secure chip, and it is outside every block. And the rule the receiver applies is a DEMONSTRATION rule: a real node writes its own.

[travessia]

[ui.manual.proves]	That generation works, that the export gate REFUSES naming the clause, and that RECEPTION works: a node ingests a signal into its local adapter (§3.11.H1.1 c.9) and answers differently, with the base model provably frozen. Both runs replay bit-exact.

[ui.manual.does_not_prove]	[!] That the network propagates. The joint between THESE TWO RUNS is A PERSON: that is a fact about this panel, and it stays true. The gate refused here. [!] Elsewhere a signal does cross on its own, is admitted, AND is learned from: a node answering a sealed exam went from eight right out of twelve to twelve out of twelve after receiving, in both directions, and both learners were run on it side by side and scored identically. What THESE TWO RUNS do not show is that; and what stays closed is the CANONICAL learning path, influence constraints (§3.3.XA, one occurrence in the corpus) and MatchConditions (§11.7, one occurrence), neither of them our code. So the learning that was measured happened on the demonstration path, not on the canonical one. The object that crosses in these runs is DECLARED, not the WIW wire format. And no node attests to any other.

[onboarding]

[ui.manual.proves]	Proves that a node joining leaves a TRAIL: four chained receipts that outlive the process which wrote them, and the activation law of §A.10.1 concluding in full, with not one undeterminable condition: the first closure law in this project that answers in full.
[ui.manual.does_not_prove]	Does NOT prove the node received intelligence. Nothing crossed: the inherited content enters UNDETERMINABLE in both fields that would carry it, and the missing producer is named. Nor does it prove the six checks of §A.9 — two pass, four are ABSENT, and absent bars exactly like failed.

[replay_completeness]

[ui.manual.proves]	Proves that every decision now has a VERSIONED subject anchored in lineage, and that re-execution still reproduces the same seal. And proves the system can say WHY re-execution is not authoritative, citing the right law.
[ui.manual.does_not_prove]	Does NOT prove re-execution is authoritative. It is not: the PART 8 law requires equivalence across six context domains, and two of them do not exist in this system. Nor is there cryptographic proof: the corpus itself says proof is optional at this stage.

[fecho]

[ui.manual.proves]	That four MEASUREMENT defects were found by measurement, not chosen to fill a plan, and corrected; that the numbers this system publishes about itself now come from PREDICATES evaluated at check time rather than from documents that age in pairs; and that a limitation which stopped being true would now be caught by a test.

[ui.manual.does_not_prove]	It does NOT prove this is a deployable product: the system declares itself NOT DEPLOYABLE under the corpus own SSC.16.2, and the corpus REQUIRES every public artefact to say so. It does NOT prove pillar progress: no block has moved one since the pillars were first measured, this one included, and the reason is measured across four families, three of which are not software. The running count is kept in the record, not written here. It closes NO limitation: they all remain. And it closes none of the four SS7.4 limits, two were CONVERTED by earlier blocks, and converting is honest while claiming closure would be a lie.

[participation]

[ui.manual.proves]	That the terms of joining the network are WRITTEN, versioned and served from the system, five parts authored by the founder, one closed exception with a term and a record, and the tension with D-VCB2 stated by the author instead of discovered by the reader.
[ui.manual.does_not_prove]	[!] That the network applies them. The law decides who is in default and nothing then delivers less to that participant, because nothing delivers anything to any participant yet (REDE-LIM-01). And the three terms, entry period, silence window, maximum suspension, are the founder's numbers and are not written, so every evaluation returns UNDETERMINABLE (REDE-LIM-02).