Dweve
Weaving Intelligence

ENGLISH SOURCE EDITION · JULY 2026

FOUNDATION GUIDE · F04

The Dweve platform handbook

How the Fabric interface, cognition, agents, knowledge, proof, compute, and deployment postures work as one inspectable system.

WORKSPACE COGNITION AGENTS KNOWLEDGE PROOF DEPLOYMENT

Read this when you need the whole platform explained, not a product list, and not a slogan deck.

F04

CONTENTS

One platform, explained as a working system

The handbook follows the order in which an evaluator needs to understand Dweve: the operating problem, the responsibilities, a real request, the proof, the deployment boundary, and the route into production.

PRIMARY READERS	READING DEPTH	DECISION THIS SUPPORTS
Cross-functional evaluation teams	Professional	A complete reader journey from the first question to a governed result, its evidence packet, and the operating model behind it.

READING RULE

When the website uses a strong performance, availability, security, or compliance claim, this handbook treats it as a claim to verify in the buyer's proof environment. The website sources is named; the proof still has to be run.

SOURCE DISCIPLINE

Dweve-specific facts are taken directly from the English website text audited 2026-07-20; web-navigation wording is humanised for print. Evaluation guidance is editorial, and scenarios and derived visuals are illustrative. Unsupported synthesis is replaced by canonical copy or omitted.

READ THIS FIRST

What this handbook is for

It is the bridge between the website's many detailed product pages and the concrete questions an evaluation team has to answer together.

A licensed team may operate Core on organisation-controlled clusters or workstations. A regulated programme may require a fully isolated estate. A product may move part of its inference to the edge, while a browser application may need a local execution target. Core keeps those licensed operating postures inside one machine without claiming that every backend supports every cell. Core is not a managed-service product.

This handbook connects those lenses. It describes what enters the platform, which responsibility belongs to which product, where people approve or reject work, what record remains, and what changes when the same workload moves from managed Fabric on Dweve's public Mesh to infrastructure the customer operates.

It is deliberately not a substitute for technical due diligence. Architecture claims still need interfaces, source or binaries, benchmark harnesses, security evidence, and a proof on the customer's own workflow. The handbook's job is to make that diligence coherent: readers should know which question belongs at which boundary and what a convincing answer would contain.

PEOPLE

What do I use?

Fabric is the workspace. The underlying products remain visible when their evidence or controls matter.

OWNERS

What changes operationally?

The adoption and deployment posture determines who runs infrastructure, holds keys, moves updates, and carries support duties.

ARCHITECTS

Where are the boundaries?

Products meet through contracts and dependencies point downward. A surface should not rewrite the foundation beneath it.

REVIEWERS

What can I inspect?

The result is accompanied by sources, rules or policies, ownership state, and a replay or verification path where the website claims one.

PRIMARY TEST

After reading, an evaluation team should be able to draw its own workload on the platform map, remove every product it does not need, mark every human gate, and name the evidence required before production.

VOCABULARY

Five words that must not be blurred together

Most confusion starts when a product, a component, a layer, a deployment, and a commercial option are treated as synonyms.

Product

A product owns an operating responsibility: Fabric owns the workspace; Loom owns cognition; Nexus owns organised agent work; Spindle owns the knowledge-processing lifecycle; Aura owns the governed development loop; Mesh owns distributed capacity and its work lifecycle; Charter owns the organisation's portable operating record; Core owns the binary AI framework; Kera owns the language and compiler path.

Layer

A layer is an architectural responsibility in the stack. A product can touch more than one layer, while a layer can be implemented through products and open foundations. The website's layer map is a dependency model, not a pricing catalogue.

Foundation

An open foundation is an inspectable component used to implement a particular job such as parsing, retrieval, policy evaluation, numeric behaviour, event provenance, execution evidence, or proof verification. Source availability and release status must be checked component by component.

Entry door

An entry door is how a customer begins: use the Fabric workspace, build through an API, or license the platform. It changes the buyer's operating responsibility. It does not create a separate architecture.

Deployment posture

A posture states where the workload runs and who operates it: managed through Fabric and the public Dweve Mesh within the declared service perimeter, licensed on customer-controlled infrastructure, or fully disconnected. A posture is not a product tier and it is not proof of regulatory suitability by itself.

TERM	ANSWERS	DOES NOT ANSWER
Product	Who owns the job?	Where it runs
Layer	Where responsibility sits?	What to buy
Foundation	What is inspectable?	Commercial rights
Door	How do we start?	Who hosts it
Posture	Who operates and where?	Which workflow fits

The distinctions are editorially strict because they are operationally different decisions.

EVALUATION DISCIPLINE

How to read strong claims

The website is the canonical source for this edition. Canonical does not mean a buyer should accept every statement without reproducing or scoping it.

A qualitative architectural statement, such as downward dependencies, a typed boundary, or three deployment postures, can be examined through system maps, manifests, interfaces, and an installed environment. A measured statement, such as throughput, latency, recall, operation coverage, energy use, or activated experts, needs the exact hardware, corpus, method, version, and comparison boundary that produced it.

A security or assurance statement needs a threat boundary and evidence. A commercial statement needs currency, cadence, conditions, included rights, and responsibility. A compliance statement needs particular care: support for record-keeping or human oversight is not a universal legal conclusion, and “aligned” is not the same as “certified”.

This handbook therefore separates mechanism from acceptance. It explains the architecture the site claims. The adoption chapters then convert those claims into tests a customer can run.

SOURCE

Locate and classify

Point Spindle at your sources. Get a graph of Canonical atoms with six-dimensional quality, 47-type bias reports, and forward and backward lineage on every fact.

Can another reader find the exact source?

SCOPE

Keep the boundary attached

Retain workload, hardware, version, method, conditions, exclusions, and non-SLA wording.

Would the statement survive outside its original context?

PROOF

Test and record

Translate the statement into an observable customer result and retain inputs, raw output, reviewer, exception, and decision.

Can an independent reviewer repeat the conclusion?

A strong claim becomes useful when source, scope, and proof stay together.

KNOWN SOURCE ISSUE

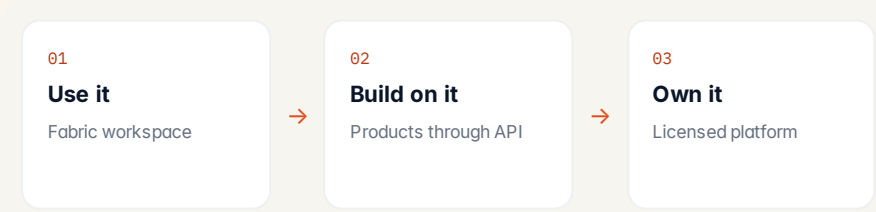
Some listed shapes are currently parity cases or do not yet show a Kera lead. They remain visible in the benchmark report rather than being removed from it. A matrix by operation family and size makes the wins, the parity cells, and the pending-analysis cells all legible at once.

01

THE PLATFORM MODEL

One system, three practical ways in

The first decision is not which box to buy. It is how the organisation wants to use, integrate, or operate the platform, and which responsibilities it is prepared to take on.



Different entry responsibility; shared product boundaries and substrate.

USE IT

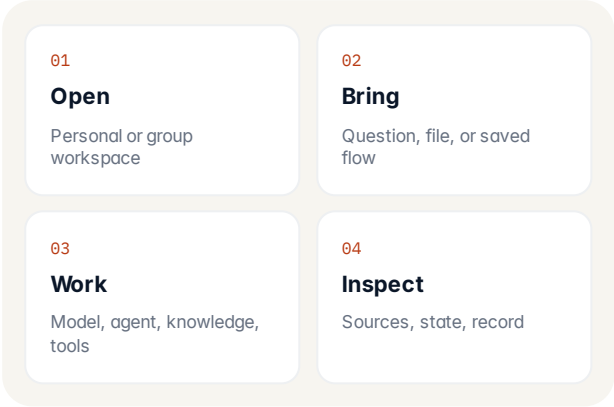
Door one: use the workspace

Fabric is the user-facing starting point: people work with conversations, files, saved knowledge, shared rooms, models, agents, and workflows without operating the substrate themselves.

In managed Fabric, Dweve operates the service boundary and public-Mesh capacity. A business uses the Fabric web app, its business API, or the Agent SDK. Nexus and Spindle capabilities support work through Fabric and the public Mesh, but the customer does not receive those products for direct operation. Aura is a separate CLI coding agent. Core is not part of managed access. Direct product operation belongs to a licensed tier.

The workspace is more than a chat window in the product description. It has identity and participation, personal and group contexts, documents and governed knowledge, model and agent registries, flow graphs, roles, work objects, and records. Those concepts matter when a workspace moves from personal assistance into organisational work: someone must own the sources, decide which models and agents are permitted, control sharing, and retain the record.

The managed option reduces infrastructure work. It does not remove governance work. The organisation still decides purpose, data, access, authority, retention, acceptable output, and what a person must review.



Fabric is the surface. It does not collapse the responsibilities underneath it.

OPERATED BY

Dweve

Within the managed service boundary described on the site.

CUSTOMER OWNS

Purpose + authority

Sources, roles, acceptable use, and review remain organisational decisions.

FASTEST USE

Workspace

The site positions Fabric as the direct starting point.

NEXT STEP

Proof

Use a real bounded workflow, not a generic conversation.

BUILD ON IT

Door two: build through the API

The API option is for teams that already have applications, users, processes, or operational systems and want Dweve responsibilities behind their own surface.

The managed business path combines the Fabric web and app interface with a business API and Agent SDK. Nexus and Spindle capabilities are used through Fabric on Dweve's public Mesh; they are not handed over for direct operation. Aura remains a separate CLI coding agent. Direct control of Nexus, Spindle, Core, and the other platform products belongs to licensed tiers. An API integration must preserve those boundaries instead of blurring source, model, agent, tool, policy, and human state into one opaque response.

A sound API integration begins with a typed workload contract: the initiating event, identity, minimal source references, policy or rule version, requested capability, permitted actions, review authority, expected result, error states, and evidence to retain. The website repeatedly presents typed contracts as the coupling surface between layers. The same discipline should exist at the customer boundary.

The customer operates the calling application and owns its user experience, identity mapping, upstream data quality, downstream action, and retention duties. The service boundary operates the called platform components in the selected posture.

ILLUSTRATIVE REQUEST CONTRACT

```
work_id       case-4f9e
actor         reviewer:credit-ops
capability    explain_decision
sources       applicant_snapshot, policy@2026-03
authority     recommend_only
review_gate   credit_officer
result        reason + source_refs + state
evidence      trace_ref + work_receipt
```

The fields are illustrative. The architectural point is that authority and evidence are part of the request, not an afterthought.

INTEGRATION TEST

Reject any design in which a consequential result can leave the platform without a stable work identity, source/version references, an authority state, and a route to the supporting record.

OWN IT

Door three: operate the licensed platform

A licensed deployment moves the platform onto infrastructure the customer controls and transfers operational responsibility with it.

The website describes licensed operation as the same platform on customer hardware, behind the customer firewall and under customer keys. It also states that usage on included licensed products is not charged per query. Those are separate facts: the first concerns operational control; the second concerns the commercial shape. Neither makes operation effortless.

Customer control means customer duties. Infrastructure, capacity, identity integration, key custody, network policy, backups, restore, monitoring, change windows, administrator access, evidence retention, and incident handling need named owners. Dweve supplies the artefacts, support, and rights defined by the selected licence and scope; the customer runs the environment.

The licensed option matters when the workload cannot cross a chosen perimeter, needs low or local latency, must continue independently of a managed service, requires a customer change process, or needs rights unavailable in a service plan. It should not be chosen merely because “on-premises” sounds safer. The installed topology and operating practice still have to earn that conclusion.

RESPONSIBILITY	BUSINESS WORKSPACE	LICENSED
Infrastructure	Dweve operates	Customer operates
Keys	Service boundary	Customer custody
Network perimeter	Managed service	Customer-defined
Updates	Dweve process	Customer promotion
Usage charge	Metered/monthly	Included licensed use
Governance	Customer	Customer

Exact rights, products, support, and conditions follow current pricing and contract scope.

ACCEPTANCE EVIDENCE

Install, isolate, update, roll back, restore, replay, rotate a key, remove Dweve connectivity, and continue the selected workflow before calling the posture customer-controlled.

DISCONNECTED OPERATION

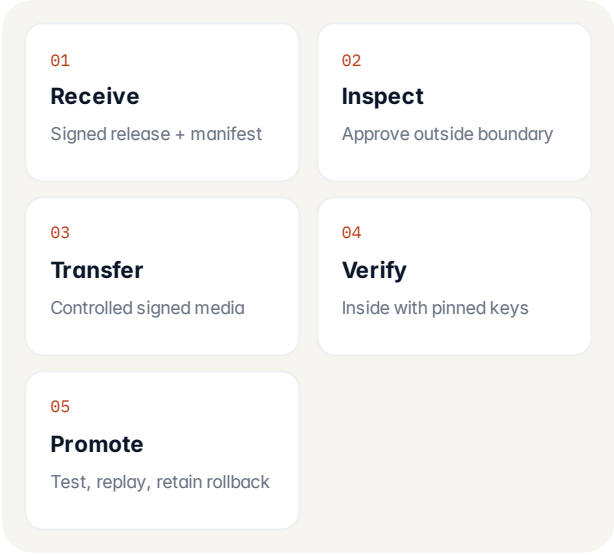
Air gap is a deployment posture, not a slide

The deployment page makes unusually specific promises: the same binaries, no outbound dependency, an offline-signed licence, and continued licensed operation without a kill switch.

Those promises are testable. A real air-gapped acceptance exercise starts with network absence, not a firewall exception list. Installation media, artefact manifests, signatures, licence validation, model and policy packages, configuration, and documentation enter through a controlled transfer. The customer verifies them inside the perimeter. Operation, evidence generation, replay, administration, and licence validation then proceed without an outside service.

Updates reverse the discipline: obtain a signed release outside, inspect and approve it, transfer it on controlled media, verify it again inside, stage it, run compatibility and replay tests, promote it, and retain both the new and previous artefact. A rollback has to work without contacting Dweve.

Disconnected does not mean dependency-free. The environment still depends on local identity, clocks, storage, keys, source feeds, model and policy packages, operators, and support procedures. Every one needs a degraded-mode decision.



A practical media-shuttle path. Exact procedures belong in the deployment runbook.

AIR-GAP ACCEPTANCE	No outbound dependency	
LICENCE	**Verifies offline**	TEST
INFERENCE	**Runs with network absent**	TEST
TRACE	**Created and retained locally**	TEST
REPLAY	**Verifier runs inside**	TEST
ROLLBACK	**Previous signed artefact**	TEST

A signed claim becomes assurance only after the installed environment passes the exercise.

COMPOSITION

One shared substrate does not mean one giant application

The platform claim is coherence through explicit boundaries: products reuse lower responsibilities rather than rebuilding them in every surface.

The stack page describes a downward dependency structure. Upper responsibilities consume lower ones through typed contracts; lower layers do not depend on the user-facing products above them. This matters because the platform is meant to be replaceable and deployable in different shapes. If Fabric knew the internal memory layout of a Core kernel, or if a parser depended on the workspace UI, that promise would collapse.

A shared substrate therefore means shared rules for identity, artefacts, versions, numeric behaviour, retrieval, policy, proof, storage, security, telemetry, and transport, not one database into which every product writes whatever it wants. The boundary is the part another team or replacement implementation must keep. Everything behind it can evolve if the contract and acceptance evidence remain valid.

The architecture has to be tested at seams. Can a model or provider change without losing work identity? Can a retrieval component change without dropping source references? Can a managed workload move to a licensed environment and replay retained decisions? Those are composition tests.

06	**Compute**	Core execution and Kera compilation
05	**Policy + proof**	Rules, certificates, and replay
04	**Cognition**	Loom experts, reasoning, and solvers
03	**Knowledge**	Spindle and source-aware retrieval
02	**Work**	Nexus, Aura, and governed workflows
01	**Surface**	The products remain distinct. No giant Fabric box swallows them.

A responsibility view, not a replacement for the complete twelve-layer technical map.

BOUNDARY TEST

For every arrow, write down the input type, output type, version, error model, authority state, evidence reference, and compatibility rule. If the answer is “the shared database”, the boundary is not yet explicit.

CURRENT PUBLISHED WEBSITE SOURCES

The product map in one page

Each product has one leading responsibility. The selection question is what the workload needs, not how many product names can fit into a proposal.

PRODUCT	OWNS	RECEIVES	RETURNS	DOES NOT REPLACE
Fabric	Workspace and participation	Questions, files, users, flows	User result + work view	Cognition or runtime
Loom	Cognition	Typed task + context + sources	Reasoned result + trace state	Team authority
Nexus	Organised agent work	Mission, roles, capability, policy	Work result + receipt	Base reasoning
Spindle	Knowledge lifecycle	Sources, rights, processing policy	Knowledge + lineage	Collector or parser
Aura	Development loop	Goal, repository, tools, permissions	Change + engineering record	Generic business work
Mesh	Distributed capacity	Proposal, policy, resource, artefact	Result + receipt/settlement	Local product logic
Charter	Organisation record	Parties, sources, events, policy	Portable state + workflow record	Every specialist system
Core	Binary AI framework	Graph + numeric profile	Result + verification data	Language/compiler
Kera	Language and compiler	Typed graph programme	Target artefact + manifest	Core runtime

The full product pages define the detailed boundary. This table is the selection index.

COUNT NOTE

The table names the the current product pages. It deliberately does not repeat the website's disputed "eight surfaces" count.

BOUNDARY THAT MATTERS

Fabric, Loom, and Nexus solve three different questions

The user surface, cognition system, and organised-work runtime cooperate, but none is a synonym for “the AI”.

Fabric: where people participate

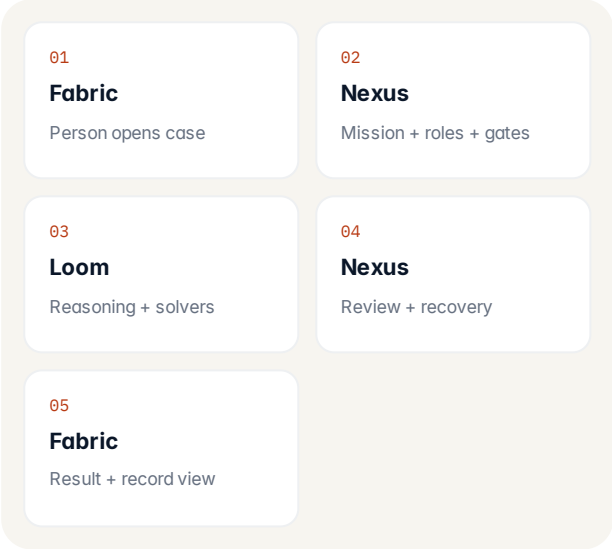
Fabric owns the workspace graph: people, rooms, objects, saved work, model and agent choices, knowledge surfaces, roles, and the view of the resulting record. It answers: **where does the person work, what can they see, and which shared context are they in?**

Loom: how the task is reasoned through

Loom owns the cognition graph described through perception, memory, search, reasoning, exact solvers, experts, routing, merge semantics, expression, versions, and replay. It answers: **which cognitive work is required, who or what performs it, and how is the result composed?**

Nexus: how accountable work is organised

Nexus owns agents as workers in a mission: identity, capability, task passport, team shape, workflow, hand-off, memory, policy checkpoint, approval, recovery, and receipt. It answers: **who does each part, under which authority, and what happens when work fails?**



A simplified collaboration path. Knowledge, policy, tools, and proof join at their own boundaries.

SURFACE
Fabric
People, workspace, context, visibility.

ORGANISATION
Nexus
Workers, tasks, gates, recovery.

COGNITION
Loom
Experts, search, reasoning, solvers.

KNOWLEDGE BOUNDARY

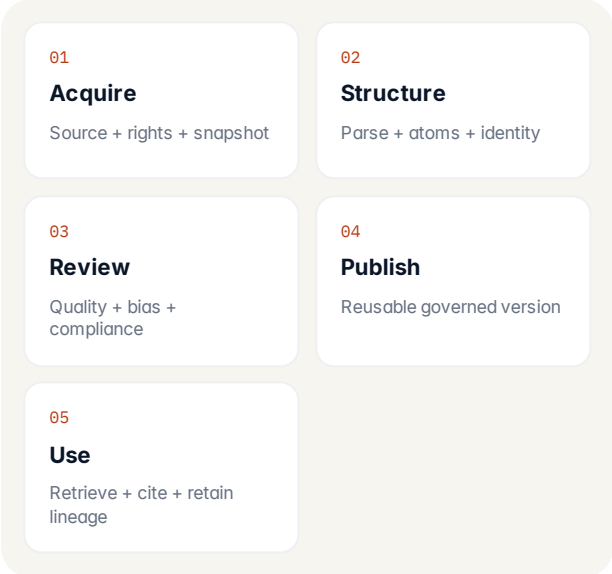
Spindle owns the knowledge lifecycle, not merely retrieval

A retrieved passage is only safe to reuse when its source, version, rights, transformations, quality state, and later change remain attached.

Seven stages and hard gates exist for one reason: a fact that cannot be traced back to its source is not a fact. Spindle is a pipeline, not a pile. Each fact enters as a candidate and advances only when it meets the quality, consensus, and provenance requirements of the next stage. AI Act Article 10 compliance requires a quality score of 0.7 or above at the Certified gate. Consensus requires at least 3 independent sources with a diversity score of 0.6 or above (entropy-based). If a fact fails a gate it stays where it is, flagged with a reason. It does not leak into training or answers.

That distinction matters when a source changes. A conventional copied chunk may be overwritten in the current index. A governed record needs two truths at once: the decision made in March must still point to the March source and transformation state, while new work should use the approved July version. Historical provenance and active knowledge are related, not interchangeable.

The boundary also carries rights. Access to a source does not automatically grant every agent, user, site, model, or downstream export the right to use or disclose it. Rights and policy need to travel with the knowledge identity.



The lifecycle updates knowledge without erasing what an earlier decision used.

KNOWLEDGE IDENTITY	Policy handbook · March edition	
SOURCE	publisher/path + snapshot	BOUND
RIGHTS	workspace + purpose scope	ACTIVE
TRANSFORM	parser/config + root	BOUND
REVIEW	quality owner + state	APPROVED
SUCCESSOR	July edition	LINKED

Illustrative shape; product interfaces and fields require technical confirmation.

ENGINEERING · DISTRIBUTED CAPACITY · ORGANISATION RECORD

Aura, Mesh, and Charter govern three different estates

These products are easiest to confuse when reduced to broad labels such as “agents”, “network”, or “business platform”. Their centres are distinct.

AURA

Engineering estate

A goal becomes a plan, bounded tool use, repository changes, tests, reviews, corrections, and an engineering record. Permission and quality gates remain in the loop.

MESH

Distributed estate

A work proposal crosses sites without casually crossing local ownership. Capacity, policy, isolation, faults, receipts, return, and settlement have a lifecycle.

CHARTER

Organisation estate

Parties, customers, money, work, knowledge, sources, domain rules, approvals, events, providers, and packs live on a portable organisation graph.

The common platform beneath them

All three can use cognition, knowledge, policy, proof, and compute. That commonality should reduce duplicated implementation; it does not merge their authority models. Aura may propose a code change but needs repository and tool permissions plus engineering acceptance. Mesh may schedule work but needs participant policy, isolation, evidence, and owner return. Charter may propose an organisational action but needs a domain policy gate and an authorised person or workflow state to give it effect.

Selection follows the governed asset. Choose Aura when the central asset is a software change and its engineering evidence. Choose Mesh when work or capacity crosses independently owned sites. Choose Charter when the central asset is the organisation’s durable operating record and the workflows around it.

COMBINATION EXAMPLE

A Charter workflow may initiate work in Nexus, use Loom for reasoning, retrieve sources through Spindle, ask Aura to implement an approved systems change, and use Mesh for distributed capacity. Composition does not transfer original authority: every hand-off carries its own contract and gate.

COMPUTE BOUNDARY

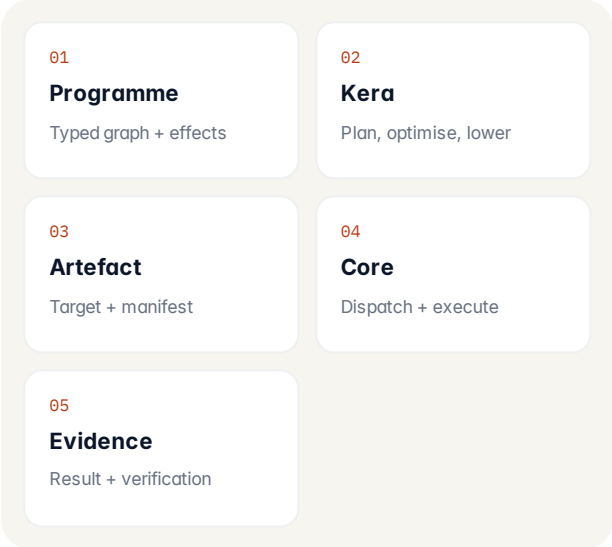
Core and Kera separate what the graph means from how it runs

Core is the binary AI framework and runtime responsibility. Kera is the graph-native language and compiler path that prepares work for supported targets.

Core is described through an operation vocabulary, numeric formats, model and graph representation, kernels, packing, dispatch, training and inference routes, runtime modes, backend targets, determinism, and verification. Its job is to execute supported computation under an explicit numeric and target contract.

Kera is described through source and graph programmes, types, effects, planning, fusion, allocation, dispatch, intermediate and target artefacts, direct emission, target selection, distributed work, tooling, and lowering into Core. Its job is to reject invalid programmes early, preserve visible effects and types, and produce a versioned artefact for a target.

The separation lets an evaluator ask two questions. Is the programme valid and is the compiler transformation semantics-preserving? Then: does the runtime implement the required operations, formats, target, performance, and deterministic result? One successful benchmark cannot answer both.



A conceptual path. Exact IR names, target status, and proof artefacts follow current specifications.

EVALUATION SPLIT

```
compiler acceptance
  type and shape checks
  effect visibility
  legal transforms
  target artefact identity

runtime acceptance
  operation coverage
  numeric result contract
  backend parity
  performance and resources
```

Treat compilation and execution evidence as related but separate assurance surfaces.

02

A REQUEST FROM START TO FINISH

The result is the final state of a governed process

A useful platform explanation names the question, sources, context, reasoning, workers, tools, policies, human authority, output, evidence, and failure paths, not only the answer at the end.

Every material transition has an owner and a record boundary.

ILLUSTRATIVE REGULATED WORKFLOW

The walkthrough: explain a declined credit decision

The example is fictional but structurally faithful to the banking and trace material on the website. Its purpose is to make every platform responsibility concrete.

A bank receives a supervisory question about a revolving-credit limit increase declined four months earlier. The examiner asks for the applicant factors used, the policy in force on that date, the model or rule path, any exception, the officer who reviewed the recommendation, the notice sent to the customer, and a replay of the same decision.

Today those elements may sit in a credit engine, policy wiki, model registry, case system, email thread, identity log, data warehouse, and archive. The visible score is not the decision record. Reconstructing the whole state can take days and may still leave uncertainty about which version was active.

In the Dweve platform model, the workflow is scoped around one stable work identity. The bank's current source systems remain. The platform receives the minimum case snapshot and versioned sources it needs, organises specialist work, runs the relevant reasoning and exact calculation, applies policy gates, presents a recommendation to an authorised officer, and retains the resulting evidence references.

ILLUSTRATIVE CASE	case-4f9e · limit increase	
QUESTION	**Why was the request declined?**	FIXED
CASE DATE	**2026-03-12**	FIXED
POLICY	**credit-policy@2026-03**	PINNED
AUTHORITY	**recommendation + officer review**	ACTIVE
EVIDENCE	**source + rule + owner + replay**	REQUIRED

Identifiers and dates are illustrative. The source website provides the scenario pattern, not a customer result.

SUCCESS CONDITION

The examiner can retrieve the original record, verify its integrity, replay the supported decision path, identify the human decision state, and read a reason without opening the live production case.

INITIATION

Step 1: give the work a stable identity and authority boundary

Before a model or agent wakes up, the platform needs to know what work this is, who initiated it, what it may do, and what must remain a recommendation.

Work identity is the spine through which later sources, tasks, artefacts, approvals, errors, and results can be joined. A conversational session identifier is not enough when the work has a business or legal lifecycle. The identity must survive a page reload, agent substitution, model change, retry, review queue, deployment migration, and later archive retrieval.

Authority is equally early. The request states whether the platform may retrieve and explain, prepare a draft, recommend an outcome, execute a bounded action, or only assemble evidence. A policy and a human role determine the next transition. If authority is added only in the interface after reasoning finishes, downstream tools may already have acted outside the intended boundary.

Nexus task passports, Fabric work identity, Aura goal records, Charter proposed actions, and sector decision packets all express variants of this principle on their product pages: the work, actor, state, and permitted transition are explicit objects.

FIELD	PURPOSE	FAILURE PREVENTED
Work ID	Join every transition	Orphaned actions/results
Actor	Name initiating identity	Anonymous authority
Purpose	Bound the use	Context reuse for another aim
Capability	Select allowed work	Tool/model overreach
Authority	Recommend, draft, or act	Accidental legal effect
Review gate	Name required role/state	Rubber-stamp or no owner
Retention	Set record expectation	Evidence disappears

An illustrative initiation contract derived from recurring product responsibilities.

DESIGN RULE

The reviewer should be able to see the same work identity and authority state that the runtime enforces. Display text alone is not a control.

GROUNDING

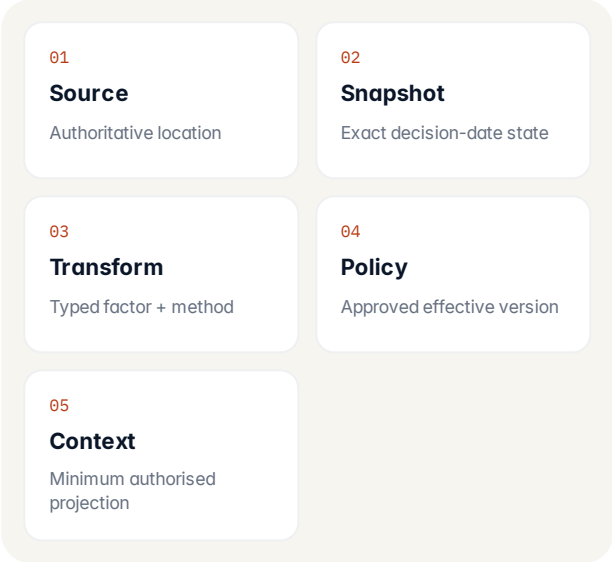
Step 2: bind the source and version that were actually used

A later reviewer needs the policy and case state from the decision date, not the policy currently open in the knowledge base.

The case snapshot must be minimal but sufficient. Instead of handing a broad customer record to every component, the integration prepares the typed factors required for the selected workflow and references the authoritative source record. Each factor needs provenance, meaning, effective time, and correction behaviour. A derived factor also needs the transformation that produced it.

The policy is a versioned runtime input. A human-readable title is not enough: the record needs an artefact identity, effective interval, approval state, source, and relevant rule or clause. If the policy changes in July, the March decision continues to reference the March artefact; a new decision follows the newly approved version.

Spindle and the open parsing, collection, and retrieval foundations support this source path. The platform-level requirement is simpler to state: no candidate or conclusion should lose the path back to the source bytes or structured record from which it came.



Grounding is a chain of identities, not a citation pasted onto generated prose.

SOURCE MANIFEST	Inputs pinned for case-4f9e	
APPLICANT	case snapshot hash/reference	PINNED
POLICY	credit-policy@2026-03	PINNED
MODEL/RULES	approved manifest	PINNED
TRANSFORM	factor map + version	PINNED
RIGHTS	purpose + role scope	CHECKED

CONTEXT BOUNDARY

Step 3: project context instead of copying the whole estate

The right specialist receives the smallest authorised view that lets it perform its part of the task.

Context is not one unbounded prompt. It is a projection assembled for a particular task, identity, policy, and point in the workflow. Fabric knows the workspace and participant; Nexus knows the mission, role, and task; Loom needs the cognitive problem and supporting sources; a tool needs only its action parameters; the reviewer needs the facts, rule, result, uncertainty, and evidence needed to exercise authority.

This separation reduces accidental disclosure and makes later explanation possible. If every component receives the entire case, it becomes difficult to show which facts influenced a result and impossible to prove that a restricted fact did not cross a boundary. A projection can be logged by identity and version, reviewed, and regenerated from authorised sources.

Context budgets on the Aura page and context projection on the Fabric page describe product-specific versions of the same concern: relevance, permission, and resource limits must constrain what enters a work step.

WORKSPACE

Participant view

Case status, visible sources, prior messages or work, and permitted actions.

AGENT

Task view

Mission, role, capability, inputs, tools, policy, and expected artefact.

COGNITION

Reasoning view

Question, grounded facts, sources, constraints, solver needs, and output contract.

REVIEWER

Authority view

Recommendation, decisive factors, rule, uncertainty, conflicts, history, and evidence route.

PRIVACY TEST

Record which context projection crossed each boundary. Then attempt to retrieve a prohibited field from every downstream component. “We did not prompt for it” is not evidence that it was unavailable.

COGNITION

Step 4: Loom decomposes cognition into inspectable jobs

The website does not present Loom as one model asked to do everything. It presents perception, memory, search, reasoning, solvers, experts, routing, and expression as separate work.

For the illustrative credit case, perception reads the typed case and policy structures; memory supplies authorised case or domain context; search retrieves relevant policy sections and supporting material; reasoning relates the factors to the rule; an exact solver handles calculations or constraint satisfaction that should not be improvised in language; specialist experts inspect different aspects; merge semantics reconcile their outputs; and expression turns the already structured result into an explanation.

This decomposition creates review points. A missing policy passage is a search problem, not proof that the reasoning model failed. A numeric mismatch is a solver or numeric-profile problem. A fluent but unsupported sentence is an expression or output-validation problem. An expert conflict is a routing and merge problem. The system can refuse or hand back when a required source or check is unavailable.

The website publishes counts and benchmark statements for experts, active routing, search, and the compute beneath Loom. Those numbers belong in a technical evaluation with the relevant system card and harness. The platform decision here is whether the decomposition fits the workload and exposes the states the reviewer needs.

07	**Expression**	Render the checked result for its audience
06	**Experts**	Apply specialist capability selectively
05	**Solvers**	Perform exact or constrained work
04	**Reasoning**	Relate facts, constraints, and alternatives
03	**Search**	Retrieve candidates with source identity
02	**Memory**	Bring authorised retained context
01	**Perception**	Interpret typed input and source structures

The product page calls these distinct internal responsibilities. Exact implementation belongs in the Loom handbook.

SOLVERS AND DETERMINISTIC COMPUTE

Exact work should not be delegated to fluent prose

When the workflow depends on arithmetic, constraint satisfaction, a rule table, or a fixed algorithm, the result needs a defined operation and numeric contract.

A language model can describe a calculation while quietly performing a different one. The platform pages repeatedly separate learned or expert reasoning from exact solvers and deterministic lower-layer computation. In the credit case, affordability ratios, thresholds, date comparisons, and policy constraints should be represented as explicit inputs and operations, not inferred from a paragraph after the fact.

The selected numeric profile controls representation, rounding, range, overflow behaviour, and supported target parity. Kera compiles the graph for a target; Core executes the supported operations; numeric and verification foundations provide the checks described on their pages. The operation graph, input identities, numeric profile, compiler/runtime artefacts, and output become part of the evidence path.

Determinism does not make the policy good or the factors correct. It makes a supported execution reproducible under pinned inputs, versions, and modes. That narrower guarantee is valuable precisely because it isolates later disagreement: reviewers can debate the source or rule without first debating which machine produced the “real” number.

ILLUSTRATIVE EXACT SUBGRAPH

```
inputs
  verified_income_cents
  committed_outgoings_cents
  proposed_limit_cents
  policy.threshold@2026-03

operations
  disposable = income - outgoings
  ratio      = proposed_limit / disposable
  eligible   = ratio <= threshold

outputs
  ratio, eligible, rule_ref
```

Illustrative logic only. A bank owns the legitimate policy and verified input definitions.

ACCEPTANCE TEST

Run the same pinned subgraph on every target proposed for production. Compare output bytes or the exact guarantee of the chosen determinism mode, then retain the artefacts and verifier result.

AGENTS AND WORKFLOW

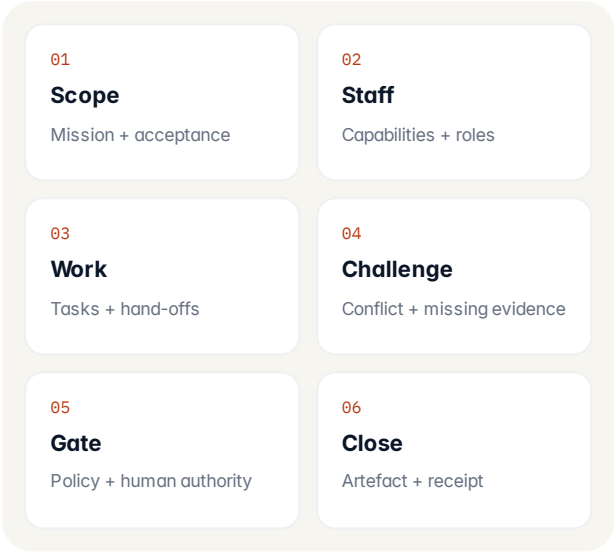
Step 5: Nexus turns cognitive work into accountable teamwork

A specialist result becomes organisational work only when roles, tasks, hand-offs, policy, state, and recovery are explicit.

A role has a purpose, capabilities, boundaries, inputs, outputs, and responsibility within the wider operation. Nexus describes agents by what they can do and tasks by what they require, matching work to available capability while accounting for current state, demonstrated proficiency, load, language, modality, policy, and resource limits.

Each task has inputs, output expectations, dependencies, permitted tools, policy checks, and a state. Collaboration happens through typed hand-offs so that one specialist can cite another's artefact without rewriting it. A disagreement is retained and routed to a merge, challenge, or human decision; it is not silently averaged into smoother prose.

The workflow graph also defines recovery. If a source is unavailable, an agent fails, a model cannot be reached, a tool is denied, or the produced artefact does not meet its contract, Nexus chooses from allowed recovery paths: retry with the same state, reassign, substitute an approved provider, return to an earlier checkpoint, escalate, or fail closed.



Organisation is a runtime responsibility, not a metaphor for several prompts.

STABLE OBJECT

Task passport

Identity, capability, state, and expected artefact.

CONTROL

Policy checkpoint

Action or transition can be blocked before effect.

FAILURE

Recovery path

Retry, substitute, reroute, escalate, or stop.

OUTCOME

Work receipt

Who and what participated in the result.

ACTION

Step 6: tools are requested through a permission boundary

Reading a source, querying a system, writing a file, changing code, or submitting a decision are materially different capabilities.

Tool use is where an answer becomes an action. The platform pages show permission matrices, policy checkpoints, proposed actions, guardians or output filters, capability-limited execution, sandboxing, and execution records across Aura, Nexus, Charter, Mesh, and Selvedge. The shared principle is that an agent does not inherit every capability available to the host process.

A tool request names the work and actor, requested capability, parameters or input artefacts, purpose, policy result, and required approval. The host decides whether the request is allowed, denied, altered, or held for a human. The execution boundary limits network, files, credentials, resources, and downstream effects according to the selected tool and posture.

In the credit example, retrieving a policy section may be automatically permitted for the authorised case role. Writing a draft explanation may be allowed inside the work record. Sending a customer notice or changing a credit limit should remain behind an authenticated human or existing decision-system gate. The exact boundary belongs to the bank.

CAPABILITY	DEFAULT TREATMENT	EVIDENCE
Read approved source	Role + purpose policy	Source identity + query
Run exact calculation	Pinned graph/profile	Inputs + artefact + result
Draft explanation	Bounded work object	Draft + source refs
Write case note	Approval or scoped role	Change + actor + prior state
Send notice	Human/authoritative system gate	Approved content + delivery state
Change limit	Outside example autonomy	Existing bank control

Illustrative policy. The customer defines the legitimate authority for each action.

SECURITY RULE

A denied tool request is a first-class result. It must not disappear as a vague “agent failed” message; the work record should show which boundary stopped it and what recovery followed.

AUTHORITY

Step 7: human oversight must change what happens next

A review screen is not meaningful oversight if the person lacks information, time, competence, or power to refuse and correct the result.

The officer in the walkthrough receives a recommendation, not a decorative approval button. The review view should identify the case and requested decision; show the decisive verified factors; cite the policy and version; expose conflicts, missing sources, and uncertainty; identify the model, expert, solver, and tool path at the appropriate level; show prior overrides or related state; and link to the supporting evidence.

The officer can approve, reject, request correction, add a reason, route to a second reviewer, or stop the workflow. The chosen action changes the authoritative state and is attached to the same work identity. An override is not an embarrassment to hide. It is evidence about policy, data, model behaviour, edge cases, or training that may need follow-up.

The organisation must also decide whether review occurs before legal or operational effect, after a reversible low-impact action, or only on exception. Dweve can implement gates and records. It cannot decide which governance design is lawful or proportionate for the customer's purpose.

REVIEW CARD	Decline recommendation	
REASON	**Rule M-7 threshold exceeded**	SHOWN
FACTORS	**Verified income + commitments**	SHOWN
POLICY	**credit-policy@2026-03**	SHOWN
CONFLICT	**No unresolved sources**	CLEAR
AUTHORITY	**Officer decision required**	OPEN

The reviewer can approve, reject, correct, escalate, or stop. The interface must enforce the same state as the runtime.

OVERSIGHT TEST

Give reviewers cases with a wrong source, a conflicting rule, a persuasive but invalid explanation, and a correct refusal. Measure whether they detect the problem and whether the workflow respects their response.

OUTCOME

Step 8: close the work with a result and a record

The final result is not only prose. It is an outcome state connected to the exact sources, rules, work, tools, reviewer, and artefacts that produced it.

For the illustrative case, the customer-facing result is a plain explanation of the decision, the material factor, the applicable policy, and the route to human review or appeal. The internal result also includes the authoritative officer state, case-system update or reference, policy and model manifests, exact calculation output, agent artefacts, tool records, and evidence identifiers.

Different readers receive different projections of the same underlying work. The customer should not receive internal fraud signals, privileged notes, secrets, or unrelated personal data. The examiner may need more technical detail and independent verification material. Operations needs status and recovery information. The board needs aggregate control evidence, not every intermediate value.

The work can close only after its required output contract is satisfied. Missing evidence is not a successful result. A refused or escalated case can still be a correctly completed workflow if that is the contractually defined safe outcome.

AFFECTED PERSON

Plain reason

Decision, decisive factor, rule/policy, human route, correction or appeal path.

REVIEWER

Decision view

Facts, sources, rule, reasoning state, conflicts, authority, and actions.

EXAMINER

Evidence packet

Versions, identities, signatures, proof or trace, replay path, and retention state.

OPERATOR

Run state

Services, errors, retries, recovery, delivery, archive, and change history.

ONE HISTORY, BOUNDED VIEWS

The goal is not one unrestricted dossier for everyone. It is one coherent history from which authorised, purpose-appropriate views can be produced and verified.

EVIDENCE VOCABULARY

The trace is not the same thing as a log or a work receipt

The website uses several evidence concepts. A real evaluation has to know which question each one answers.

ARTEFACT	PRIMARY QUESTION	TYPICAL CONTENTS	BOUNDARY
Source provenance	Where did this fact come from?	Source identity, snapshot, rights, transform, lineage	Does not prove the conclusion
Decision trace	Why and how did this computation decide?	Inputs, intermediates, policy checks, outputs, root/signature	Needs supported replay mode
Event ledger	What happened across the system?	Typed events, order, actors, artefacts, chain	Does not expose every reasoning step
Execution transcript	What did bounded code actually do?	Module, capabilities, calls, results, errors, seal	Does not prove business purpose
Proof certificate	Can a verifier check the statement?	Claim, commitments, proof, format, root/signature	Depends on statement and verifier
Work receipt	Who or what contributed to this result?	Mission, roles, tasks, sources, tools, approvals, artefacts	Organisational record, not all runtime bytes
Audit/application log	What did the application report?	Human-readable operational events	Best-effort unless separately guaranteed

The exact product formats belong in the trace and open-foundation guides. This table prevents evidence terms from becoming interchangeable marketing words.

A consequential workflow may use several artefacts linked by stable identities. That is not needless duplication: the source, computation, system chronology, sandbox execution, formal statement, and organisational work are different assurance questions. The integration should expose their relationships without stuffing every byte into one file.

VERIFICATION

Replay means re-checking pinned work, not reenacting a story

The trace page describes a verifier that authenticates the artefact, recomputes its Merkle root, re-executes recorded operations, and replays policy gates, failing closed at the first invalid stage.

A replay starts from the retained artefact and its declared mode. The verifier parses and authenticates it, reconstructs the canonical identity of inputs and nodes, checks the root and signature, re-executes the supported operation graph using the pinned artefacts and determinism contract, and evaluates the recorded policy path. The output matches the expected root or verification fails.

That is different from asking a current model the same question again. The model may have changed, the source may be newer, the prompt may differ, and a probabilistic path may produce different text. Replay is also different from showing a video or application log. It checks the retained computational and policy evidence according to its format.

The website states that verification can be offline and supplier-independent through the open verifier and public formats. A buyer should test that by archiving the verifier, public keys, specs, sample artefacts, and every dependency required to verify after the production environment is unavailable.

01 **Parse + authenticate**
Validate format, manifest, identity, and signature material.

02 **Recompute root**
Rebuild the canonical Merkle structure and compare.

03 **Re-execute**
Run the recorded supported operations under the declared mode.

04 **Replay policy**
Check the policy gates and versions that governed the result.

05 **Return status**
Clean verification or a precise failing stage; no partial success.

ACCEPTANCE EXPECTATION

```
$ verifier check case-4f9e.aion
stage 1  parse/authenticate   ok
stage 2  merkle root          ok
stage 3  operation replay     ok
stage 4  policy replay        ok
result   verified             exit 0
```

Illustrative console output. The current trace documentation owns exact commands and formats.

FAILURE AND RECOVERY

Failure is part of the architecture, not an exception paragraph

A governed workflow needs explicit behaviour when a source, model, agent, tool, policy, reviewer, target, or evidence check fails.

FAILURE	UNSAFE RESPONSE	GOVERNED RESPONSE	RECORD
Source unavailable	Invent or use stale copy silently	Stop, use approved snapshot with warning, or escalate	Source state + choice
Sources conflict	Choose the convenient one	Preserve conflict; apply ownership rule or human review	Both sources + resolution
Model/provider down	Switch invisibly	Use approved substitute with manifest or pause	Provider change + result
Agent artefact invalid	Continue with partial output	Reject contract; retry, reassign, or escalate	Failure + recovery
Tool denied	Find another path around policy	Record denial; request approval or stop	Request + gate result
Policy missing	Treat absence as approval	Fail closed or route to policy owner	Missing version + owner
Reviewer absent	Auto-approve after timeout	Queue, delegate by explicit rule, or stop	Authority transition
Replay mismatch	Publish anyway	Quarantine result; investigate artefact/target	Mismatch + incident

The exact recovery policy belongs to the customer and product configuration. The platform must make it enforceable and observable.

SAFE COMPLETION

Refusal, escalation, or a blocked action can be a successful workflow outcome when continuing would violate the source, policy, authority, or evidence contract.

END-TO-END RESULT

The complete packet for the illustrative case

The packet is not one indiscriminate archive. It is a linked manifest that lets authorised reviewers retrieve and verify the artefacts relevant to their question.

The case manifest identifies the work, purpose, initiating actor, decision date, deployment, and final state. It references the minimal case snapshot, source policy and effective version, data transformations, model and rule manifests, context projections, Loom reasoning and solver artefacts, Nexus tasks and hand-offs, tool requests and results, officer review, customer-facing explanation, authoritative-system update, and trace or certificate references.

Integrity fields detect a change to the retained identities or artefacts. Access policy determines which reviewer may retrieve which part. The affected person receives the reason and contestability route. The officer sees the decision state and source details. The examiner can receive the evidence needed for independent review. Operations can investigate failure and delivery without reading unnecessary case content.

Retention must preserve the verification dependencies for the required period: formats, public keys, verifier, relevant artefacts, and version metadata. Deletion, correction, and legal-hold requirements have to be designed against that evidence need; the platform cannot decide the lawful balance by itself.

DECISION PACKET	case-4f9e · closed	
IDENTITY	**work + purpose + date**	SEALED
SOURCES	**case + policy + transforms**	SEALED
COGNITION	**reasoning + solver artefacts**	SEALED
WORK	**tasks + tools + recovery**	SEALED
AUTHORITY	**officer + decision state**	SIGNED
OUTPUT	**notice + delivery reference**	CLOSED
PROOF	**trace + verifier dependencies**	VERIFIED

Illustrative manifest. A production schema must be taken from the implemented interfaces and retention design.

EXAMINER TEST

Select a random historical case. Retrieve the packet without production engineers assembling it, verify it in an isolated environment, and reconcile the human state with the authoritative case system.

BOUNDARY OF ASSURANCE

What a valid proof still cannot tell you

Integrity, deterministic replay, and a complete record answer important questions. They do not answer every question about a legitimate or beneficial decision.

A valid trace can show that the recorded operation graph ran on the recorded inputs, passed the recorded policy gates, produced the recorded result, and has not been silently altered under the declared verification scheme. It cannot establish that the applicant data was true, that the policy was fair or lawful, that the purpose was proportionate, that missing context would not change the outcome, or that the reviewing officer exercised meaningful judgement.

Likewise, source provenance can show where a factor came from without proving the source is unbiased. A signed officer state can prove who approved without proving they understood. A deterministic calculation can reproduce the same threshold result without proving the threshold is a good policy. A sandbox transcript can show what code touched without proving the code served a legitimate purpose.

This is why the platform model includes organisational governance around technical evidence. Evidence makes questions answerable. People, policy owners, affected persons, auditors, security teams, and regulators still have to ask them.

CAN SHOW

Integrity

The retained artefact and committed inputs have not changed undetected.

CAN SHOW

Reproduction

The supported pinned work re-executes or verifies under the declared mode.

CANNOT SETTLE

Legitimacy

Whether the purpose, policy, impact, and data use are acceptable or lawful.

CANNOT SETTLE

Quality of judgement

Whether sources were sufficient and the human decision was competent or fair.

GOVERNANCE PRINCIPLE

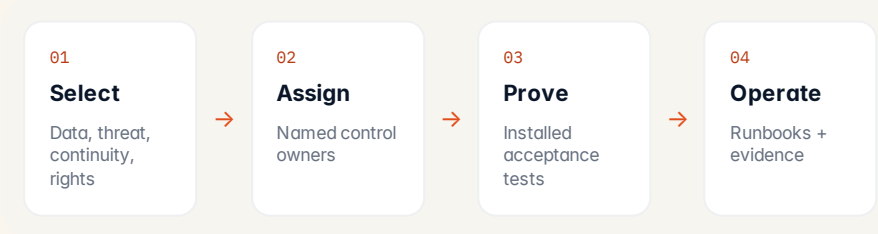
A platform that exposes evidence reduces the cost of accountability. It does not transfer accountability away from the organisation using it.

03

DEPLOYMENT AND OPERATION

Where control sits changes. The platform model does not.

Managed public-Mesh participation, licensed operation, and air-gapped operation are responsibility arrangements. A serious selection compares custody, connectivity, operations, change, recovery, and evidence, not generic hosting labels.



The deployment decision is complete only when each promised property has an owner and an acceptance test.

DEPLOYMENT DECISION

Choose a posture from obligations, not fashion

Start with the workload boundary: what may leave, what must remain available, who may administer it, which rights are required, and how failure must behave.

Managed Fabric is the shortest route when the organisation can use the described service boundary and wants Dweve to operate the platform infrastructure through its public Mesh. A licensed deployment is appropriate when infrastructure, keys, network, release promotion, or continuity must sit under customer control. Air-gapped licensed operation adds the requirement that installation, licence validation, work, evidence, verification, and recovery continue without outbound connectivity.

None of the postures answers the governance question by itself. A customer can run a poorly governed system inside its own perimeter, or a carefully governed one through a service. The posture determines where technical and operational controls can be exercised. Purpose, authority, source quality, decision policy, human review, legal basis, retention, and affected-person rights still need explicit design.

A mixed estate is possible in principle: work through managed Fabric, embedded calls through its business API or Agent SDK, and directly operated products in a licensed environment for restricted workloads. That creates a new obligation: work identities, source versions, policies, and evidence references must remain intelligible across the boundaries. Do not assume that “same platform” removes integration and transfer design.

QUESTION	BUSINESS WORKSPACE	LICENSED	AIR GAP
Who runs infrastructure?	Dweve	Customer	Customer
Outbound operation?	Expected	Customer-defined	Not required
Key custody?	Service controls	Customer	Customer, local
Release promotion?	Dweve	Customer process	Controlled transfer
Best starting test	Bounded workflow	Install + restore	Network absent
Governance owner?	Customer	Customer	Customer

A directional comparison derived from the deployment page. Contract, architecture, and threat-model review must settle the exact boundary.

DECISION RULE

Pick the least operationally complex posture that can demonstrably satisfy the workload’s real control, continuity, and rights requirements.

MANAGED OPERATION

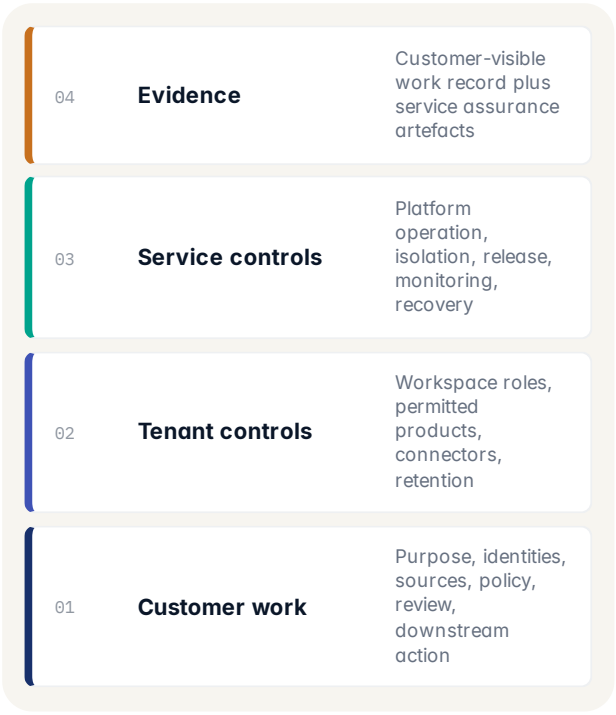
Managed Fabric: define the service boundary precisely

Participation in Dweve's public Mesh is the starting property. The operating model still needs identities, data paths, administrative boundaries, retention, support access, and export procedures.

Document every path by which information enters the service: browser use, API calls, file upload, connectors, knowledge ingestion, administrator configuration, model endpoints, and support workflows. For each path, name the controller, the initiating identity, the permitted data classes, the validation step, and the retained record. Then do the same for outputs, callbacks, downloads, tools, and exports.

Separate tenant and workspace administration from platform administration. Customer administrators should have a defined scope over participants, roles, sources, models, agents, flows, and retention. Dweve operational access should be constrained by the service design and support process. The handbook does not invent those controls; the production service description, security materials, and contract must evidence them.

Before production, exercise deletion, participant removal, credential rotation, data export, incident escalation, and service exit. The purpose is not paperwork. It is to discover whether an operational question can be answered without opening an emergency ticket and whether the organisation can retrieve the records it is responsible for retaining.



A boundary model to validate against the current service description, not a claim about controls the website does not specify.

EVIDENCE TO REQUEST

Current service scope, data-flow diagram, identity model, administrative-access process, subprocessor position, retention behaviour, incident route, backup/restore commitment, export format, and exit procedure.

CUSTOMER-OPERATED PLATFORM

Licensed: customer control includes customer toil

The value of licensed operation is real control over infrastructure and keys. Its cost is an operating system of people, procedures, capacity, releases, and evidence.

Treat the licensed platform as a production service, not a software delivery. Name an accountable service owner, platform operators, security owner, identity owner, data and knowledge owners, product or workflow owners, evidence custodian, change authority, and incident lead. Decide which duties can be combined and which require separation. Record who can grant an agent capability, approve a source, promote a policy, access a trace, rotate a key, or restore a backup.

Build an environment specification that pins the supplied artefacts and all customer dependencies: compute, accelerators where used, operating platform, storage, databases, object stores, queues, clocks, certificate infrastructure, identity provider, secret store, observability, backup targets, and model packages. The site's same-platform statement does not imply every customer topology behaves identically.

Capacity planning should use the chosen workflow corpus. Measure concurrency, queue depth, model and tool latency, graph size, evidence volume, retention growth, restore duration, and degraded behaviour. Generic benchmark numbers are poor substitutes because the dominant cost can move between cognition, tools, knowledge, evidence, and customer systems.

OPERATING OWNERSHIP	licensed-production	
SERVICE	named accountable owner	ASSIGN
PLATFORM	install, monitor, recover	ASSIGN
SECURITY	keys, network, access	ASSIGN
KNOWLEDGE	sources, versions, quality	ASSIGN
AUTHORITY	policy and review gates	ASSIGN
EVIDENCE	retention + disclosure	ASSIGN
CHANGE	test, promote, roll back	ASSIGN

An unassigned line is an operational dependency, not a future detail.

LICENCE BOUNDARY

Included use, product rights, source rights, support, update access, environments, and redistribution are commercial terms. Confirm them in the current quote and agreement; do not infer them from a deployment diagram.

DISCONNECTED OPERATIONS

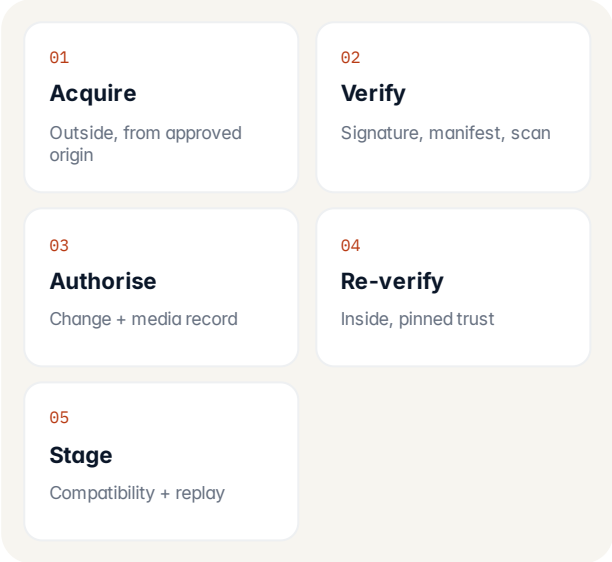
Air gap: design both sides of the transfer boundary

The inside environment cannot depend on an outside control plane. The outside staging area cannot be allowed to smuggle trust across the boundary.

Maintain two explicit zones. The external acquisition zone receives releases, model packages, vulnerability information, licence artefacts, and documentation. It verifies the publisher and manifest, scans or inspects the package, records approval, and prepares controlled media. The internal intake zone treats the media as untrusted until it independently verifies identity, signature, hashes, contents, compatibility, and approval reference.

The reverse path requires equal care. Operational evidence, diagnostics, or support bundles leaving the enclave should be minimised, reviewed, authorised, and recorded. Prefer a reproducible technical description over raw sensitive material. If an artefact must leave, the process should say who may approve it, how it is protected, what receiving party may do with it, and how the transfer is reconciled.

Time is a hidden dependency. Signed artefacts, certificate validation, event ordering, retention, and investigation need a trustworthy local time source and a documented approach to drift. Identity and licence validation also need local continuity. Test restart after long disconnection and after a clock or certificate event, not only on installation day.



Inbound path. The outbound support path requires a separate, least-data flow in the opposite direction.

- **Prove cold start:** reboot every dependency with all external links physically absent.
- **Prove licence continuity:** cross normal validation and renewal boundaries using the documented offline mechanism.
- **Prove recovery:** restore from local backups into an isolated clean environment.
- **Prove support:** diagnose a seeded fault without uncontrolled data export.
- **Prove removal:** decommission an environment and account for keys, media, backups, and retained evidence.

CUSTODY MODEL

Keys, identities, and authority are different control planes

Putting a cryptographic key under customer custody does not automatically put user identity, agent authority, policy approval, or evidence disclosure under equivalent control.

Map at least four identities. A human identity authenticates a person and carries organisational roles. A service identity allows one platform component or customer application to call another. An agent identity binds a delegated work actor to a capability and lifecycle. A signing identity attests to an artefact, release, work state, or proof root. They may share infrastructure, but they should not be collapsed conceptually.

For every privileged operation, identify the actor, credential, authority source, approval condition, scope, duration, revocation path, and evidence. This includes registering a model, adding a source, changing a policy, granting a tool, creating a team, releasing a workflow, viewing sensitive traces, exporting evidence, rotating trust roots, and promoting a platform update.

Key custody also needs lifecycle detail: generation, backup or escrow policy, activation, rotation, compromise response, retirement, and verification of historic records after rotation. A system that can sign new work but can no longer verify retained work has broken its accountability promise.

PLANE	ANSWERS	TYPICAL RECORD
Human identity	Who is the person?	Authentication + role
Service identity	Which component called?	Credential + endpoint
Agent identity	What was delegated?	Capability + lifecycle
Policy authority	Who may decide?	Rule + approval state
Signing identity	Who attested?	Key id + signature
Disclosure authority	Who may inspect?	Purpose + access event

One directory can support several planes. The authority semantics still need to remain visible.

SEPARATION TEST

A platform administrator should not silently become a policy owner, case reviewer, evidence examiner, and signing authority merely because they can operate the infrastructure.

OBSERVABILITY AND EVIDENCE

Logs tell operators what happened; evidence tells examiners why

The two records overlap, but they serve different questions, audiences, sensitivity levels, and retention duties.

Operational telemetry supports availability, performance, capacity, threat detection, and incident response. It records service events such as request timing, errors, queue depth, resource use, component health, administrative actions, and network behaviour. It should minimise payload content and still carry stable identifiers that can be reconciled with work records when authorised.

Work evidence supports provenance, authority, explanation, replay, contestability, and audit. It records source and rule versions, task and agent states, relevant model or solver manifests, tool approvals and results, human gates, outputs, and integrity references. It may be far more sensitive than a metric or service log and should have purpose-specific access.

A trace identifier connects the views; it must not become permission to dump the whole evidence packet into general logging. Define who can pivot from an alert to a work record, under which incident purpose, and what that access itself records. Define the reverse path too: an examiner reviewing a case may need deployment and release evidence without gaining broad infrastructure access.

OPERATE

Telemetry

Health, latency, error, capacity, security signal, and administrator event.

EXPLAIN

Work record

Sources, versions, reasoning artefacts, tasks, authority, output, and state.

VERIFY

Proof material

Commitments, signatures, verifier dependencies, and supported replay artefacts.

GOVERN

Access record

Purpose, actor, scope, disclosure, export, correction, and deletion event.

DESIGN REQUIREMENT

Make correlation possible without making sensitive evidence ubiquitous. Stable identifiers and deliberate access beat payload-rich logs.

CHANGE CONTROL

Updates are evidence-bearing changes

A release can alter behaviour, compatibility, security, replay, and the meaning of a retained record. Promotion therefore needs more than a successful installation.

Inventory the complete change set: platform binaries, product components, schemas, policies, connectors, model packages, solver code, knowledge indexes, trust roots, and infrastructure dependencies. Classify which items come from Dweve and which are customer-controlled. A platform upgrade that leaves a policy or connector incompatible is still a failed change.

A staging corpus should include normal workflows, boundary values, denied actions, conflicting sources, unavailable dependencies, human escalations, evidence verification, and at least one historic replay. Compare results according to the workload's determinism mode. Where variation is expected, define acceptable invariants before running the test, not after seeing the output.

Retain the signed release identity, dependency manifest, approval, test results, migration record, promotion time, operator, and rollback decision. If a migration changes stored evidence or indexes, keep the mapping needed to interpret historic work. Never make a rollback plan that depends on deleting the new state and hoping the old binaries understand what remains.

01 **Acquire and attest**
Verify origin, manifest, rights, and support status.

02 **Read the delta**
Map security, schema, behaviour, interface, and operating changes.

03 **Stage with a corpus**
Run success, refusal, failure, replay, and recovery cases.

04 **Approve the boundary**
Security, workflow, evidence, and operations owners sign their part.

05 **Promote and observe**
Use a reversible window, reconcile health and work outcomes.

06 **Close or roll back**
Preserve the evidence for either decision.

REPLAY IS A RELEASE TEST

A new version should still verify the historic artefacts it promises to support. If it cannot, the compatibility boundary must be explicit before promotion.

CONTINUITY AND RECOVERY

Backup restores service; preservation restores accountability

A conventional backup can return databases and files while leaving the organisation unable to verify, explain, or replay historical work.

Group recovery material by dependency. Service state includes configuration, identities and role mappings, workflow definitions, queues, and operational databases. Knowledge state includes source snapshots where retained, atoms, indexes, lineage, quality state, and policy bindings. Evidence state includes work manifests, referenced artefacts, signatures, roots, access state, and retention metadata. Verification state includes schemas, public keys, verifiers, supported runtimes, and version manifests.

Set recovery objectives per workload, not only per component. The authoritative customer system may recover before the agent work record, creating a period in which decisions exist but cannot be explained. Conversely, restoring a queue twice can repeat an external action. The workflow contract needs idempotency, reconciliation, and a known recovery order.

Run a clean-room restore. Start with empty replacement infrastructure and no undocumented operator state. Restore the selected production snapshot, re-establish trust, process a safe test case, retrieve a historical case, verify its proof material, reconcile it with the authoritative system, and record elapsed time and gaps. A green backup job is not this test.

RECOVERY EXERCISE	quarterly-clean-room	
SERVICE	**configuration + identity**	RESTORE
KNOWLEDGE	**sources + lineage + index**	RESTORE
EVIDENCE	**records + commitments**	RESTORE
VERIFIER	**keys + runtime + schemas**	RESTORE
WORKFLOW	**new bounded case**	EXECUTE
HISTORY	**old case + replay**	VERIFY
SYSTEMS	**authoritative reconciliation**	CLOSE

Measure achieved recovery, data loss, verification success, and manual exceptions.

RETENTION TENSION

Backup, deletion, legal hold, correction, and proof preservation can pull in different directions. Resolve that policy explicitly; the platform cannot infer it.

PORTABILITY AND EXIT

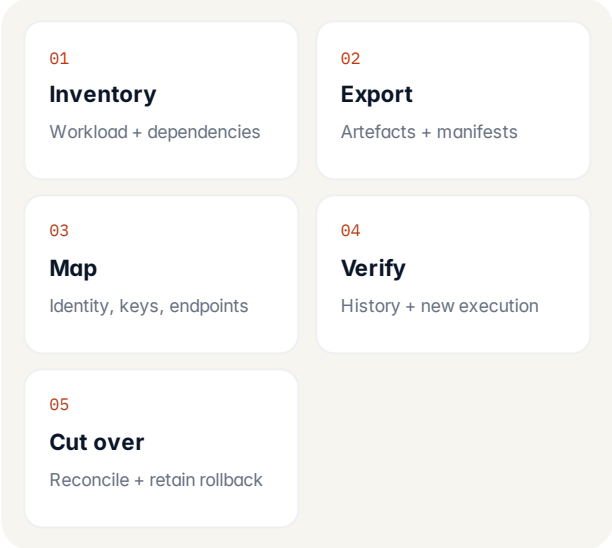
Moving posture should not mean rebuilding the workload

The deployment page says organisations can move between managed public-Mesh participation and licensed operation without replatforming. Treat that as a portability property to prove against a selected workflow.

Define what "the workload" contains before testing movement: source references and retained snapshots, policy and rule versions, product configuration, workflow graph, agent and tool contracts, identities and role mappings, models and solvers, evidence schema, historic records, and external-system bindings. Some elements can move unchanged; others need environment-specific credentials, endpoints, or custody.

A credible migration performs export, validation, transfer, import, dependency remapping, identity reconciliation, secret re-establishment, historical-record verification, a shadow run, cutover, and rollback. It preserves work identities and source/version meaning. It also explains which operational history stays with the former posture and how an examiner can interpret records that span both.

Exit is the same discipline without a destination chosen by Dweve. The organisation should be able to retrieve its governed artefacts in documented forms, validate completeness, preserve supported verification, revoke access, settle retained copies, and demonstrate that critical workflows can continue through an agreed alternative. Exact export capabilities and obligations require current technical and contractual confirmation.



The same logical workflow can cross a posture boundary while environment bindings change explicitly.

ACCEPTANCE TEST

Choose one representative historic case and one new case. After migration, retrieve and verify the historic record, execute the new case, and reconcile both with their authoritative system entries.

04

ADOPTION

Start with one workflow the organisation can judge

A platform adoption becomes credible when a real source, policy, authority path, result, and evidence packet work together. The first workflow should teach the operating model, not maximise surface area.

Adoption is a sequence of evidence, not an estate-wide promise made at kickoff.

USE-CASE SELECTION

Pick a first workflow with a visible boundary

The best first workflow is valuable enough to matter, bounded enough to understand, and rich enough to exercise source, authority, failure, and evidence.

Write the workflow in operational language: a named actor receives a named trigger, uses named sources and rules, produces a named result, passes a named authority gate, changes a named system or hands off a named artefact, and retains a named record. If this sentence cannot be completed, the scope is not ready for product selection.

Prefer a repeated workflow with known examples, an identifiable owner, accessible source material, measurable handling cost, and a reversible or reviewable first action. Avoid “answer any employee question”, “automate operations”, or “build an agent platform”. They hide incompatible purposes and make source quality, permissions, and success impossible to judge.

Consequence determines the proof burden. A drafting assistant can begin with human approval and no direct system action. A workflow that recommends an eligibility outcome needs policy lineage, controlled features, explanation, reviewer authority, contestability, and stronger evidence. A workflow that executes payments or changes access needs explicit tool constraints, separation of duties, idempotency, and recovery.

TRIGGER

One named event

Not a universal assistant prompt.

OWNER

One accountable role

Able to decide source, rule, and acceptance.

RESULT

Observable state

A reviewable artefact or reconciled action.

CORPUS

Known cases

Normal, boundary, denial, conflict, and failure.

GOOD FIRST-WORKFLOW TEST

Can a domain owner explain ten representative cases, the permitted source set, the reviewer’s decision, the downstream state, and what evidence would resolve a dispute?

CURRENT-STATE DISCOVERY

Map today's work before designing tomorrow's graph

Automation built from an idealised procedure will fail on the undocumented hand-offs, exceptions, local data, and authority that keep the real process running.

Observe cases from trigger to closure. Record the queue, actors, systems, documents, communications, lookups, calculations, judgements, approvals, rework, wait states, and final updates. Mark where people copy information, reinterpret policy, use private notes, phone an expert, or bypass a system because the official path does not fit. Those are not annoyances to remove blindly; they are evidence about missing contracts.

Separate deterministic transformations from interpretive work and discretionary authority. Extraction, validation, calculation, search, synthesis, recommendation, approval, and execution have different failure modes. This separation helps decide when Loom reasoning is useful, when Core or a solver should own exact work, when Nexus should coordinate tasks, and where a human gate must remain.

Collect the existing audit trail and compare it with the questions stakeholders actually ask after a problem: Which source applied? What did the worker see? Who changed the case? Which calculation ran? Why was it escalated? Who approved? Did the customer receive the result? The gap becomes the evidence design.

MAP	CAPTURE	WHY IT MATTERS
Work	Steps, queues, waits, rework	Flow and team shape
Knowledge	Sources, versions, local notes	Spindle and Loom inputs
Exact logic	Rules, calculations, thresholds	Solver/Core boundary
Authority	Roles, approvals, overrides	Policy and gates
Systems	Reads, writes, side effects	Tool contracts
Record	Logs, reasons, correspondence	Evidence packet

The map is a design input. Preserve disagreement and exceptions instead of smoothing them away.

DISCOVERY OUTPUT

Parallel sessions can both be individually correct and collectively destructive if they edit the same file. The cross-agent work ledger tracks file-level claims, and dead-process claims can be reaped. Work can coexist.

PLATFORM COMPOSITION

Use the smallest coherent product set

The products separate responsibilities. Adoption should add a product because its responsibility is needed, not because the portfolio exists.

Begin from the surface and the hard requirement. Use Fabric when people need the workspace. Add Loom when the workflow needs source-aware cognition, memory, search, reasoning, experts, or solver orchestration. Add Nexus when work must be split into governed agents, tasks, gates, retries, and hand-offs. Add Spindle when raw sources need durable processing, structure, lineage, quality, or bias/compliance stages.

Use Aura when the object being governed is software development through goals, plans, tools, models, quality loops, and audit. Use Mesh when approved work must use distributed capacity while preserving local boundaries, proposal lifecycle, receipts, and settlement. Use Charter when a domain needs its party spine, source tree, engines, workflows, events, packs, and organisational ownership.

Core and Kera sit deeper in the implementation story. Core is for binary and numeric operations, runtime modes, backend execution, determinism, and verification. Kera is the graph language, compiler, type and effect system, planner, lowering path, and target emission described in the published material. Treat them as explicit technical dependencies where their responsibilities are required, not as UI tiles the business must adopt separately.

05	**Substrate**	Shared contracts, trace, Core/Kera where applicable
04	**Knowledge**	Spindle where sources become governed reusable material
03	**Work**	Nexus where agents, tasks, and gates coordinate
02	**Cognition**	Loom is not a model router with a larger catalogue.
01	**Surface**	Fabric or an existing customer application

A common workflow-shaped composition. Aura, Mesh, and Charter enter when their distinct responsibilities are the work.

ARCHITECTURE TEST

For every included product, name the state it owns, the contract it exposes, the failure it contains, and the evidence it contributes. Remove any product without a clear answer.

KNOWLEDGE READINESS

Prepare sources as governed inputs, not uploads

A document becomes usable only when the organisation can identify it, scope it, version it, interpret its authority, and recognise when it conflicts or expires.

A personal node is not a miniature data centre; it has a human owner who may return without warning. Fabric is the user facing control surface: it determines what capacity the person has chosen to offer, and Mesh sees only the resources currently announced as available. The governing rule is simple: local use always has priority over contributed work.

Decide what may be transformed. Parsing, chunking, atomisation, enrichment, hierarchy, indexing, and summary can make material more useful, but each transformation can lose context or introduce an interpretation. Preserve lineage back to the retained source and version. Define which transformed artefact may support retrieval, which may support recommendation, and which must never substitute for authoritative wording.

Design conflict behaviour with the domain owner. Newer may supersede older, a central rule may outrank local guidance, two jurisdictions may both apply, or ambiguity may require a human owner. Loom can preserve and reason over conflict; it should not invent the authority rule. Spindle can record stages and quality; it cannot declare a source legitimate without organisational criteria.

verification records	policy-credit-eligibility	
OWNER	credit policy committee	NAMED
VERSION	2026-03 · effective 01 Apr	PINNED
AUTHORITY	approved operating policy	SCOPED
LINEAGE	original → atoms → index	LINKED
CONFLICT	regional exception wins	RULED
REVIEW	quarterly or event-driven	OWNED
WITHDRAW	quarantine + dependent cases	PLANNED

Illustrative source governance fields, not a current product schema.

READINESS GATE

Do not release a workflow whose decisive sources have no owner, effective-version rule, conflict behaviour, and route from output back to the authoritative material.

GOVERNANCE DESIGN

Turn policy and authority into executable boundaries

A policy paragraph is not yet a gate. A job title is not yet authority. Both need scoped, versioned conditions that the workflow and reviewer can see.

For each decision or action, write the eligibility condition, required evidence, allowed result states, permitted actor, monetary or risk limits, separation-of-duties rule, review or approval requirement, escalation owner, expiry, and exception route. Bind the rule version to the work record. If the organisation cannot express a condition, classify the step as judgement and preserve the responsible human role.

Give agents capabilities, not ambient trust. A research agent may retrieve from approved sources and draft a structured finding. It may not approve a case, register a new source, modify policy, or write to the authoritative system. An execution agent may call a narrowly scoped tool only after the required review state exists. Its identity, input, action, result, and receipt remain attached to the work.

Human review needs a meaningful interface. Show the material facts, source and version, confidence or uncertainty where appropriate, conflicting evidence, relevant rule, proposed result, permitted reviewer actions, and effect of each action. Record approval, rejection, correction, request for more evidence, or escalation as distinct states. A single “approve” button beneath an opaque answer is ceremonial review.

STATE	WHO MAY ENTER IT?	REQUIRED EVIDENCE	NEXT
Draft	Assigned worker/agent	Sources + work id	Validate
Ready for review	Workflow gate	Rule + proposed result	Officer
Approved	Authorised officer	Reason + identity	Execute
Rejected	Authorised officer	Reason + correction	Close/rework
Escalated	Rule or officer	Conflict + owner	Specialist
Executed	Scoped tool identity	Approved state + receipt	Reconcile

Illustrative decision-state machine. The domain owner defines the actual authority model.

CONTROL TEST

Attempt to skip each transition, reuse an expired approval, exceed a limit, change the source after review, and execute twice. The system should refuse or surface the inconsistency.

SYSTEMS INTEGRATION

Design the customer integration as a durable contract

The platform can govern its own work and still create an ungoverned outcome if the surrounding application loses identity, authority, version, or reconciliation state.

Define an inbound contract with a stable work identifier, initiating identity and purpose, minimal data or source references, event time, expected capability, authority ceiling, policy selection rule, deadline, correlation identifier, and duplicate behaviour. Validate required fields and reject ambiguous states. Do not silently invent a policy version or treat a missing identity as a system user.

Define an outbound contract with result type, work state, human-review state, material reasons, source references, warnings, action eligibility, evidence reference, error classification, and retry semantics. A natural-language answer alone is a poor operational interface. The consuming system should be able to distinguish “no eligible result”, “insufficient evidence”, “denied by policy”, “awaiting human”, “dependency failed”, and “completed”.

Reconciliation closes the loop. Record the external action receipt or authoritative-system version alongside the platform work. Poll or subscribe for uncertain outcomes. Detect duplicates and conflicting updates. Provide an operator queue for states that cannot be reconciled automatically. A successful tool call is not equivalent to an accepted business transaction.

ILLUSTRATIVE RESULT ENVELOPE

```
work_id        case-4f9e
state          approved_for_action
authority      officer:217 · 2026-07-19T09:41Z
result_type    eligibility_decision
result         declined
reasons        affordability_rule, evidence_gap
sources        policy@2026-03, snapshot@sha256:…
action         notice.create · permitted
evidence_ref   trace:01J…
reconcile      pending
```

The field names are illustrative. The design requirement is explicit state, authority, provenance, action eligibility, and reconciliation.

CONTRACT OWNERSHIP

Version the customer interface independently from prompts and user-interface copy. Test backward compatibility, unknown fields, missing dependencies, retries, and duplicate delivery.

EVALUATION

Build an acceptance corpus before a polished demo

A governed system is judged on representative cases, boundaries, refusals, failures, and recovery, not on the one prompt chosen because it works.

Sample historic cases across the real distribution, then deliberately add rare and difficult conditions. Include clean normals, boundary values, incomplete information, conflicting sources, policy transitions, source withdrawal, unusual language or formats, unsupported requests, attempted permission escalation, tool timeout, reviewer disagreement, duplicate events, and downstream rejection.

For every case, define expected invariants rather than forcing a single prose string. The correct source and rule version must be selected; forbidden data must remain unavailable; deterministic calculations must match; the authority ceiling must hold; required warnings must appear; unsupported action must be refused; human state must be preserved; and the evidence packet must contain the declared artefacts.

Keep the corpus versioned and protected. Record why each case exists, which risk or requirement it tests, its permitted use, expected outcome class, reviewer, and last approval. Add production incidents and accepted edge cases after minimisation. Separate training, configuration, evaluation, and demonstration material so that success is not manufactured by leakage.

BASELINE

Normal work

Representative volume, formats, roles, and expected outcomes.

BOUNDARY

Near the line

Thresholds, ambiguity, missing facts, policy transition, and conflict.

ABUSE

Disallowed work

Prompt injection, source escape, privilege escalation, and unsafe tool intent.

RESILIENCE

Failure and recovery

Unavailable source, model, tool, reviewer, target, replay, and restore.

RELEASE GATE

A demo proves a possible happy path. A controlled corpus provides repeatable evidence about the workflow boundary. Require the second before production.

RELEASE STAGES

Move through shadow, assistance, and bounded action

Each stage changes who can see the result, who holds authority, what can act, and how quickly a mistake can propagate.

In shadow mode, the workflow receives a controlled copy of real triggers and produces no operational output. Compare source selection, task state, result class, evidence, latency, and failure with the current process. Resolve data-handling and worker-notice requirements before using production material. Shadow mode is useful only if someone reviews the disagreements.

In assistance mode, an authorised worker sees a draft, recommendation, retrieved material, or proposed plan and remains the only actor who can commit the result. Measure accepted, corrected, rejected, and escalated work, not only usage. Inspect whether workers understand the source and limitation, whether review becomes mechanical, and whether corrections feed governance rather than hidden prompt changes.

Bounded action permits a narrow side effect after defined automated and human gates. Begin with reversible, low-blast-radius tools, explicit limits, idempotency, and reconciliation. Keep an emergency stop and manual path. Broader autonomy is a new risk decision requiring evidence from the previous boundary; it is not the default reward for passing a time period.

01 **Offline corpus**
Validate contracts, sources, policy, evidence, refusal, and replay.

02 **Shadow**
Observe real distribution with no operational output.

03 **Assist**
Human owns every committed result and correction.

04 **Bounded action**
Narrow tool, explicit gate, limit, receipt, and reconciliation.

05 **Expand or hold**
Risk owner decides from evidence, incidents, and operating capacity.

ROLLBACK MEANS MORE THAN DISABLE

Preserve queued work, prevent duplicate action, reconcile partial external effects, retain the record, notify operators, and restore the manual route.

VALUE AND CONTROL METRICS

Measure the work system, not token theatre

Usage and response speed can be useful operating signals. They are not proof of better decisions, safer work, or lower total cost.

Establish a baseline from the current process: demand, handling time, wait time, first-time completion, rework, escalation, defects, complaints, exception inventory, cost per completed case, and effort to assemble evidence. Segment by case type and consequence. An average can improve while difficult or disadvantaged cases become worse.

Measure workflow outcomes after release: completion by outcome class, worker acceptance and correction, source and policy mismatch, denied actions, human-gate time, tool success and reconciliation, incident severity, proof verification, recovery time, and evidence retrieval effort. Pair productivity with quality and control. Faster throughput with more hidden correction is not a gain.

Interpret cause cautiously. Policy changes, demand mix, staffing, seasonality, source quality, customer-system releases, or operating practice can move the metric. Use staged comparison, annotated change history, qualitative case review, and counter-metrics. Do not claim that the platform caused an improvement the evaluation design cannot isolate.

DIMENSION	USEFUL MEASURE	COUNTER-MEASURE
Flow	Lead + handling time	Rework + queue transfer
Quality	First-time correct	Severity-weighted defect
Authority	Gate completion	Override + rubber-stamp rate
Safety	Denied unsafe action	False denial + workaround
Evidence	Packet retrieval time	Missing/unverifiable item
Cost	Cost per closed case	Operating + correction cost

Measure a balanced system. Targets and thresholds belong to the workflow owner and risk model.

NO INVENTED ROI

Build the business case from measured baseline volume and cost, explicit implementation and operating effort, conservative benefit ranges, and the cost of risk, not brochure percentages.

STEADY-STATE OWNERSHIP

Operate the workflow as a governed product

After release, sources change, policies change, models change, systems fail, people learn workarounds, and the case mix moves. Someone must own the whole result.

The workflow owner is accountable for purpose, boundary, outcomes, stakeholder experience, and continued fitness. Source owners approve and maintain material. Policy owners own rules and exceptions. Technical owners maintain product configuration and integrations. Security owns threat controls and response. Operations owns service health and recovery. Evidence or assurance owners maintain retrieval, verification, and disclosure. Human managers own staffing, competence, and meaningful review.

Create change triggers rather than relying on an annual review: new or withdrawn source, policy revision, model or platform release, tool interface change, new data class, changed consequence, drift in outcome distribution, repeated correction, significant denial, incident, complaint, audit finding, law or regulatory change, and deployment migration. Each trigger has an owner and a required re-evaluation scope.

Hold a regular case review that joins operational and governance views. Sample successful, corrected, denied, escalated, and failed work. Inspect the source and rule, agent and human state, external result, evidence packet, and user impact. Decide whether the case is an isolated error, a corpus addition, a source problem, a policy ambiguity, a training issue, a contract defect, or a reason to narrow the workflow.

MONTHLY CONTROL REVIEW	workflow-credit-explain	
OUTCOME	**flow + quality + impact**	REVIEW
SOURCES	**changes + conflicts + expiry**	REVIEW
POLICY	**versions + overrides**	REVIEW
AUTHORITY	**gates + escalation + access**	REVIEW
SYSTEMS	**tools + reconciliation**	REVIEW
EVIDENCE	**sample retrieval + verify**	REVIEW
CHANGE	**decisions + owners + dates**	RECORD

The cadence and sample size follow consequence, volume, and observed change.

OWNERSHIP RULE

No platform metric, security control, or model evaluation substitutes for an accountable domain owner who can stop or narrow the workflow.

SCALE GATES

Expand only when the boundary is still explainable

More users, sources, tools, autonomy, domains, or deployments change the risk and operating model. Treat each as a deliberate boundary change.

A scale proposal should say exactly what changes: volume, user population, jurisdiction, language, data sensitivity, source authority, model, tool, action limit, reviewer role, workflow branch, product component, deployment, or retention. Re-run the requirements and corpus that the change can affect. A successful pilot does not validate a materially different workload.

Require evidence that current operation is stable: owners are active, source and policy updates are timely, corrections are understood, denial and escalation work, incidents close, external actions reconcile, historic cases verify, recovery has been exercised, and support capacity matches demand. Expansion built on unresolved exceptions turns local knowledge into systemic fragility.

Preserve reversibility. Use segmented release, scoped identities, feature or authority gates, queue limits, side-effect ceilings, and an alternative route. Define stop conditions in advance: missing source, policy ambiguity, severe defect, evidence gap, reconciliation failure, control bypass, unexpected impact, or inability to support the workload.

EXPANSION	RE-OPEN	REQUIRED PROOF
More volume	Capacity + recovery	Load, queue, restore
More sources	Authority + conflict	Lineage, quality, version
New tool	Permission + side effect	Denial, receipt, reconcile
More autonomy	Human authority + limits	Boundary and abuse corpus
New domain	Purpose + policy + owner	Fresh workflow acceptance
New posture	Custody + operation	Install, isolate, migrate

Scaling a dimension reopens the controls coupled to that dimension.

HEALTHY OUTCOME

Holding, narrowing, or reversing an expansion is evidence that governance works. Growth is not the only successful decision.

05

DECISION AND VERIFICATION

Turn the brochure back into questions and evidence

The last chapter is a working diligence pack: what to decide, what to ask Dweve to provide, what to demonstrate in the chosen environment, and what this publication deliberately does not claim.

DECISION RECORD		selected-workflow
PURPOSE	**bounded and owned**	DECIDE
PRODUCTS	**minimum coherent set**	DECIDE
POSTURE	**responsibility accepted**	DECIDE
PROOF	**customer corpus passed**	VERIFY
OPERATION	**owners + runbooks ready**	VERIFY

A procurement decision should retain the evidence and assumptions behind every line.

DECISION WORKSHEET

The platform decision on one page

Complete this with domain, security, architecture, operations, assurance, legal/procurement, and the people who perform or receive the work.

Work and consequence

- What exact trigger, actor, source set, decision or deliverable, authority gate, downstream state, and record define the workflow?
- Who can be affected, what can go wrong, which outcomes are irreversible, and which manual route remains?
- Which cases, languages, jurisdictions, data classes, tools, and actions are explicitly outside scope?

Platform composition

- Is the surface Fabric or an existing application through API?
- Which product owns cognition, organised work, knowledge processing, development, distributed capacity, domain state, exact compute, or compilation?
- Can each included dependency state its contract, failure, owner, and evidence contribution?

Deployment

- Which custody, connectivity, continuity, operational-control, and licence-right requirements determine the posture?
- Who accepts infrastructure, key, identity, change, monitoring, backup, recovery, and support duties?

Governance and proof

- Who owns source authority, policy versions, agent capabilities, human review, tool permissions, evidence access, retention, and contestability?
- Which artefacts make the result explainable, verifiable, and reconcilable with the authoritative system?
- Which claim needs architecture inspection, a measured test, security evidence, commercial confirmation, or legal assessment?

Release and operation

- Does the corpus include boundary, refusal, abuse, failure, recovery, and historic replay cases?
- What changes at shadow, assistance, and bounded-action stages; who can stop each stage?
- Which metrics, counter-metrics, case reviews, change triggers, recovery tests, and expansion gates continue after release?

DECISION QUALITY

A “yes” should cite a named artefact, result, owner, or accepted contractual term. An unanswered line is an assumption to resolve, not a reason to write “TBD” and proceed.

TECHNICAL DILIGENCE

Ask Dweve to demonstrate the platform as a system

The demonstration should follow one representative workload across products and boundaries. Feature tours are useful only after the end-to-end contract is visible.

Begin from a customer-supplied trigger, representative source set, policy or rule version, and expected authority boundary. Show the work identity entering Fabric or the API, source/version binding, context projection, Loom or solver artefacts, Nexus task states where used, tool permission, human review, final result, authoritative-system receipt, evidence packet, and verification or replay path.

Seed failure deliberately: remove a source, introduce a conflict, deny a tool, time out a model, submit an invalid agent artefact, make the reviewer unavailable, reject the downstream write, and alter a retained artefact. Show the configured response and the record for each. Ask which behaviour is product mechanism, which is customer configuration, and which requires custom integration.

It keeps one description of the work and prepares a suitable plan for the machine available. The plan changes, but the job does not quietly become a different job. This helps Dweve bring the same underlying products to more kinds of hardware without rebuilding their foundations each time. Support can vary by product and target.

01 **Trace one success**
Identity → source → cognition → work → authority → action → packet.

02 **Force one refusal**
Request a capability or source outside the declared boundary.

03 **Force one failure**
Break a dependency and inspect containment and recovery.

04 **Verify one history**
Retrieve a retained case and run the supported verification path.

05 **Move one boundary**
Change version, identity, or posture and show explicit remapping.

06 **Export one record**
Give an authorised examiner the minimum coherent evidence.

KEEP THE RAW RESULT

Retain requests, configuration and artefact versions, timing, resource context, outputs, exceptions, and reviewer notes. A sales summary is not the proof record.

SECURITY DILIGENCE

Security questions must attach to a boundary

"Secure" is not a platform-wide observable. Ask how identities, data, models, agents, tools, evidence, administrators, releases, and deployments constrain one another.

Identity, access, and administration

- How are human, service, agent, and signing identities issued, scoped, rotated, revoked, and audited?
- Which privileged roles exist at service, tenant, workspace, product, workflow, source, policy, tool, and evidence levels?
- How are support access, emergency access, separation of duties, session controls, and customer-operated administration handled in each posture?

Data and knowledge

- Where can payloads, embeddings or indexes, caches, prompts, outputs, logs, traces, backups, and support artefacts reside?
- How are connectors authenticated; sources classified, quarantined, withdrawn, and kept out of unauthorised contexts?
- How do deletion, correction, legal hold, retention, export, and proof preservation interact?

Models, agents, and tools

- Which providers and models are permitted, how are manifests pinned, and what data can leave the selected deployment boundary?
- How do agent capabilities, task inputs, sandbox or execution controls, network access, tool arguments, approval gates, limits, receipts, and kill paths work?
- How are prompt injection, malicious sources, poisoned knowledge, unsafe output, privilege escalation, and supply-chain changes tested?

Evidence and platform operation

- What is signed or committed, who holds keys, how are historic records verified after rotation, and who may disclose evidence?
- How are releases built, signed, delivered, scanned, promoted, rolled back, and supported, especially with no outbound connectivity?
- What security assurance, incident process, vulnerability handling, recovery exercise, and customer responsibility material is current?

INTERPRETATION RULE

The website security page supplies the canonical public description. The installed architecture, assurance pack, contract, and customer threat model must settle the precise control.

PROCUREMENT DILIGENCE

Commercial clarity is part of architecture clarity

A technically portable platform can still be operationally or contractually difficult to move. Confirm rights, responsibilities, conditions, and exit artefacts alongside the design.

Identify the selected entry door, products, deployment posture, environments, users or usage basis, implementation scope, support level, update access, term, renewal mechanism, and any proof or pilot conditions. Keep one-time services separate from recurring service and licensed rights. Keep estimated infrastructure and customer operating effort separate from supplier charges.

For licensed use, confirm which binaries and source materials are delivered, the right to install and operate them, included products and environments, usage treatment, affiliate or contractor access, backup and disaster-recovery copies, offline licence mechanism, update and security-fix access, support boundary, and rights at expiry or termination. The website's public claims do not replace the agreement.

For managed use, confirm service boundary, location commitments, availability and support terms, usage measurement, overage or limit behaviour, retention, exports, deletion, incident and change notice, subprocessors where applicable, and exit assistance. For any posture, confirm which evidence artefacts the customer can retrieve and whether historic verification remains supported after exit.

TOPIC	PUBLIC SITE CAN FRAME	AGREEMENT MUST SETTLE
Products	Responsibilities and routes	Included rights + versions
Deployment	Managed/licensed /air gap	Exact boundary + duties
Usage	Public pricing shape	Meter, conditions, limits
Source	Availability statements	Delivered scope + licence
Support	General route	Hours, response, severity, term
Exit	Portability promise	Formats, time, assistance, rights

Commercial due diligence protects the operating model the technical design depends on.

VOLATILE INFORMATION

Pricing, availability, release status, certifications, support terms, and commercial rights can change. Verify the current published description and written offer at the decision date.

WHAT TO COLLECT

The evidence request pack

Request the smallest set that can validate the selected workflow and posture. More documents are not automatically more assurance; every artefact should answer a named question.

Architecture and product

- Current component and dependency map; interface and work-state contracts for the selected path; product and foundation version manifest.
- Source, rule, model, agent, tool, human, result, trace, and replay artefact examples from a non-sensitive representative workflow.
- Deployment topology, data-flow and trust-boundary diagrams, dependency inventory, sizing method, and migration/exit mechanism.

Operation and security

- Identity and privileged-access model; key lifecycle; connector, model, agent, tool, and evidence-access controls.
- Release provenance and update/rollback procedure; monitoring boundary; incident and vulnerability process; backup and clean-room restore evidence.
- For air gap: signed media workflow, offline licence validation, local verification, no-outbound acceptance, local recovery, and support-data transfer procedure.

Quality and assurance

- Benchmark definitions, corpus, hardware, versions, harness, raw outputs, repetitions, comparison boundary, and limitations for any measured claim relied upon.
- Customer-workflow acceptance plan and results covering normal, boundary, refusal, abuse, failure, recovery, historical verification, and reconciliation.
- Current assurance reports, security statements, conformity or certification evidence actually applicable to the chosen service and scope.

Commercial and responsibility

- Current product, service, licence, source, update, support, environment, usage, continuity, export, exit, and retained-verification rights.
- A responsibility matrix covering infrastructure, identities, keys, data, sources, policy, models, agents, tools, evidence, releases, incidents, recovery, and user governance.
- Decision log recording accepted gaps, compensating controls, owners, deadlines, expiry, and conditions for production or expansion.

CLOSE THE LOOP

Every requested item should end in verified, not applicable with reason, accepted risk with owner, contractual commitment, remediation with date, or a decision not to proceed.

PUBLICATION BOUNDARY

What this handbook claims, and what it does not

This is an explanatory synthesis of the current website. It preserves strong public statements while refusing to turn them into unqualified guarantees.

The handbook claims that the public platform story can be understood as one set of bounded responsibilities: people or applications enter through a workspace or API; products own cognition, work, knowledge, development, distributed capacity, domain state, exact compute, and compilation; shared contracts and trace connect the result; deployment moves operational control without intentionally creating a new product architecture.

It also records the website's strong deployment and trace positions, including customer-operated licensed use, air-gapped operation without outbound dependency, offline licence validation, no kill switch for licensed operation, provenance-aware evidence, integrity verification, and supported replay concepts. These statements are framed as acceptance targets because the website alone cannot prove an installed system.

The handbook does not certify security, compliance, legal suitability, fairness, correctness, availability, performance, source completeness, current release status, or commercial entitlement. It does not replace current specifications, agreements, service descriptions, source repositories, benchmark evidence, assurance reports, or workload-specific assessment.

EXPLAINS

Mechanism

Responsibilities, flow, controls, artefacts, posture, and operating questions.

PRESERVES

Qualification

Scope, conditions, customer duties, evidence needed, and source conflicts.

DOES NOT GRANT

Assurance

No certification, legal conclusion, SLA, benchmark reproduction, or fitness warranty.

REQUIRES

Current proof

Installed test, applicable documentation, raw evidence, contract, and owner decision.

CANONICAL-SOURCE RULE

If this edition and the website differ, the current website owns the public statement. If the website and a signed agreement differ, the agreement governs the relationship. Record the version and decision date.

SOURCE HYGIENE

Known ambiguities are part of the diligence record

The current site contains a few portfolio and release-count statements that do not align perfectly across website sources. This edition makes the ambiguity visible instead of silently selecting the most convenient number.

Core is not something you have to install or learn. It stays underneath the Dweve product you are using. The products feel different because they do different jobs. The machine underneath them is the same. Loom uses it when it thinks through a problem. Nexus uses it when a team of specialists works on your task. Spindle uses it when it processes and protects knowledge.

The open-source overview and registry use different counts, while page coverage and the placement of Kera or Jacquard vary across the site. Source release and availability language also needs component-level checking. This handbook does not use a single total as a claim. A dedicated foundations guide should maintain a dated component registry with published page, repository, licence, release, and verification status.

Measured claims elsewhere on the site can use different workloads, baselines, hardware, versions, or definitions. A number belongs to its original website source and method. It should not be transferred into a general platform promise. Pricing, compliance language, and product availability are similarly time-sensitive and should be verified at use.

AMBIGUITY	EDITORIAL TREATMENT	BUYER ACTION
Product count	name current products and components	Confirm licensed scope
Foundation count	Avoid one total	Request dated registry
Release/source status	Qualify per component	Inspect repository/delivery
Benchmark	Keep website source + method	Reproduce on workload
Compliance	Describe mechanism	Obtain scoped assessment
Price/rights	Mark as volatile	Use current written offer

Uncertainty that could change a decision belongs in the decision record.

WHY PUBLISH THIS?

A credible handbook should help a reader find the limits of the public material. Hiding them would make the document longer but less informative.

FIND A SUBJECT

Subject index

Page references point to the substantive explanation, worked example, control, boundary, or decision record, not merely to a passing label.

SUBJECT INDEX CONTINUED

Subject index · E–P

Terms are grouped alphabetically. Consecutive page references are collapsed into ranges.

SUBJECT INDEX CONTINUED

Subject index · P–W

Terms are grouped alphabetically. Consecutive page references are collapsed into ranges.

CLOSING NOTE

A platform earns trust at the joins

The useful question is not whether one model can answer. It is whether an organisation can govern the full movement from source to result and still understand it later.

Dweve's public architecture is strongest when read as a set of separations that reconnect through explicit contracts: source is not context; cognition is not exact calculation; an agent is not authority; a tool call is not a completed business action; a log is not evidence; integrity is not legitimacy; customer control is not customer readiness; and deployment portability is not an excuse to ignore environment bindings.

That model gives an evaluation team something better than a list of features. It gives them a way to draw the workload, remove unnecessary parts, locate the boundaries, decide who owns each state, specify what must fail closed, and ask for a record that an authorised outsider can inspect without reconstructing the story from screenshots and memory.

The next step should therefore be concrete: choose one bounded workflow, bring its real source and policy, define the authority and action ceiling, select the smallest coherent product set and deployment posture, build the acceptance corpus, and require the platform to produce both the result and the evidence needed to judge it.

START HERE	one bounded workflow	
SOURCE	**owned + versioned**	BRING
POLICY	**explicit + scoped**	DEFINE
AUTHORITY	**human + tool ceiling**	ASSIGN
RESULT	**observable + reconciled**	SPECIFY
EVIDENCE	**retrievable + verifiable**	REQUIRE
POSTURE	**responsibility accepted**	SELECT
CORPUS	**normal + failure + abuse**	TEST

Then expand from evidence. Do not begin by declaring a universal transformation programme.

DWEVE.COM

Use the current website to verify product, deployment, security, source, and commercial information for the decision date.