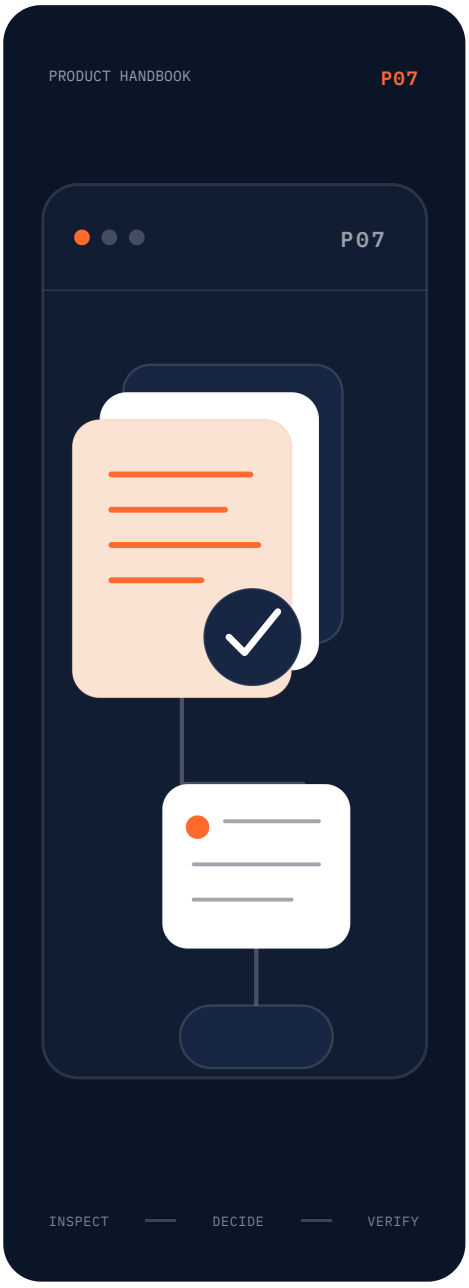
PRODUCT HANDBOOK · P07

The Charter product handbook

Responsibility, state, contracts, a complete workflow, failure, evidence, deployment, security, recovery, and production acceptance for Charter.

CHARTER RESPONSIBILITY WORKFLOW EVIDENCE DEPLOYMENT ACCEPTANCE



Read this when you need Charter explained as a working product boundary, not a list of features.

P07

CONTENTS

Charter, from responsibility to production

The handbook keeps one illustrative scenario, one customer moves from first signal to renewal, throughout the product, technical, governance, and operating explanation.

PRIMARY READERS	READING DEPTH	DECISION THIS SUPPORTS
Product, architecture, delivery and operations teams	Professional + technical	A complete product explanation for teams deciding where Charter fits, how it works, and what they must prove before relying on it.

READING RULE

Product mechanism comes from Charter product page. Illustrative fields and acceptance procedures show what to demand from the current implementation; they are not invented feature claims.

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 · DECISION

Who should read the Charter handbook

This handbook is for product owners, architects, engineers, security and operations teams, reviewers, and buyers deciding whether Charter is the right boundary for a real workflow.

Charter owns the organisation's single governed operating graph. Read the boundary chapter to decide whether that responsibility exists in the target workflow and whether another product already owns it. Read the walkthrough to see the state and evidence that move through one case. Read the final chapter to understand deployment and customer operating duties.

For a decade the default strategy was to buy specialized SaaS and integrate later. Sales owns a CRM, finance owns accounting, delivery owns projects, support owns tickets, HR owns employee records, marketing owns campaigns, knowledge sits in a wiki. The company looks connected on dashboards, but underneath it is a set of vendor ledgers stitched together by APIs, scheduled jobs and manual work. That was already expensive. With AI it becomes the wrong foundation, because a model cannot safely help run a company scattered across vendor boundaries. It can summarize a thread or draft a reply, but it cannot reliably understand the business unless it is grounded in the actual records, permissions, lifecycle states, documents, events and audit trail.

The website is the only factual source for this edition. Current interfaces, packages, availability, source/repository status, benchmark conditions, assurance, commercial rights, support, and contracts must be verified at the decision date. Strong public claims become acceptance targets, not brochure guarantees.

PRODUCT

Charter

the organisation's single governed operating graph

PRIMARY INPUT

Typed work

a business event or user request bound to a party, purpose, domain, current record version, permitted fields, applicable pack/policy, and authority

PRIMARY OUTPUT

Result + state

an updated company record or explicit no-change result with proposal, human authority, event, domain state, and audit linkage

DECISION

Fit + proof

Select, test, operate, and govern the boundary.

PRIMARY ACCEPTANCE

Follow one party from first signal through quote, signed acceptance, invoice/project, support, and renewal; attempt a forbidden field read and expired proposal; fail a provider; replay events; reconcile outbox delivery; run a rights request; and export the owned record.

READ THIS FIRST · TERMS

Vocabulary and source discipline

The handbook separates the Charter product from its lower components, user or API surfaces, deployment posture, commercial rights, and illustrative workflow.

Product responsibility

Charter is the product boundary for the organisation's single governed operating graph. A product can use foundations and other products without absorbing their state or guarantee. The caller consumes a product result; it should not reach into lower implementation internals.

Contract, manifest, and evidence

A contract defines typed meaning at a boundary, including errors and authority. A manifest identifies actual versions and dependencies used. Evidence is a typed artefact that supports a specific question, provenance, state, action, integrity, proof or replay, not a synonym for every log.

Scenario and claim status

API input is EUR 0.20 per 1M tokens, cached input EUR 0.04, and output EUR 0.40. The pre-order rates are EUR 0.15, EUR 0.03, and EUR 0.30. Business Workspace packages start at EUR 299 a month; Independent Builder is the separate one-person route at EUR 99. Third-party model, search, tool, action, and service charges remain separate.

TERM	ANSWERS	DOES NOT ANSWER
Product	Who owns the job/state?	Where or how sold
Surface	How a caller enters?	Who owns lower work
Foundation	Which mechanism is inspectable?	Product availability/right
Posture	Who operates and where?	Workflow suitability
Evidence	Which proposition can be checked?	Legitimacy/correctness by itself
Illustration	How a flow could be evaluated?	Implemented schema/customer result

Here is the whole idea, one simple step at a time, with an everyday example for each one.

KNOWN PORTFOLIO AMBIGUITY

The current published page set exposes the current product set while some overview wording uses a different count or positions Kera underneath the visible product set. This handbook names the selected product description and avoids relying on one disputed total.

01

RESPONSIBILITY AND BOUNDARY

Charter owns one product job and its state

The first chapter defines what enters, what the product owns, what leaves, which dependencies it may use, and which decisions remain outside it.

The handbook's repeated product-boundary model.

BOUNDARY · RESPONSIBILITY

Charter in one accountable sentence

Charter owns the organisation's single governed operating graph. Everything else in the handbook follows from keeping that job and its state visible.

Charter is a single-tenant, AI-native company operating system. Instead of connecting many systems and asking AI to infer the seams, Charter puts every business entity inside one operating graph: leads, companies, contacts, deals, quotes, invoices, subscriptions, projects, tickets, employees, partners, investors, documents, knowledge pages, workflows and AI actions. The result is not an all-in-one app in the usual sense. It is one deployment, one data model, one login, one permission system and one audit trail that the whole business runs on.

The principal input is a business event or user request bound to a party, purpose, domain, current record version, permitted fields, applicable pack/policy, and authority. The principal output is an updated company record or explicit no-change result with proposal, human authority, event, domain state, and audit linkage. Those two sentences define the integration boundary better than a screenshot: a caller must be able to submit a typed, authorised request and distinguish completion, refusal, dependency failure, human wait, and unreconciled external action in the result.

Charter owns the company record and lifecycle. AI proposes inside the permissions of that graph; it does not self-authorise. A single data model does not remove domain ownership, legal duties, external provider constraints, or the need to isolate tenants and deployments.

The Charter boundary expressed as responsibility rather than deployment shape.

DELETION TEST

Remove Charter from the architecture. Name the state and guarantee that disappear; do not silently assign them to a model, workflow, or integration.

BOUNDARY · STATE MODEL

The stable state the product must own

Durable state is the material that must remain interpretable across retries, provider changes, upgrades, deployment moves, incidents, and historical review.

The Charter boundary includes party spine, source tree, domain-engine records, permissions and purpose projection, workflow graph, proposed actions, policy gates, run state, events/outbox, provider registry, country/domain packs, evidence, and lifecycle history. This is a responsibility inventory, not a claim that every field sits in one database or that the published description publishes one schema. Different stores may suit different objects; the product still owns their coherent lifecycle and identity.

Classify each state object as immutable/content-addressed, versioned mutable, append-only event, derived projection or cache, ephemeral attempt state, external reference, or retained evidence. Then name its owner, schema, access, retention, backup, migration, integrity, and invalidation rule. Derived data must point back to its inputs; mutable data needs concurrency and supersession semantics.

A retry should append an attempt or transition beneath the same work identity. An upgrade should migrate state with an explicit old/new version relation. A provider change should update the manifest without changing the business object. An erasure or correction should record its authorised effect without making surviving evidence impossible to interpret.

04	**Evidence**	Events, decisions, receipts, transcripts, proofs, and access state
03	**Versions**	Schema, product, source, policy, model/agent/tool, runtime, and deployment
02	**Lifecycle**	Requested, active, waiting, refused, failed, completed, reconciled, superseded
01	**Identity**	Work, object, actor, source, artefact, and external correlation

The entry point changes. The product boundaries and underlying architecture do not.

STATE REVIEW

If an operator must read a log or ask the original engineer to reconstruct a state transition, the owned product model is incomplete.

BOUNDARY · INGRESS

The input contract is more than a prompt

Identity, purpose, source scope, version, capability, authority, expected result, and evidence requirements belong in the work envelope.

The product's principal input is a business event or user request bound to a party, purpose, domain, current record version, permitted fields, applicable pack/policy, and authority. A production request should separate human-readable intent from the fields that determine access and execution. The caller does not gain a broader capability by describing it in prose; Charter resolves the permitted operation through authenticated identity, configured policy, and the selected workload contract.

The envelope should name tenant/workspace or deployment scope, work and idempotency identity, caller and subject, purpose, source/object references, requested product capability, version-selection rule, authority ceiling, deadline and resource budget, determinism mode where relevant, expected result schema, evidence profile, and callback or reconciliation identity.

Validation happens before expensive or consequential work. Missing identity, ambiguous purpose, unknown schema, inaccessible object, duplicate request, unavailable capability, incompatible version, or authority that cannot be established returns a typed rejection. The product should never silently use "current policy", a system administrator, or a broad default source set to fill a material gap.

ILLUSTRATIVE CHARTER ENVELOPE

```
work_id       charter-4f9e
actor         role:approved-caller
purpose       bounded-workflow
objects       content-addressed refs
capability    selected_charter_operation
authority     recommend_only
versions      approved manifest
result        typed state + artefact
evidence      standard packet
```

Illustrative fields. Use the current product interface and schema in implementation.

INGRESS TEST

Send missing, duplicated, stale, cross-tenant, over-broad, incompatible, and unsupported requests. None should enter hidden partial work.

BOUNDARY · RESULT

The output contract must expose operational state

A natural-language answer, success boolean, or transport status cannot carry the lifecycle, authority, provenance, and reconciliation state a caller needs.

The principal product output is an updated company record or explicit no-change result with proposal, human authority, event, domain state, and audit linkage. Its envelope should distinguish the artefact a user or downstream system consumes from the state needed to interpret it: completed, completed with limitation, refused, awaiting review, failed before effect, uncertain after possible effect, superseded, or closed and reconciled.

Return work and result identity, schema and semantic version, actual product/runtime manifest, material source/object references, result payload or artefact reference, warnings and uncertainty, policy/authority/human state, permitted next actions, external receipt or reconciliation state, evidencetrace and proof page reference, retention/export classification, and typed error details.

The contract should remain stable across UI, API, event, or batch surfaces even when transport mapping differs. A successful HTTP request can carry a valid refusal; a failed connection can occur after external effect; a stream can end with partial work. The caller must not infer business outcome from transport alone.

RESULT STATE	CALLER BEHAVIOUR	EVIDENCE
Completed	Consume declared artefact	Manifest + result
Awaiting authority	Present/queue review	Basis + gate state
Refused	Do not retry around policy	Rule/reason + owner
Failed safely	Retry/escalate by class	Failure + no-effect state
Effect uncertain	Reconcile before retry	Attempt + target correlation
Superseded	Resolve newer work	Version relationship

Result-state semantics belong to the product contract; exact enum names are implementation-specific.

CALLER TEST

A caller should choose the safe next step from the typed state without parsing prose or inspecting an internal log.

BOUNDARY · CONTRACTS

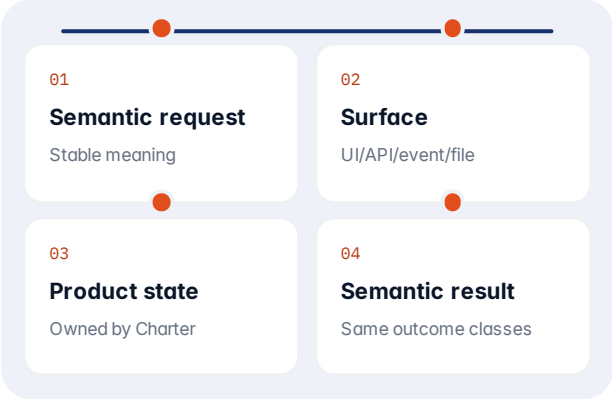
Interfaces carry the same meaning across surfaces

Charter can be reached through the surfaces named on its route and the platform around it; transport diversity must not create semantic drift.

Charter's route describes one party graph seen through multiple domain lenses, four planes on one substrate, field- and purpose-scoped projections, proposed actions that are signed and expire, human authorisation before commit, a run machine, event/outbox fan-out, provider adapters, and portable country/domain packs.

Loom is not a model router with a larger catalogue. It is a runtime for composing heterogeneous cognitive components into one typed execution graph. A node can be a perception transform, a memory operation, a micro model, an embedding or retrieval stage, a reasoning mode, a constraint set, a native or integrated solver, a knowledge operation, a proof checker, a domain expert, or an expression stage. The planner resolves the graph from the task, policy, available capability, evidence requirements, and execution budget. Only the required graph becomes active.

A compatibility suite should drive every supported surface with the same work cases and compare resulting product state, artefact identity, authority, and evidence. Provider-specific or transport-specific information may live in extension fields, but it cannot change the meaning of a completed, refused, failed, or awaiting-authority result without an explicit contract version.



The surface adapts transport; it should not redefine the work.

PARITY EXERCISE

Create equivalent work through two selected surfaces and reconcile the resulting work identity, owned state, result, and evidence.

BOUNDARY · COMPOSITION

Dependencies point downward through declared artefacts

Charter's answer is to stop connecting many systems and asking AI to infer the seams. The company keeps the ground truth, and AI becomes a governed participant inside it rather than an external copilot guessing from fragments.

The direct callers are employees, customers, business events, domain workflows, and governed assistants. The product can depend on party/source graphs, domain engines, packs, policy, provider adapters, and evidence/event foundations. Adjacent operational and organisational systems include identity, finance/communications providers, local models, data-rights processes, and operators. These relationships should appear in one versioned dependency and capability manifest so startup, failure, upgrades, packaging, and offline closure can be reasoned about.

A lower component receives the minimum typed input required for its mechanism and returns an artefact, reference, state, and evidence. The product may validate, combine, route, or present that result according to its own responsibility; it should not modify the lower artefact while continuing to cite its original identity. Cross-layer feedback uses an event or explicit control contract.

Mark each dependency required, optional, or selected by policy; identify approved implementations and capabilities; record version compatibility, data/trust boundary, timeout/failure mapping, determinism effect, update coupling, and evidence. Then sever optional dependencies and verify the declared degraded mode instead of discovering hidden control-plane calls in production.

04	**External context**	identity, finance/communications providers, local models, data-rights processes, and operators
03	**Lower capabilities**	party/source graphs, domain engines, packs, policy, provider adapters, and evidence/event foundations
02	**Charter**	the organisation's single governed operating graph
01	**Callers**	employees, customers, business events, domain workflows, and governed assistants

Responsibility/dependency view; exact deployment services follow the current manifest.

HIDDEN-DEPENDENCY TEST

Resolve the selected component closure with external networks disabled. Every unexpected fetch, account, model, key, licence, or telemetry call is an architecture finding.

BOUNDARY · DATA

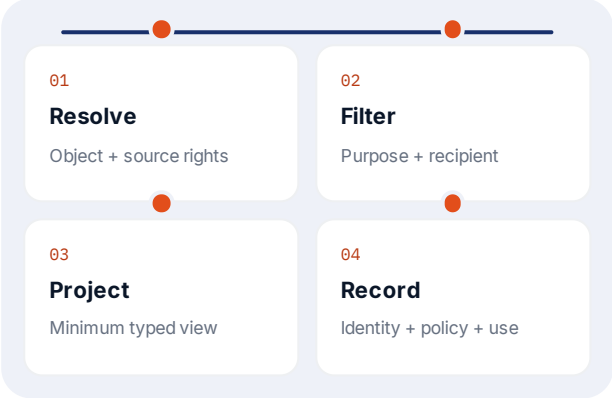
Data moves by purpose-scoped projection

The component initiating a call can know more than the recipient. Access at one layer does not automatically flow to a model, agent, tool, reviewer, or support operator.

Before Charter passes material to party/source graphs, domain engines, packs, policy, provider adapters, and evidence/event foundations, it should resolve the task purpose, recipient capability, source/object rights, field/data classification, deployment/provider boundary, transformation, retention, and evidence policy. Prefer immutable references or a minimum typed projection over copying the upstream workspace, case, corpus, repository, or estate state.

Draw control flow and data flow separately. Control includes request, gate, retry, cancellation, callback, and reconciliation. Data includes payloads, source snapshots, indexes, prompts, model/tool inputs and outputs, caches, operational telemetry, evidence, backups, support bundles, and exports. Attach encryption/key ownership, store, retention, and administrator path to each data edge.

Test role and purpose changes, object revocation, provider fallback, cache invalidation, replay, support access, export, and deletion. A projection that was legal for the original worker may be illegal for a substituted agent or historical reviewer. A fallback that preserves availability can still violate the selected data boundary.



Data authority is evaluated at the receiving boundary, not inherited from orchestration.

DATA QUESTION

Where can sensitive data from a Charter work item appear, including caches, logs, evidence, backup, replay, support, and provider paths?

BOUNDARY · IDENTITY

Human, service, agent, and signing identities stay separate

Infrastructure administration, product use, delegated work, policy ownership, approval, and cryptographic attestation are different authorities even if one directory supports them.

Charter is organized as four planes over one operating graph. The experience plane is the web app that gives people a coherent interface. The policy plane is the API and policy layer that validates every request and applies permissions. The operating graph stores the durable business state. The execution plane handles events, workflows, jobs, realtime collaboration, AI actions and external integrations. It holds together because domains share spines: a support ticket can see subscription status because support and billing share a customer, and a project inherits context from a won deal because delivery and CRM agree on the company.

For each operation, record authentication method, tenant/workspace, subject, role or capability, purpose, resource/field scope, policy and effective version, approval or delegation, time/expiry, result/reason, revocation path, and evidence. Privileged platform access needs a separate support/emergency process and cannot silently inherit domain or evidence-disclosure rights.

Exercise revoked identities, stale roles, expired capability, delegated and break-glass access, cross-tenant/purpose calls, self-approval, reused signatures, and support access. Enforce the same decision through every surface, including API, agent, replay, administrator, and evidence export paths.

PERSON

Human identity

Authentication, organisational role, purpose, review, correction, and accountability.

RUNTIME

Service identity

One component calling another under a bounded product contract.

DELEGATE

Agent identity

Capability, task, lifecycle, tool ceiling, and result responsibility.

ATTEST

Signing identity

Purpose-specific key binding to an artefact or state.

SEPARATION TEST

Operating Charter must not automatically grant authority to approve its domain results or inspect every retained work artefact.

BOUNDARY · GOVERNANCE

Policy and authority are runtime inputs

The product exposes and enforces configured boundaries; it does not invent the organisation's purpose, source authority, review roles, or acceptable consequence.

The request and product state should bind the applicable policy, authority ceiling, human gates, permitted actions, separation requirements, expiry, and exception route. Charter can make those conditions executable and observable; the customer's named owners still decide their content and legitimacy.

Express each material transition as allowed actors/capabilities, required facts or artefacts, rule/policy version, automatic versus human condition, limits, permitted next states, escalation owner, and evidence. If a judgement cannot be reduced to an approved rule, preserve it as an accountable human decision rather than hiding it inside prompt wording.

A reviewer needs the source and version, material facts, proposed result or action, conflicts/uncertainty, meaningful alternatives, consequences, and reason capture. Reopen the gate if the source, policy, result, action, or authority changes after review. Test stale and self approval, missing reviewer, excessive batch approval, and tool execution without gate state.

STATE	AUTHORITY	NEXT
Draft	Assigned worker/agent	Validate
Ready	Workflow contract	Review
Approved	Named human/policy	Act
Rejected	Named authority	Close/rework
Escalated	Rule/reviewer	Specialist
Executed	Scoped tool	Reconcile

Illustrative authority states; the domain owner supplies the actual policy.

GOVERNANCE BOUNDARY

Charter owns the company record and lifecycle. AI proposes inside the permissions of that graph; it does not self-authorise. A single data model does not remove domain ownership, legal duties, external provider constraints, or the need to isolate tenants and deployments.

BOUNDARY · NON-CLAIMS

The product's limitations belong in its architecture

A technically strong mechanism is easier to trust when the publication states what it cannot establish or own.

Charter owns the company record and lifecycle. AI proposes inside the permissions of that graph; it does not self-authorise. A single data model does not remove domain ownership, legal duties, external provider constraints, or the need to isolate tenants and deployments.

Product outputs depend on source quality, policy, customer data, configuration, model/agent/tool providers, infrastructure, operator practice, human competence, and the selected deployment. A valid trace does not make the purpose legitimate; a deterministic result does not make the rule correct; a signed approval does not prove understanding; a customer-hosted runtime does not guarantee good security operations.

Performance figures belong to their published workload, corpus, hardware, version, configuration, method, and comparison. Availability, source status, interface support, deployment rights, usage treatment, certifications, and support can change and must be verified against the current published description, delivered manifest, assurance material, and written agreement.

CAN EXPLAIN

Mechanism

How Charter owns its declared work and emits state/evidence.

CAN TEST

Installed behaviour

Contracts, failure, recovery, deployment, performance, and evidence on a real workload.

CANNOT SETTLE

Legitimacy

Purpose, lawfulness, fairness, proportionality, and organisational judgement.

CANNOT GRANT

Rights

Commercial, source, support, deployment, and modification terms.

HONEST PRODUCT TEST

For every strong statement, name the artefact or customer exercise that would support it and the conclusion that still would not follow.

02

HOW IT WORKS

Follow one customer moves from first signal to renewal

One illustrative work item turns architecture nouns into a complete sequence with source versions, product state, human authority, failure, recovery, and a record.

Illustrative flow; exact schemas and interfaces come from the current product implementation.

HOW IT WORKS · SCENARIO

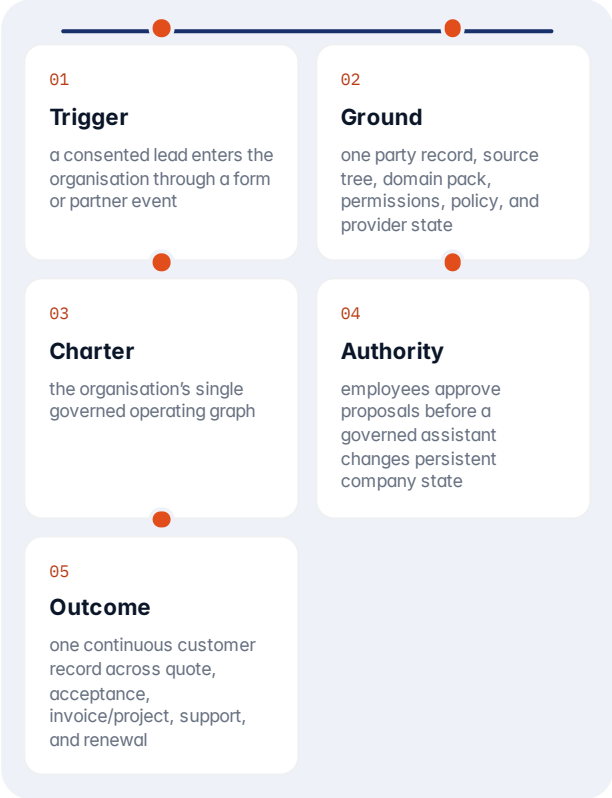
Walkthrough: one customer moves from first signal to renewal

This illustrative scenario follows one work identity through source, product state, exact or interpretive work, human authority, action, result, and evidence.

The trigger is a consented lead enters the organisation through a form or partner event. The approved inputs are one party record, source tree, domain pack, permissions, policy, and provider state. The example is deliberately bounded: it is not a customer result, performance claim, or current schema. Its purpose is to make the product boundary testable in a case with material versions, decisions, failure, and a record.

Charter's distinction is not that it has AI buttons. It is that AI is bound to the operating model. The assistant knows where the user is, which record is open, which domain they are working in and which permissions apply. Reads are policy gated: every read tool, system search and knowledge retrieval inherits RBAC, field-level rules and purpose-level policy, so sensitive HR, finance and recruiting fields are not exposed to the model just because a same-role user can read a nearby page. AI does not need special data paths, it needs stricter ones.

Human authority remains explicit: employees approve proposals before a governed assistant changes persistent company state. Consequential effect is separate from proposal: domain workflows commit events, invoke providers, and reconcile outbox deliveries. The resulting closure is one continuous customer record across quote, acceptance, invoice/project, support, and renewal. Later pages break this path into contracts, failure, recovery, evidence, deployment, and acceptance.



Illustrative end-to-end path; current implementation interfaces supply exact object and state names.

SCENARIO ASSUMPTIONS

All identities, dates, organisations, outcomes, percentages, and field names are illustrative. Source rights, policy, human roles, integrations, and retention must be designed for the actual workflow.

HOW IT WORKS · WORK SPINE

Stage 1: create one stable work identity

The same business work survives decomposition, retries, provider changes, review, action, reconciliation, correction, and replay.

When a consented lead enters the organisation through a form or partner event, the ingress surface validates tenant/workspace or deployment scope, caller and subject, purpose, source/object access, workload contract, policy selection, authority ceiling, evidence profile, and idempotency. Only then does Charter accept a stable work identity.

Task, attempt, artefact, trace, event, receipt, and external-correlation identities derive beneath the work. A retry appends an attempt; a correction or new decision supersedes the prior version; a duplicated delivery resolves to the existing state; an external callback resolves through its correlation. None silently becomes a new unrelated case.

The work manifest begins at acceptance and grows with actual versions used. This matters when configuration or “current” product state changes during a long-running case. Historical review loads the manifest bound to the work, not whatever is latest at review time.

WORK RECORD	charter-work-4f9e	
PURPOSE	**bounded scenario**	BOUND
CALLER	**role + tenant**	BOUND
SOURCES	**approved references**	PENDING
AUTHORITY	**declared ceiling**	BOUND
STATE	**accepted**	OPEN
EVIDENCE	**packet manifest**	OPEN

Illustrative record shape; use the product’s implemented object model.

IDENTITY CHECK

Start from the authoritative external record and the product record independently. Both should resolve to the same work and closed/superseded state.

HOW IT WORKS · GROUNDING

Stage 2: bind sources, policy, and actual versions

The result must be traceable to the material that applied to this work at its effective time, including conflicts and transformations.

The scenario depends on one party record, source tree, domain pack, permissions, policy, and provider state. Each decisive input needs an owner, publisher or system of record, scope, authority, effective interval, rights, confidentiality/data class, transformation lineage, conflict/precedence rule, review/withdrawal policy, and stable identity.

Charter should consume approved references or minimum projections, validate versions and access, and preserve which material actually entered the work. Retrieval or selection may rank candidates, but ranking cannot decide authority. Missing, withdrawn, stale, conflicting, or insufficient source state becomes a refusal or escalation according to the workflow.

The manifest also pins product, workflow, model/expert/agent/tool, solver or numeric profile, policy, schema, compiler/runtime, target, deployment, and relevant trust identities. Content hashes are useful for immutable artefacts; signed version identifiers and effective metadata cover state that cannot be reduced to a file.

SOURCE + VERSION BINDING	effective-work-state	
SOURCE	identity + authority	PINNED
POLICY	version + effective time	PINNED
TRANSFORM	method + lineage	LINKED
RUNTIME	product + dependencies	PINNED
CONFLICT	rule or escalation	CHECKED
RIGHTS	purpose + recipient	CHECKED

The “current” version is not evidence of what a historical case used.

GROUNDING TEST

Change the source or policy after review. New work should bind the new version; old work should retain the old path; pending work should follow an explicit reopen rule.

HOW IT WORKS · MECHANISM

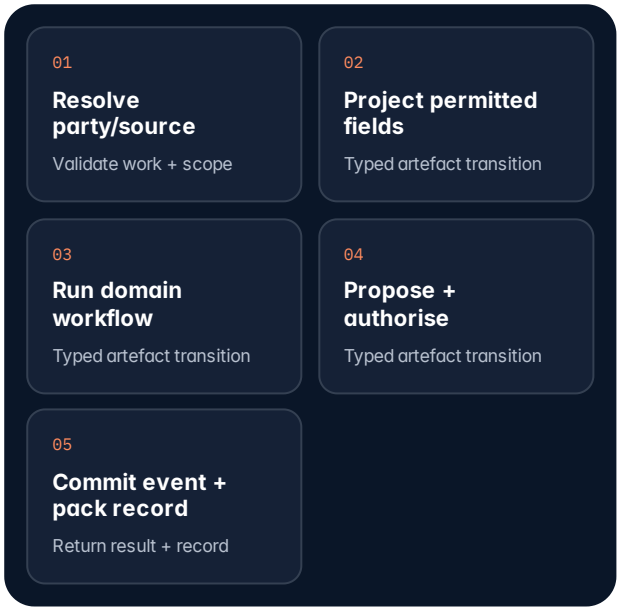
Stage 3: execute the Charter product flow

The published description's conceptual sequence is Resolve party/source → Project permitted fields → Run domain workflow → Propose + authorise → Commit event + pack record. Every transition has an owner, validation, failure state, and evidence contribution.

Charter's route describes one party graph seen through multiple domain lenses, four planes on one substrate, field- and purpose-scoped projections, proposed actions that are signed and expire, human authorisation before commit, a run machine, event/outbox fan-out, provider adapters, and portable country/domain packs.

Most enterprise architectures model the company as a federation of external systems, each with its own identity, state machine, permission model, event history, retry behaviour, and API. The integration layer tries to make those incompatible worlds look coherent. Charter starts with one operating graph and builds the domains on top of it. Parties, money, work, people, knowledge, documents, events, policy, and AI actions share one system boundary. The selected licence defines operational scope; source and modification rights are separate.

The result is an updated company record or explicit no-change result with proposal, human authority, event, domain state, and audit linkage. That output should expose enough typed state for the caller to act safely and enough references for authorised review. Internal implementation can parallelise, stream, cache, retry, or choose an approved provider, but it must not lose ordered/causal identity or change the product's declared result meaning.



Conceptual Charter flow from the current product description.

TRANSITION CHECKLIST

Identity; input/version; authority; validation; timeout/cancel; retry/idempotency; failure class; output schema; evidence; next owner.

HOW IT WORKS · COGNITION AND COMPUTE

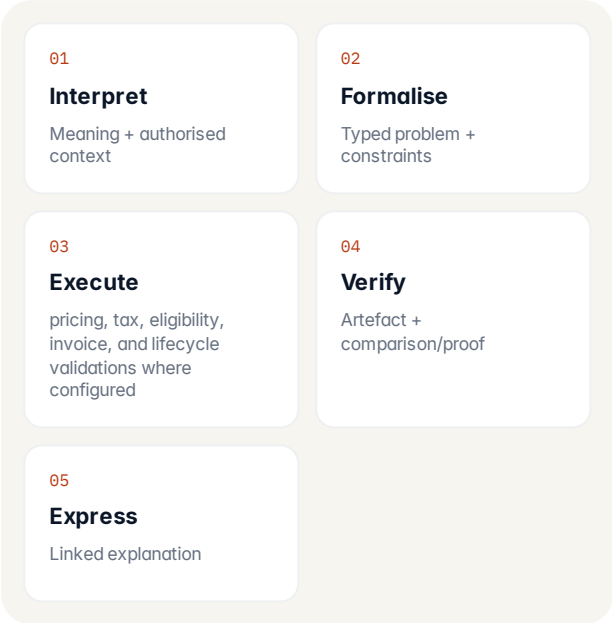
Stage 4: keep exact work outside fluent invention

The scenario's exact boundary is pricing, tax, eligibility, invoice, and lifecycle validations where configured. Interpretation and expression can use cognitive components; exact rules, arithmetic, simulation, or compilation use typed deterministic contracts where required.

The product identifies which subproblem requires source-aware interpretation and which requires exact mechanics. The interpretive side receives authorised context, catalogue/model and constraint versions, and a requested determinism/evaluation mode. The exact side receives typed units, shapes, rules, numeric profile, target and comparison or proof requirement.

A cognitive artefact may frame the exact question and explain the verified result. It must not recompute or paraphrase away a threshold, amount, constraint, compiler diagnostic, or simulation output. Conversely, the exact component returns a result/proof under its formal contract; it does not decide source legitimacy, purpose, or human authority.

Failure remains typed: insufficient context, conflicting sources, unsupported operation, invalid solver/proof, overflow/tolerance breach, target mismatch, model/provider outage, constraint conflict, or expression that cannot remain faithful. The workflow can retry approved components or escalate; it cannot invent a successful exact answer.



A separation of responsibilities; some product scenarios may use only a subset.

EXACTNESS TEST

Can an independent reviewer verify the exact artefact without reading the prose, and verify that the prose preserves the artefact without rerunning the model?

HOW IT WORKS · REVIEW

Stage 5: keep human authority meaningful

Employees approve proposals before a governed assistant changes persistent company state.

The review state packages the material facts, source and policy versions, product result, exact artefacts where used, uncertainty and conflict, proposed action or publication, affected object or person, limits, and the real alternatives available to the reviewer. The interface should help a competent person challenge the work rather than encourage a fast confirmation.

For this scenario, employees approve proposals before a governed assistant changes persistent company state. The reviewer identity, organisational role, purpose, scope, separation requirements, information shown, decision, reason, time, expiry, correction or escalation, and downstream effect attach to the work. A reviewer can approve, reject, correct, request evidence, narrow, or escalate according to policy.

If sources, policy, result, action, recipient, or target state change after review, reopen or invalidate the gate. Test missing, wrong-role, self, expired, delegated, bulk, and emergency review. Measure correction and disagreement; a low correction rate may indicate excellent work or ceremonial oversight.

SEE

Material basis

Facts, source/policy version, uncertainty, conflict, and exact artefacts.

CHOOSE

Real alternatives

Approve, correct, reject, request evidence, narrow, or escalate.

UNDERSTAND

Consequence

What publication or side effect follows each decision.

RECORD

Authority state

Identity, scope, reason, time, expiry, and transition.

MEANINGFUL REVIEW TEST

Seed a plausible but wrong recommendation. Can the authorised reviewer see enough to catch it before effect?

HOW IT WORKS · SIDE EFFECTS

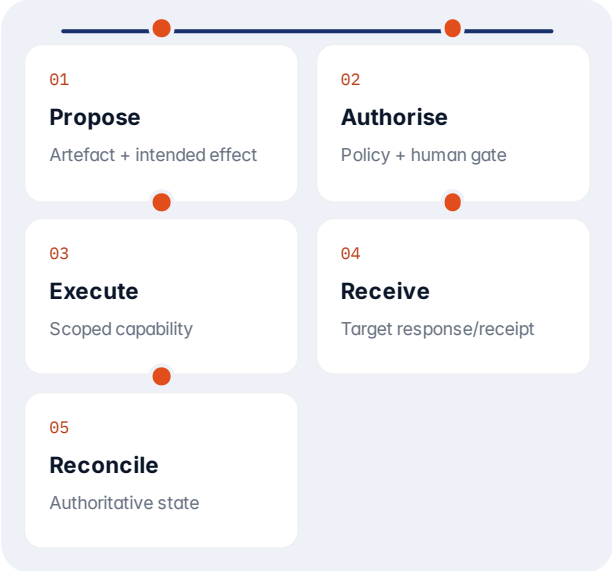
Stage 6: separate proposal, action, and reconciliation

Domain workflows commit events, invoke providers, and reconcile outbox deliveries.

The operating graph holds durable company state, identities, relationships, domain records, and lifecycle transitions. The execution plane handles events, outbox, queues, workflows, scheduled jobs, realtime collaboration, AI actions, external integrations, and audit materialisation. The planes share contracts and identifiers. They do not become separate products.

A successful call is an execution attempt, not automatically a closed business transaction. Attach the target response or receipt, then query or consume authoritative state as needed. Timeouts after possible effect enter an “uncertain, reconcile” state. Retry only after idempotency or reconciliation proves it is safe.

Record proposal, approval, execution transcript or action receipt, provider/target identity, request commitment, response, external correlation/version, reconciliation attempts, final state, notification, and correction. A compensating action is new governed work; it does not erase the first effect.



The target system, not the tool call alone, closes an external action.

AMBIGUOUS-TIMEOUT TEST

Drop the response after the target accepts the action. The workflow must reconcile without either losing or duplicating the effect.

HOW IT WORKS · EVIDENCE

Stage 7: build the evidence packet with the work

The result is one continuous customer record across quote, acceptance, invoice/project, support, and renewal; its record should be retrievable by an authorised reviewer without reconstructing the case from screenshots, current configuration, or operator memory.

Charter is part of the licensed Dweve platform, operated on customer infrastructure. The same product can run in a connected estate, a private site, or a fully air-gapped environment with local Dweve AI and no external model calls. The agreement defines the estates, isolation, support, escrow, and any separately selected source or modification rights.

Evidence design for this product includes Retain party/source identity, projected fields and purpose, domain/pack/policy versions, workflow/run transitions, assistant proposal, signature/expiry, human decision, committed event, provider receipt, outbox delivery, erasure or correction effects, and audit reference.. Operational telemetry can reference the packet but should not become its only store. Sensitive artefacts remain behind purpose-specific access; the packet manifest supports an officer, operator, examiner, affected person, or support engineer resolving different authorised subgraphs under one integrity identity.

Define completeness before work can close. If a declared consequential packet item is missing or cannot be committed, the workflow may have to hold publication/action or enter an explicit evidence-incomplete state. Retain schemas, keys, verifiers, runtimes and manifests required for supported historical checking.

WORK RECORD	scenario-packet	
IDENTITY	**work + purpose + actors**	SEALED
SOURCES	**versions + lineage + rights**	SEALED
PRODUCT	**Charter + dependency manifest**	SEALED
AUTHORITY	**policy + human states**	SIGNED
OUTCOME	**result + target receipt**	CLOSED
PROOF	**integrity + verifier deps**	VERIFIED

Illustrative packet manifest; access determines which artefacts a reviewer may resolve.

PACKET TEST

Select a random historic case, retrieve it without production engineers, verify integrity offline, and reconcile the recorded human and target states.

HOW IT WORKS · FAILURE

Failure paths are first-class product behaviour

A source, model, agent, tool, reviewer, target, store, network, policy, evidence check, or deployment dependency can fail before, during, or after possible effect.

For Charter: Reject stale record versions, over-broad field projection, missing purpose, invalid pack, expired proposal, unauthorised approval, failed policy gate, duplicate outbox delivery, provider rejection, partial domain transition, erasure conflict, and an event that cannot reconcile with the persistent party state. Every failure maps to a typed state, safe default, retry/timeout/cancel rule, owner, alert, evidence, and recovery or escalation path. A valid refusal is not a system error; an effect-uncertain timeout is not safe to retry blindly; an evidence mismatch is not a warning to ignore.

Model failure by boundary: invalid/missing input, authentication/authority denial, source conflict/unavailable, lower artefact invalid, model/provider outage, resource/budget limit, human unavailable, tool denied, external target ambiguous, persistence/queue error, integrity/replay mismatch, and operatordeployment page failure. Combine faults after the single-fault paths work.

Inject failures through the production-like path and preserve original work identity, every attempt and decision, actual safe state, telemetry, packet, target reconciliation, recovery time, manual intervention, and runbook change. Do not clean state before the investigator sees it.

FAILURE CLASS	UNSAFE RESPONSE	GOVERNED RESPONSE
Source/policy	Guess/current default	Stop, approved snapshot, or escalate
Dependency	Invisible fallback	Approved substitute + manifest or hold
Authority	Auto-approve	Queue/delegate by explicit rule or stop
Tool/target	Blind retry	Receipt/idempotency/reconcile
Evidence	Publish anyway	Hold/quarantine/investigate
Recovery	Overwrite attempts	Append transitions + retain history

The configured response belongs to the workload and consequence.

SAFE COMPLETION

Refusal, escalation, or stopped action can be the correct completed outcome when continuing would violate the contract.

HOW IT WORKS · RECOVERY

Recovery preserves identity and reconciles reality

Retry, resume, reassign, substitute, checkpoint, compensate, restore, and replay solve different failure states and must not be treated as synonyms.

Retry repeats a safe attempt under the same contract; resume continues from a durable checkpoint; reassign changes the responsible worker; substitute changes an approved implementation and manifest; compensate performs a new action to counter a prior effect; restore reconstructs platform state; replay or verification examines historical work. Each transition keeps the original and new identities.

For Charter, recovery must preserve party spine, source tree, domain-engine records, permissions and purpose projection, workflow graph, proposed actions, policy gates, run state, events/outbox, provider registry, country/domain packs, evidence, and lifecycle history. If the product calls lower components or external systems, it also preserves their artefact/receipt identities and compatibility. Queues and callbacks need idempotency; mutable objects need version/concurrency rules; human approvals need expiry/revalidation; changed source/policy or provider requires an explicit reopen decision.

A clean-room recovery test restores configuration, identities, product state, sources/knowledge, policies, model/agent/tool manifests, evidence/proof material, schemas/verifiers, queues and external correlations. It runs one safe new case, retrieves/verifies one historic case, and reconciles authoritative target state.

01 **Classify**
Know whether effect occurred and which state is durable.

02 **Choose**
Retry, resume, reassign, substitute, escalate, compensate, restore, or stop.

03 **Execute**
Use explicit owner, version, limits, and idempotency.

04 **Reconcile**
Compare product, evidence, and authoritative external state.

05 **Close**
Retain all attempts and improve corpus/runbook.

RECOVERY PROOF

Follow one party from first signal through quote, signed acceptance, invoice/project, support, and renewal; attempt a forbidden field read and expired proposal; fail a provider; replay events; reconcile outbox delivery; run a rights request; and export the owned record.

03

ADOPTION AND OPERATION

Make Charter a service the organisation can rely on

Deployment, security, observability, change, recovery, acceptance, and ownership decide whether product architecture survives production.

A product is production-ready only when each step has an owner and exercised evidence.

ADOPTION AND OPERATION · POSTURE

Choose the deployment posture from workload obligations

Managed Fabric access, licensed customer operation, and air-gapped operation are responsibility and dependency arrangements, not product features or generic security rankings.

For the scenario, one customer moves from first signal to renewal, begin with permitted data movement, administrative/key custody, continuity, latency and integration locality, supplier independence, update control, source/modification and licence rights, evidence custody, support, and the organisation's ability to operate Charter.

Controlled, Autonomous, and Independent are Government & Critical Infrastructure package tiers. Every tier carries the full licensed platform in a customer-operated, physically isolated estate. The difference is source access, modification rights, technology transfer, and the independence path. List prices are EUR 2,000,000, EUR 4,000,000, and EUR 6,000,000 per year, with 25 percent pre-order pricing on all three.

A mixed estate can use more than one posture, but work identity, artefact versions, data transfer, policy, evidence, and historical interpretation must cross explicitly. Select the least operationally complex posture that can demonstrably satisfy the workload's real obligations.

Directional comparison; current service description, architecture, and agreement settle exact responsibilities.

POSTURE DECISION

A more isolated posture is not automatically safer if the organisation cannot patch, monitor, restore, investigate, and support it.

ADOPTION AND OPERATION · MANAGED

Receive Charter capabilities through managed Fabric

Define the managed Fabric or public-Mesh service surface, support, evidence, and customer responsibility boundaries without implying that the customer operates Charter directly.

Map how Fabric, where Charter capabilities are exposed when included is reached; which product and evidence stores are involved; whether model, tool, connector, or provider paths leave the immediate service; how tenant/workspace access is separated; and which administrative or support paths can reach configuration, payloads, telemetry, or evidence.

The current service description and agreement should settle region, tenancy/isolation, encryption and key model, customer and Dweve administration, identities, providers/subprocessors where applicable, retention/deletion, backups/recovery, availability/change/incident, evidence access/export, usage/limits, support, and exit.

Exercise a representative tenant: create work through Fabric, where Charter capabilities are exposed when included, remove a participant or credential, change a permitted provider or dependency, test deletion/export and packet retrieval, raise an incident, confirm support access controls, and document the customer owner for every source, policy, role, action, and retained result.

04	**Assurance**	Service evidence plus customer work packet
03	**Business Workspace**	Platform operation, isolation, release, recovery
02	**Tenant controls**	Roles, products, models/tools, retention
01	**Customer work**	Purpose, sources, policy, people, outcomes

Validate this responsibility model against the current service description.

MANAGED EVIDENCE

Request the current data/control flow, admin/support process, incident route, recovery commitment, export format, and applicable assurance material.

ADOPTION AND OPERATION · LICENSED

Operate Charter as a licensed platform service

Customer control over infrastructure and keys brings customer responsibility for capacity, security, changes, recovery, evidence, and service continuity.

Specify the complete Charter environment: signed product and dependency packages, models/policies/sources where used, compute and target hardware, operating platform, stores/queues, identity, keys/trust, network, observability, backups, evidence/verifier dependencies, licence material, and integrations. Pin versions and supported combinations.

Assign an accountable service owner; platform, security, identity, source/knowledge, workflow/policy, evidence, change, incident and support owners; and administrators with separated privileges. Document capacity, service objectives, on-call/escalation, vulnerability/patch process, update windows, clean restore, and customer-modification support.

Acceptance includes install, cold restart, dependency outage, representative load, least privilege, key rotation, backup/restore, product workflow failure, historic packet verification, update/rollback, supplier disconnect, export, and decommission. A binary that starts has not passed this operating test.

WORK RECORD	licensed-production	
SERVICE	**accountable product owner**	ASSIGN
PLATFORM	**install/observe/recover**	ASSIGN
SECURITY	**keys/network/access**	ASSIGN
DOMAIN	**source/policy/review**	ASSIGN
EVIDENCE	**retain/verify/disclose**	ASSIGN
CHANGE	**test/promote/rollback**	ASSIGN

An unassigned row is an operational dependency, not a later detail.

COMMERCIAL BOUNDARY

Confirm included product/version, environments, usage, source/modification, updates, support, DR copies, offline rights, and exit in the current agreement.

ADOPTION AND OPERATION · DISCONNECTED

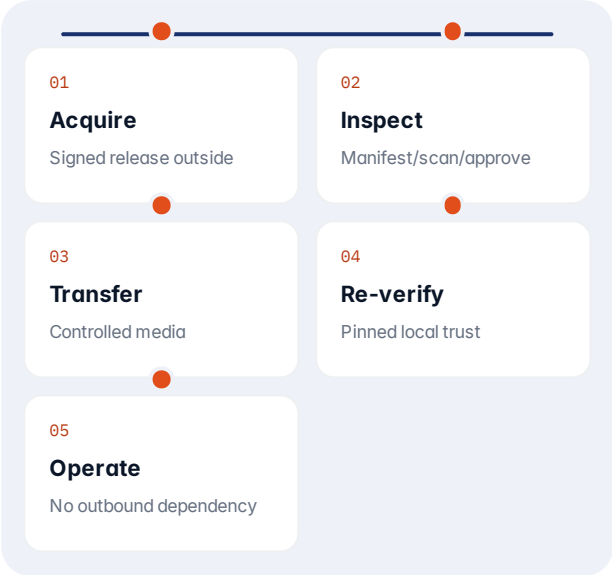
Prove Charter works in an air gap

No outbound dependency covers installation, licence validation, work, evidence, verification, administration, recovery, updates, and support, not merely model inference.

An ordinary Charter deployment licence provides the operational binaries and configuration required for the licensed scope. It does not include proprietary source or modification rights. Eligible customers can separately purchase controlled review, operational source and internal modification rights, a managed customer fork, or independent-operation rights.

Operate with physically unavailable external DNS, package registry, model/provider, telemetry, licence, account, certificate, time, and support services. Updates move through an approve–transfer–reverify–stage–corpus/replay–promote path with a locally usable rollback. Outbound diagnostics are minimised, reviewed, authorised, protected, and recorded.

Test cold start after prolonged disconnection, licence/time/key/certificate boundaries, normal and failed product work, local packet generation and offline verification, backup/restore, update/rollback, support without raw data export, and removal. Every unexpected network attempt or external prerequisite is an air-gap finding.



Air-gap intake path for Charter; outbound support needs a separate least-data path.

CABLE-ABSENT PROOF

Run the complete scenario, its failure and recovery cases, and the historical verifier while every external path is physically unavailable.

ADOPTION AND OPERATION · SECURITY

Threat-model the product as part of a system

Untrusted callers, sources, models, agents, tools, dependencies, administrators, support processes, evidence readers, and packages create chained paths across product boundaries.

A request to label an incoming ticket may need one classifier. A request to find the cancellation date in a contract may need document perception and exact span extraction. A request to find the most relevant policy may need an embedder, retrieval, relevance scoring, ranking, and reranking. A request to determine whether several conditions can all be satisfied may need a constraint solver. A request to explain why an outcome occurred may need causal reasoning, memory, evidence retrieval, and verification. A complex cross-domain request may need several experts and several solver or reasoning paths before a language model expresses the result.

Use default-deny capabilities, minimum context projection, structural separation of data and instructions, output/schema validation, policy after model generation, sandbox/tool limits, purpose-specific keys and signatures, immutable/signed packages, least-privilege administration, secret/PII handling, rate/resource controls, monitoring, and incident containment.

Build an abuse corpus and chained tests, not only prompt strings. Attempt a malicious source to influence output, output to select a tool, tool to access a forbidden resource, error/log to expose a secret, support/export to cross the boundary, and fallback to change provider/data locality. Retain findings and regression cases.

BOUNDARY	QUESTION	EVIDENCE
Identity	Can authority be delegated/escalated?	Permit/deny trail
Source/model	Can data become instruction?	Adversarial corpus
Agent/tool	Can output expand capability?	Gate + transcript
Admin/support	Can operation bypass domain policy?	Privileged-access record
Package	What runs and from where?	Manifest + signatures
Evidence	Who can inspect/export?	Purpose + access event

Threat paths cross product, deployment, and organisational boundaries.

SECURITY RULE

A network perimeter cannot substitute for identity, capability, data, action, and evidence controls inside it.

ADOPTION AND OPERATION · OBSERVABILITY

Observe service health without duplicating sensitive work

Telemetry answers availability, performance, capacity, threat, and incident questions; governed work evidence answers source, authority, decision, action, and replay questions.

The operating graph holds durable company state, identities, relationships, domain records, and lifecycle transitions. The execution plane handles events, outbox, queues, workflows, scheduled jobs, realtime collaboration, AI actions, external integrations, and audit materialisation. The planes share contracts and identifiers. They do not become separate products.

Define metric/logtrace and proof page schema, tenant/privacy handling, sampling, payload prohibition/redaction, retention, clocks/order, exporter, local buffering, access, alerts, service objectives, dashboards, and the authorised pivot into a work packet. Ensure licensed and air-gapped operations remain observable without a managed telemetry service.

Seed normal saturation, dependency failure, access denial, malicious input, evidence mismatch, dropped exporter, clock drift, queue duplication, and target reconciliation fault. Follow alert to safe state and relevant authorised evidence, then close incident and change record without broad payload disclosure.

OPERATE

Metrics/health

Rate, latency, resource, queue, dependency, SLO, and saturation.

INVESTIGATE

Service trace/log

State/error class, component, work id, admin/change, and security signal.

EXPLAIN

Work evidence

Sources, versions, artefacts, authority, result, action, and packet.

GOVERN

Access record

Purpose, actor, disclosed subgraph, export, and correction/deletion event.

OBSERVABILITY TEST

Operations should diagnose a seeded incident without either losing work correlation or reading unrestricted case content.

ADOPTION AND OPERATION · CHANGE

Updates are product-behaviour changes

A new release, model, source, policy, agent, tool, foundation, compiler/runtime, schema, or customer dependency can change the result and historical evidence.

Build the change delta across party spine, source tree, domain-engine records, permissions and purpose projection, workflow graph, proposed actions, policy gates, run state, events/outbox, provider registry, country/domain packs, evidence, and lifecycle history. Classify interface/schema, state migration, source/policy, model/agent/tool, numeric/determinism, performance, security, operational dependency, evidence/verifier, licence and support effects. Pin both old and new manifests.

Stage on a copy, migrate, and run the product acceptance corpus: normal distribution, boundary, refusal, malicious input, dependency failure, human gate, action/reconciliation, packet verification, historic replay, load and recovery. Define acceptable variation before seeing output. Obtain domain, technical, security, evidence and operations approvals for their boundaries.

Promote segmentally with a reversible window and state reconciliation. Test rollback after new work exists; preserve new and old artefacts, mappings and verifier dependencies; prevent mixed-version queues from applying the same effect twice. Retain the decision whether to close, hold, narrow or revert.

01 **Classify delta**
Behaviour, schema, state, security, operations, evidence.

02 **Stage**
Migrate copy and run full corpus/history.

03 **Approve**
Named owner per affected boundary.

04 **Promote**
Scoped, observed, and reversible.

05 **Close/rollback**
Reconcile state and preserve evidence.

COMPATIBILITY QUESTION

Can the new version still interpret and verify the historical work the organisation is required to retain?

ADOPTION AND OPERATION · CONTINUITY

Restore the product and its accountability

Recovery covers configuration, identity, product state, sources/policies, runtime artefacts, evidence/verifiers, queues, and external reconciliation, not only a database backup.

Classify every element of party spine, source tree, domain-engine records, permissions and purpose projection, workflow graph, proposed actions, policy gates, run state, events/outbox, provider registry, country/domain packs, evidence, and lifecycle history as source-of-truth, append log, derived index/cache, configuration/secret, immutable artefact, evidence, or external reference. Set workload-level recovery objectives and a restore order that prevents new work from running with stale policy or duplicated queued action.

Preserve product/dependency packages, models/agents/tools, policies/sources needed for historical work, schemas, public keys and trust policy, verifiers/runtimes, packet manifests, backups/checkpoints, external correlations and idempotency state, documentation and operator procedures. Resolve deletion, correction, legal hold and proof preservation deliberately.

Restore into clean replacement infrastructure with no undocumented operator state. Re-establish trust and identities; rebuild derived state; run a safe new work case; retrieve and verify a historical one; replay read-only or simulated paths; reconcile authoritative external systems; measure achieved loss/outage; and record exceptions.

WORK RECORD	clean-room-restore	
SERVICE	**config + identity + keys**	RESTORE
PRODUCT	**owned state + queues**	RESTORE
INPUTS	**sources/policy/artefacts**	RESTORE
EVIDENCE	**packets + schemas + verifier**	RESTORE
NEW WORK	**safe acceptance case**	EXECUTE
HISTORY	**retrieve + verify + reconcile**	PROVE

A green backup job is not a clean-room recovery exercise.

CONTINUITY BOUNDARY

Service recovery and accountability recovery can have different dependencies and objectives. Measure both.

ADOPTION AND OPERATION · ACCEPTANCE

Evaluate the product with a representative corpus

Take the trace file, run the open-source verifier on your laptop, and you have your proof. We could disappear and the trace would still verify.

Begin with the product-specific exercise: Follow one party from first signal through quote, signed acceptance, invoice/project, support, and renewal; attempt a forbidden field read and expired proposal; fail a provider; replay events; reconcile outbox delivery; run a rights request; and export the owned record. Add historic normal cases across real formats and roles, then boundary values, missing/conflicting/withdrawn sources, incompatible versions, unsupported capabilities, policy denial, malicious input, unavailable dependency, reviewer absence, ambiguous target effect, evidence mismatch, update, and restore.

Define invariants and outcome classes instead of one expected prose string. Correct sources and versions, data boundary, capability/authority, exact artefacts, required warning/refusal, task/action state, target reconciliation, packet completeness, and verification must hold. Protect and version the corpus; separate configuration/training from evaluation.

Run in the intended posture and representative hardware/service configuration. Preserve request, manifests, corpus and rights, raw outputs, timing/resources, errors, packet/verdict, human review, target state, exceptions, and disposition. A public benchmark or demo can guide cases; it cannot replace customer acceptance.

NORMAL

Distribution

Representative volume, formats, roles, sources, and outcomes.

BOUNDARY

Decision edges

Threshold, missing, conflict, version transition, uncertainty.

ABUSE

Disallowed path

Source/prompt/schema, access, capability, tool, export, and egress.

RESILIENCE

Failure/recovery

Dependency, human, target, evidence, restore, replay, and rollback.

RELEASE RULE

A feature tour shows possibility. A controlled corpus produces repeatable evidence about the product boundary.

ADOPTION AND OPERATION · OWNERSHIP

Run the product as a governed operational service

After release, sources, policies, models, agents, tools, packages, dependencies, case mix, threats, and people change. The product needs ongoing domain and technical ownership.

The accountable Charter owner is responsible for purpose, scope, outcomes, continued fitness and stop decisions. Source/policy/domain owners govern inputs and authority; technical owners maintain contracts/configuration; security owns threat controls; operations owns health/change/recovery; evidence owners maintain retrieval/verification/disclosure; human managers own competence and meaningful review.

Create review triggers for source or policy change, model/agent/tool/provider update, product/foundation release, new data/classification, changed consequence or autonomy, drift in outcome/denial/correction, repeated exception, incident/complaint/audit finding, capacity/support pressure, deployment migration, and rights/contract changes.

Hold regular case and control reviews across completed, corrected, refused, escalated and failed work. Inspect sources, product/dependency manifest, authority, action/target, packet and user impact. Decide whether to change corpus, source, policy, role, contract, capacity, security, runbook, product scope, or stop condition. Expansion is a new decision.

WORK RECORD	charter-monthly-control	
OUTCOME	**flow + quality + impact**	REVIEW
INPUTS	**source/policy/version changes**	REVIEW
AUTHORITY	**roles/gates/overrides**	REVIEW
SECURITY	**abuse/incident/access**	REVIEW
OPERATION	**SLO/capacity/change/recovery**	REVIEW
EVIDENCE	**sample retrieval + verify**	REVIEW

Record actions, owners, dates, accepted limits, and expansion/stop decisions.

OPERATING PRINCIPLE

No platform metric or supplier control substitutes for a domain owner who can narrow or stop the workflow.

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 · L–W

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

CLOSING PRODUCT RECORD

The next Charter action is a bounded product proof

Use the scenario “one customer moves from first signal to renewal” or one equally specific customer workflow. Require the product to show both its intended result and the evidence needed to judge the boundary.

Write the trigger, work identity, purpose, actors, sources and versions, product capability, lower dependencies, policy and authority, expected artefact, permitted action, external target, evidence packet, deployment, service objective, and failure/stop conditions. Remove every product or component that has no named responsibility.

Run the current Charter implementation against a protected corpus of normal, boundary, refusal, malicious, failure, recovery, historical verification, target reconciliation, update and restore cases. Exercise the selected managed, licensed, or air-gapped posture. Preserve raw manifests, outputs, timing/resources, decisions, packets, and gaps.

Then make the production decision with domain, technical, security, operations, assurance, procurement/legal, and affected-user owners: proceed, narrow, remediate, contract a missing condition, hold, or stop. The product earns reliance when that decision is evidence-backed and reversible, not when the brochure is persuasive.

WORK RECORD	charter-proof	
SCOPE	**one product/workflow**	BIND
ARCHITECTURE	**contracts + manifests**	INSPECT
CORPUS	**normal + boundary + abuse**	RUN
RESILIENCE	**failure + restore + replay**	RUN
OPERATION	**posture + owners + capacity**	PROVE
DECISION	**go/narrow/hold/stop**	RECORD

A product proof ends in a decision record, not a demo summary.

ACCEPTANCE EXERCISE

Follow one party from first signal through quote, signed acceptance, invoice/project, support, and renewal; attempt a forbidden field read and expired proposal; fail a provider; replay events; reconcile outbox delivery; run a rights request; and export the owned record.