commonline

TECHNICAL REVIEW / OCTOBER 2026

The load handoff, explained.

How Commonline converts acquired TMS data, makes unresolved meanings visible, and prepares loads for delivery to OTM.

For architects, integration leads and security reviewers.

CURRENT EVIDENCE

Working code, tested end to end on simulated systems. A real OTM tenant and customer production remain to be proven.



commonline.io · ruk@commonline.io

THE DECISION IN VIEW

Evaluate the handoff. Then prove it in your environment.

Keep acquired systems running while you examine one path into OTM: what is carried, what waits for a decision, and what the destination actually accepts.

01 / PLACEMENT

A layer beside your TMSs

One customer-hosted hub and database. Commonline converts and delivers; OTM owns planning.

02 / CONTROL

A visible human decision

Unclear meanings open an owned question. Mapping changes keep their version, preview and approval.

03 / PROOF

A bounded first test

Assess a source export, then validate loads and locations in your OTM test instance.

INSIDE THIS REVIEW

01 Scope and evidence ↗
02 System context ↗
03 The data path ↗
04 Canonical model and record trail ↗
05 Mapping and human decisions ↗
06 Delivery, acknowledgment and returns ↗
07 Storage and tenant isolation ↗
08 Deployment and recovery ↗
09 Security controls ↗
10 Optional AI assistant ↗
11 Console and tooling ↗
12 Measured performance ↗
13 Evidence and the design-partner test ↗

CHOOSE A READING ROUTE

Architecture: 02 → 03 → 06 · **Security:** 07 → 08 → 09 · **Evidence:** 12 → 13

01 Scope, audience and evidence tags

Commonline converts and delivers. It reads freight records from the TMSs a company runs, converts them into one canonical form, holds anything it cannot place with a question for a person, and delivers what is ready to OTM. OTM makes every planning decision. Commonline makes none, picks no carrier, and does not silently fill in a missing value.

It is customer-hosted: one hub process and one database per customer, running in the customer's cloud or data center. We have no standing access and the hub sends no telemetry.

Who this is for. A buyer's architect, an OTM implementation partner's technical lead, or a security reviewer who wants the mechanism and not the marketing. Security controls are summarized in section 9; the full self-assessment with file references is available on request, and a summary is at commonline.io/proof/security.html.

EVIDENCE TAGS USED THROUGHOUT

`SIMULATED`	Built and tested against simulators, public contract samples, mocks or fake endpoints. No real counterpart has run it.
`LOCAL RUN`	Measured on one developer machine or a local single-node cluster. Not a production baseline.
`LIVE, NARROW`	Touched a real external system in a bounded way, named where it is used.
Status qualifiers	**Not built:** implementation absent. **Not yet run:** implemented or configured, but not exercised in the stated setting. **Off by default:** disabled unless configured. **Not in place:** certification or external assurance absent.

THE SHORT VERSION OF THE EVIDENCE

No real OTM tenant has been connected, and no customer runs Commonline in production. The conversion, mapping, work queue, outbox and console are working code tested end to end on simulated systems. What has touched real systems is narrow: sign-in against Keycloak and, once, a free Okta developer org; read-only calls and two disposable test orders against one McLeod sandbox; and live calls to the AI model provider. A design partner's OTM test instance is where the rest gets proven.

02 System context

Everything in the dashed boundary runs in the customer's environment. Sources are polled or pushed in; OTM is the only destination for loads. A person works the exceptions in the console, which is served by the hub itself.

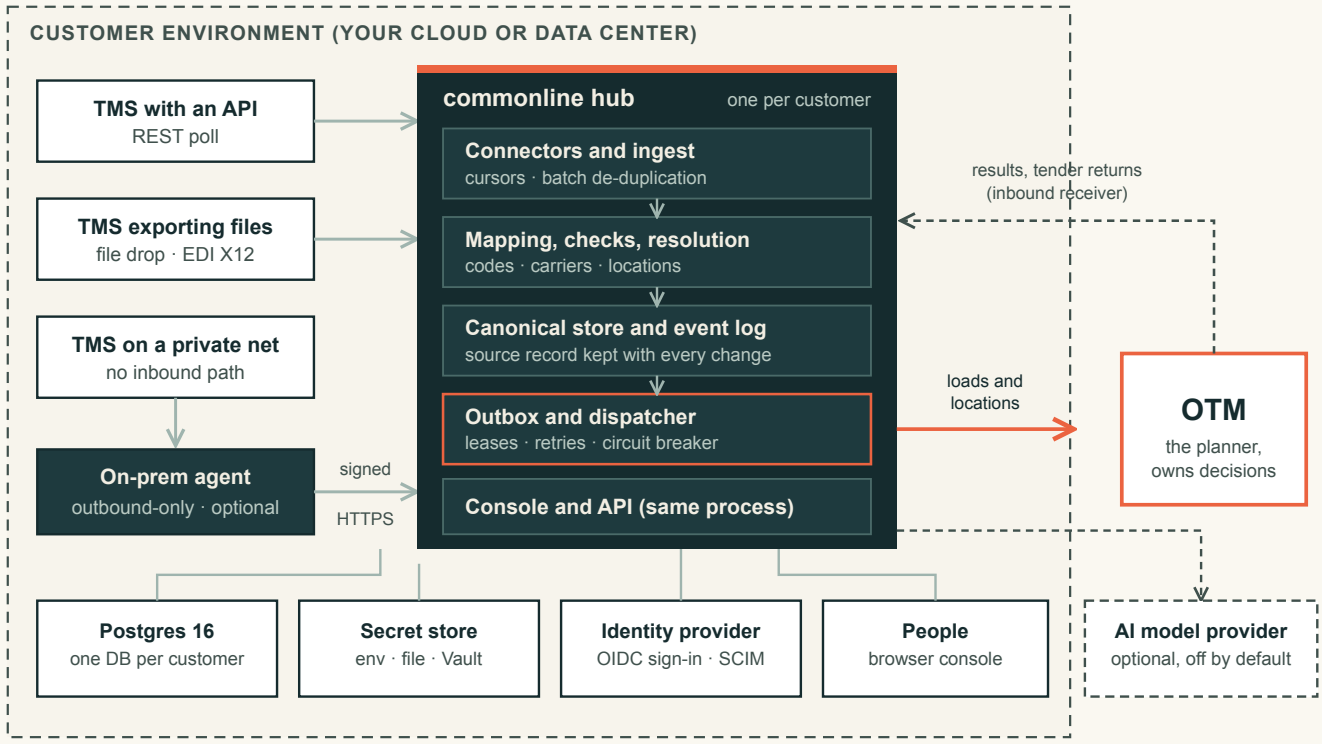

Figure 1. System context. Everything except OTM and the optional AI provider runs inside the customer's environment. The AI provider is called only when a customer sets their own key.

03 The data path, stage by stage

A record goes through the same stages whatever its source. Each stage commits its own state, so a crash between stages loses no acknowledged work and a replay lands in the same place.

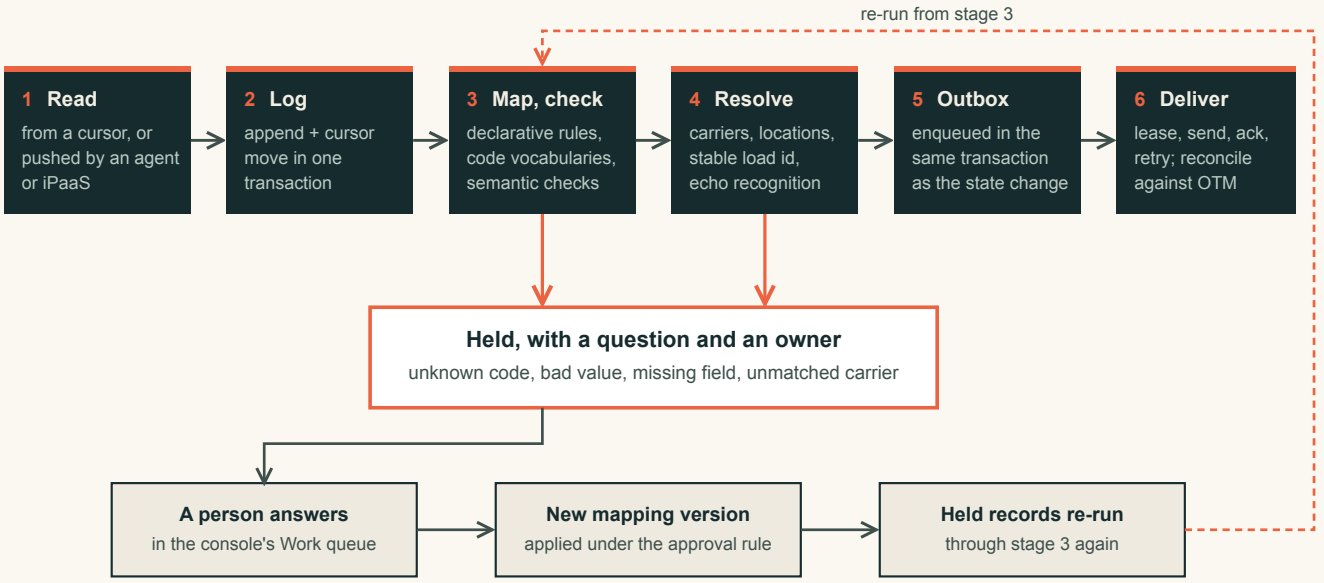

Figure 2. Data path. Coral arrows show the human decision loop. Records with blocking issues wait for an answer; an unknown charge code holds the charges while the load can proceed (section 5).

03 Data path: stage reference

The transaction boundary and recovery behavior at each step.

#	STAGE	WHAT HAPPENS	GUARANTEE
1	**Read**	Reader kinds: REST poll, file drop, EDI X12, and readers for McLeod Web APIs, TMW OData and Manhattan; plus push from an on-prem agent or an iPaaS. The optional agent buffers each batch in a local SQLite file before moving its cursor and pushes it over signed HTTPS. `SIMULATED` Vendor readers were written from public documentation; only the McLeod sandbox probes touched a real system. `LIVE, NARROW`	A batch is never dropped silently: the agent keeps it until the hub acknowledges it and says whether it was a duplicate.
2	**Log**	Records enter a partitioned, append-only event log kept in the database, with the raw source payload. Appending and moving the cursor are one transaction. Ingest de-duplicates by batch id and payload hash; the same id with a different payload is refused.	A batch is fully logged with its cursor advanced, or neither. Consumers are idempotent.
3	**Map, check**	A compiled declarative mapping turns the source record into the canonical form. Semantic checks look for zero stops, out-of-order stops, non-positive weight and a charge larger than the linehaul. A record with an issue is quarantined, not repaired.	Only an exact vocabulary match is applied automatically. Anything fuzzy or ambiguous becomes a question.
4	**Resolve**	Carriers match on MC, DOT or SCAC first, then by name: a high-confidence match applies, a medium one goes to a review queue. Locations are resolved per stop. The canonical load id is a hash of tenant, system and source id, so a restored hub produces the same ids. A load that was created by a business unit from our own tender is recognized when it comes back and is not sent to OTM twice.	Stable identity across restarts and replays. Only *open* loads go to the planner.
5	**Outbox**	The message to the destination is written in the same database transaction as the state change that caused it, together with a delivery-ledger entry that records what is owed.	No state change without its message, and no message without its state change.
6	**Deliver**	The dispatcher claims messages under a lease, sends through the OTM adapter, and applies the acknowledgment. See section 6.	At-least-once, with idempotency keys, bounded retries and a per-endpoint circuit breaker.

The hub's cycle pulls sources, replies and the target, processes records, pumps the dispatcher and processes again so that send results are applied in the same pass. A per-source ledger checks that received records equal converted, waiting and queued records, so a record cannot vanish between stages without showing up as a count that does not add up.

04 Canonical model and the per-record trail

Every source TMS is converted to one canonical form, so planning, the console and OTM never see a vendor's own shapes.

Entities

Load, asset (tractor or trailer), carrier, location, appointment, rate. A load carries ordered stops with windows and freight, charges of 16 types, and references to both the source id and our own.

Lifecycle

open, planned, tendered, covered, in transit, awaiting return, delivered, canceled. A pure state machine decides who owns each transition (origin, target or executor), so a status never moves on a guess.

Read-only entities

Appointments and rates are read, converted and shown, and are never delivered anywhere. Rate amounts need a separate permission and are not visible to viewers.

WHAT "CARRIED, HELD, LEFT OUT" MEANS IN THE PRODUCT

These words are the buyer's view; the code has a precise vocabulary behind them. Every field of every record gets a status in a field-by-field fidelity report:

- **carried**: converted and sent in the destination's message;
- **held**: waiting on a decision about this record;
- **not carried**, with a reason: no matching destination field, unverified against a real tenant, owned by the destination, not part of a tender, needs a decision, imprecise, or unreadable;
- **not read**: a source column that no rule reads, listed so omissions are visible.

THE TRAIL

For any load, the console shows its path through five stages (source, canonical, destination, execution, tender outcome), the instructions sent with their status, the reconciliation state, any held questions, and the fidelity report. The source record's raw payload is retained, so a record can be replayed through a changed mapping. The trail states that it is observed at runtime and does not imply that the destination accepted the load for planning.

05 Mapping, held records and the human loop

Mappings are data, not code. A mapping config per entity names, for each canonical field, a source path, a constant, a code vocabulary, a unit conversion, a time zone rule, a transform or a lookup. It is compiled once, with an interpreted twin kept for testing; both share the same checks.

FROM AN UNCLEAR VALUE TO A PERSON

1. The engine finds an issue: a required field is missing, a code is not in the table, or a value is bad. The message is written in plain words, for example that it does not know what a code means or that it could mean more than one thing.
2. The record is quarantined and a Work item opens in the console, with the affected records and an owner. Roster units with an unknown status or equipment code are shown as unconfirmed rather than held; an unknown charge code lets the load through and holds only its charges.
3. The question comes with options and, where available, a suggested answer marked as coming from the vocabulary, from a previous answer, or from the AI assistant. A suggestion is pre-filled and never applied by itself.
4. An answer becomes a proposed *mapping version*. A replay preview shows how many records would now fail and how many would change, before anything is applied. Each record is stamped with the mapping version it was processed under.
5. When the version is applied, every quarantined record is re-queued.

THE APPROVAL RULE

SITUATION	RESULT
The system is not live yet	Applies at once
Replay shows nothing now fails and nothing live changes	Applies; counted as safe
A person's change, under a one-person policy	Applies at once
Anything else, including every change proposed by the AI assistant, before or after go-live	Waits for approval by an integration admin; under a two-person policy the proposer cannot approve their own change

The two-person policy is the default when the hub runs with `NODE_ENV=production`, as the shipped image does. Demos and tests run one-person so that one person can walk the whole flow.

> **WHAT IS NOT BUILT**
>
> Drift detection compares a saved profile of a sample against a new one (new or missing fields, fill-rate drops, type changes, new values). It runs in the assessment tool and CLI. There is no continuous drift monitor on connected systems; at run time, a newly appearing unmapped code holds its records, which is the signal a person sees.

06 Delivery to OTM, acknowledgment and return path

DELIVERY GUARANTEES

- **Idempotent messages.** The outbox key encodes the subject, the content hash before and after, and the log position. A replay reproduces the same key and is skipped; a real change gets a new key. A newer snapshot of the same subject supersedes an older pending one.
- **At-least-once.** A claim is a 60-second lease that is extended while sending; on Postgres it uses an advisory lock and `SKIP LOCKED`. A result is applied only if the claim token still holds.
- **Bounded retries.** Backoff starts at 250 ms, doubles to a 60-second cap with jitter, and a message is parked as dead after 12 attempts, or at once on a non-retryable error. A person can revive it from the console.
- **Circuit breaker per endpoint.** After five consecutive failed batches the endpoint is left alone for five seconds, then probed with one message.

WHAT THE TESTS PROVE, AND WHAT THEY DO NOT

A test of repeated plans, lost acknowledgments, a crash mid-batch and a full log replay shows one instruction, one canonical load and one OTM release. That establishes stable keys and correct hub-side state in one process. It does not establish how many times a real receiver sees a message. Whether OTM or any source TMS de-duplicates on our message ids has not been shown. In local failover drills, some messages were delivered again after crashes and freezes, and none were lost.

ACKNOWLEDGMENT AND RECONCILIATION

A transport receipt is not treated as acceptance. A receipt-only acknowledgment marks the outbox row *sent* but leaves the reconciliation record *pending*. It becomes *confirmed* when a business result arrives (a plan for the load, a result received from OTM), *failed* on a rejection, and *needs review* if nothing arrives within 30 minutes. The console shows these states, so a request that returned success is not presented as acceptance.

THE OTM ADAPTER

MODE	WHAT IT DOES	EVIDENCE
Mock (default)	A mock OTM carries every entity kind, serves plans and de-duplicates on item id. All demos and tests use it.	`SIMULATED`
Real (`OTM_MODE=real`)	SOAP transmission of order releases and locations. Equipment, carriers, charges and status updates are refused with a permanent error until they are verified on a real tenant. Multi-stop is refused unless a REST route is configured. Request shapes come from public documentation and samples.	`SIMULATED` never run against a real OTM
Hybrid	SOAP for releases and locations, REST for carrier, equipment and status.	`NOT YET RUN` never run

So the first pilot's promise is narrow on purpose: loads and locations to a customer's OTM test instance. Trucks, trailers and carriers are converted and shown in one view, and are delivered only to the mock.

RETURN PATH

- **Planner results into the hub.** The real client does not poll. Results come through an inbound receiver, or an optional REST reader that is off by default and has only been exercised against fakes. The shape of OTM's tender offer in that reader is unverified.
- **Tenders back to a business unit's TMS.** Writers exist for an agent command channel, an iPaaS (canonical JSON or X12 204), McLeod order creation and EDI tender, and TMW order create. All are `SIMULATED`. Ambiguous outcomes are terminal for automation: a person or a read-back resolves them, and a write is not re-sent after an unclear POST without a duplicate-order lookup. This comes after the first connection and is not part of the first-pilot promise.

07 Storage and tenant isolation

- **One interface, two engines.** A single `Db` interface (run, get, all, transaction with savepoints) is implemented over SQLite, the default, and Postgres 16. The same suite runs against both. Postgres uses one connection, not a pool. On Postgres the schema is created at boot under an advisory lock; there are no migration files, and a build older than the database refuses to start.
- **One database per customer.** The tenant id is stored in the database and a mismatch refuses boot. Every hub data table carries a `tenant_id`. On Postgres, row-level security is enabled and forced with a policy bound at boot, so a missing setting fails closed; SQLite uses triggers. The hub refuses a superuser or BYPASSRLS role, and requires TLS to Postgres in production.
- **What isolation is not.** Shared-database multi-tenancy is not claimed. Row-level security is defense in depth behind the one-database-per-customer boundary, and a role that can run arbitrary SQL is outside what it protects against.
- **Business-unit scope fails closed.** A person with no business unit sees nothing, and a record with no unit is visible only to people scoped to every unit. The scope applies to loads, assets, Work items and messages, and fleet-wide connector views return 403 to one-unit users (ADR-0004).
- **Encryption at rest.** The application does not encrypt the database. Disk, volume or managed-database encryption is a deployment requirement, which the installer asks about (ADR-0001). Secrets can live in Vault (KV v2), in an AES-256-GCM file, or in environment variables. Backups are encrypted to the customer's public key when one is given at install.
- **Backups.** A script takes `pg_dump` snapshots and keeps 14 by default. The installer does not schedule it, so without a schedule there is no recovery point; there is no point-in-time recovery.

08 Deployment, availability and recovery

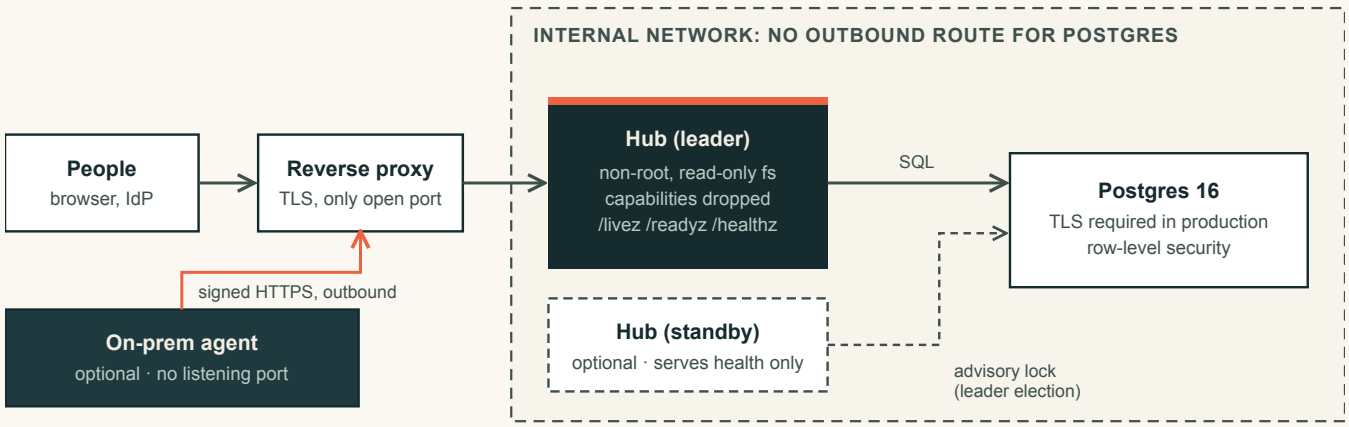

Figure 3. Customer-hosted install shape. Single hub by default; the standby is an option for Postgres-backed installs.

- **Image.** One non-root image on a digest-pinned Node base, with the package manager removed at runtime, a read-only root file system and a health check. The shipped compose file puts the hub, Postgres 16 and a reverse proxy on one host, with the database network marked internal. A Kubernetes base manifest and a two-replica variant with leader election are included. `LOCAL RUN` Compose was verified on a Mac under Colima and an amd64 emulation; Kubernetes on a local k3s cluster. No customer host has run it, and nothing has been deployed to a managed cloud.
- **Leader election** (off by default; needs Postgres). The leader holds a session advisory lock; a frozen leader is evicted when its idle session times out (5 seconds by default). A leader checks it still holds the lock before each cycle, poll and send. A standby serves health endpoints only.
- **Fails fast on database loss** (ADR-0002). There is no reconnect logic and no pool: the hub stops accepting work, answers 503 on everything but health, and exits with code 75 so the supervisor restarts it. Startup retries the first connection with backoff.
- **Outbound-only agent** (ADR-0003). The agent never listens. Its requests carry an HMAC over method, path, timestamp, nonce and body hash; replays are refused. Releases and configuration are Ed25519-signed.
- **Signed release images.** The pipeline builds multi-architecture hub and agent images with keyless signing, an SBOM and provenance. `NOT PUBLISHED` It is configured but has never published, so installs currently build from source.

WHAT WE MEASURED FOR RECOVERY `LOCAL RUN`

EVENT	RESULT
Hub killed, then ready again	Median about 0.5 s, at 2,000 and 20,000 loads, with a supervisor restarting it
Standby takes over from a killed leader (30 runs)	Median 0.74 s, 95th percentile 1.5 s
Standby takes over from a frozen leader (10 runs, 5 s lease)	Median 5.5 s
Acknowledged batches lost across hub kills	0 of 564 (10 kills) and 0 of 1,218 (20 kills)
Restore of a 202 MB database to a ready hub	About 1.8 s; the hub then waits for an operator to release the recovery hold

These ran on a busy developer machine with Postgres in Docker. They show the mechanism works; they are not an SLA. Not shown: managed-Postgres failover, real network partitions, a connection pooler, and what a receiver sees around a failover.

09 Security in brief

AREA	CONTROL
Sign-in	The customer's identity provider over OpenID Connect: authorization code with PKCE, RS256 ID tokens, issuer, audience, expiry and nonce checked. No local passwords outside demos. Tested against Keycloak with automated scenarios, and once against a free Okta developer org with one user; not against Entra or Ping.
Access	Four roles (viewer, ops admin, integration admin, transition lead), each scoped to business units from identity-provider groups; deny by default. SCIM 2.0 users and groups: deactivation takes effect on the next request and revokes sessions.
Sessions and web	Server-side sessions with only a hash of the id stored; 30 minutes idle, 12 hours absolute; HttpOnly, SameSite=Strict cookies; CSRF header on writes; CSP `default-src 'self'`, frames denied, rate and size limits.
Audit	An append-only log in which each entry's hash covers its content and the previous hash; database triggers refuse updates and deletes; optional signed checkpoints; search and export to NDJSON or CEF.
Machines	Agent requests are HMAC-signed with a five-minute window and replay nonces. Integration platforms use hashed bearer keys. OTM uses Basic or bearer authentication per tenant.
Supply chain	One runtime dependency; `npm audit` and image scanning in CI; actions pinned by commit SHA.

Position. No SOC 2, no ISO 27001, no external penetration test and no external audit; one maintainer; no production customer. A local ZAP scan found and fixed one low-severity issue; it is not a penetration test. The security pack lists each control with its source file and a status. Before a pilot with sensitive data we recommend an external test.

10 The optional AI assistant

- **Off unless the customer sets their own provider key.** Anthropic is the default provider and the only one run live; Bedrock, Azure OpenAI and OpenAI-compatible endpoints are implemented but untested live. An egress allowlist can restrict the hosts it may call.
- **What it sees.** A profile of fields, not records: field names, fill rates and distinct counts; free text and long digit strings reduced to shapes; ZIP codes cut to three digits. Short codes, states, short numbers and timestamps pass through as values. Text that looks like a prompt-injection attempt is replaced. The model provider is a subprocessor and receives this profile.
- **What it can do.** Propose mappings and answers, through the same path as a person, acting as an *agent*. Every agent change waits for a person's approval, before and after go-live, and the agent cannot approve anything. A refusal or failure degrades to rules only, and every call is audited with the model, token counts and what was sent (ADR-0005).
- **Default model.** `claude-sonnet-5-5` at low effort for mapping; overridable by configuration.

11 Console and tooling

The console is served by the hub from plain JavaScript with no build step, over one hand-written HTTP API. Every call needs sign-in, a permission and, for data, a business-unit scope.

SCREEN	WHAT A PERSON DOES THERE
Work	The queue of held records and delivery failures, each with an owner, the reason, and what to do next.
Records	List loads; open one to see its trail and the field-by-field fidelity report.
Capacity	Tractors and trailers reported by every connected system, with open orders against available equipment by city. Shown, never delivered.
Systems	Connect a system through a five-step wizard (credentials stay in the secret store; the console never sees them), review questions, mappings and connection health. **Assess a sample** profiles a CSV or JSON export with no credentials and no OTM access, orders decisions by their impact on the sample, and produces a downloadable report.
Audit	Search and export the audit log (needs the audit permission).

A command-line tool covers the same ground for engineers: onboard a never-seen system, assess a sample, replay a mapping against recorded records, reconcile, sign releases, purge a system's data, and export the audit log.

12 Measured numbers, with their conditions

MEASURE	RESULT	CONDITIONS
Load throughput, SQLite	About 4,800 loads per second on the up path; about 4,200 tenders per second; zero duplicates	One process, Apple M3 Pro, SQLite with WAL, 20,000 loads and 5,000 tenders; re-measured 2026-10-04
Load throughput, Postgres	About 206 loads per second; 147 tenders per second	One connection, 2026-09-30, before later read-ahead work that cut a 7,600-record ingest from about 46 s to about 37 to 40 s
Single tender latency	Median 0.7 ms on SQLite; about 15 to 45 ms on Postgres	Quiet system; shared developer machine
Soak	Two 15-minute runs, all invariants held	Postgres and SQLite, local, load average 4 to 13
Work-queue paging	Median 16 ms per page over 100,000 items	Synthetic, local SQLite; no capacity claim
Dependency audit	0 known vulnerabilities	`npm audit`, 2026-10-04
Local web scan	16,919 requests, no server errors; one low-severity issue fixed	OWASP ZAP 2.17.0 against a local harness, 2026-10-19; not a penetration test

These are per-process numbers on one machine. They are not a guaranteed rate in a customer's environment.

13 What is proven, and what a design partner would prove

AREA	STATUS	DETAIL
Conversion, mapping, work queue, trail, outbox, console	SIMULATED	Working code, tested end to end against simulated sources and a mock OTM.
Delivery to OTM	SIMULATED	Real adapter exists for loads and locations; never run against a real OTM. Single-process idempotency is proven; receiver-side de-duplication is not.
McLeod	LIVE, NARROW	Four read-only calls on one Track and Trace sandbox returned orders; the sandbox had no usable weight or equipment data. The Web APIs reader got a 401 and is unverified. Two disposable direct API probes were sent to the order-creation sandbox; the first was accepted and read back, and both stayed in a conditional status. These probes did not exercise Commonline's writer or establish business acceptance.
TMW, Manhattan, other readers and writers	SIMULATED	Written from public documentation, tested on fakes.
Sign-in	LIVE, NARROW	Keycloak with automated scenarios; one Okta developer org, one user, one day.
Availability	LOCAL RUN	Failover measured on a local cluster. Not shown on managed Postgres or with real partitions.
Compliance	NOT IN PLACE	No SOC 2, no ISO, no external penetration test, no production customer.

WHAT A DESIGN PARTNER WOULD PROVE, IN ORDER

1. **Source assessment.** Profile one acquired system's export, agree the field and code meanings that matter, and see what would be carried, held or left out. This needs no credentials and no OTM access.
2. **Acceptance by an OTM test instance.** Connect the real adapter to the customer's test instance, send an agreed sample, and check business-level results rather than transport receipts.
3. **Operations.** Measure the hours it takes on the customer's side and ours, and what a recovery looks like in their environment.

> **DECISIONS BEHIND THIS DESIGN**
>
> Five architecture decision records explain the choices that most affect a buyer: **0001** volume encryption plus customer-key backups, with column encryption deferred; **0002** fail fast and exit on database loss; **0003** an outbound-only agent; **0004** business-unit scope fails closed; **0005** the AI assistant's model and guardrails. They are in the repository's `docs/adr` folder and are available on request.

Prepared 2026-10-20 from the code at the time of writing and the repository's dated evidence notes. Every claim above is limited to what those show.
Contact: ruk@commonline.io.