Via One

Command Center — Master Manual

Version 1.0 · May 2026
English edition · Spanish edition published separately
Audience: internal operators, on-call engineers, system administrators, partner-facing support, distributor admins, compliance officers

Via One — Command Center

Master Manual

Version: 1.0

Document date: 2026-05-06

Audience: Internal Via One operators, on-call engineers, system administrators, partner-facing support, distributor admins, compliance officers

Languages available: English (this document), Spanish (separate edition)

How to read this document

This manual is structured for two reading modes:

- **Linear read** for new operators — go cover-to-cover, takes ~3 hours to absorb. Skim Parts I–III first, then dip into Parts IV–IX as you encounter the surfaces.
- **Reference lookup** for daily ops — use the Table of Contents or Appendix A (endpoint cheat sheet) to jump straight to what you need.

Each section ends with a small footer indicating when it was last reviewed (e.g. *Last reviewed: 2026-05-06 — v1.0*). When the underlying surface changes, the relevant section's date and version bump independently.

`[FIGURE X: description]` markers throughout indicate where screenshots should be inserted. Capture them from the live UI and replace the marker with the image.

Table of Contents

PART I — ORIENTATION

1. What is the Command Center?

The Via One Command Center is the **single web-based control plane** for the entire Via One platform. From one dashboard, an operator can:

- See live traffic across every tenant, every provider, every webhook
- Manage distributor accounts, balances, IP whitelists, and feature flags
- Run carrier-grade tests against TEMM cert, Altamira, MUWE, and other providers
- Review KYB onboarding applications and approve them
- Screen names against OFAC sanctions lists in real time
- Diagnose incidents using AI-powered NOC analysis
- Trigger kill-switches and rollbacks
- Watch the underlying ISO8583 / SOAP / REST messages flow, with full bodies

It is the hub. Every other Via One surface — the partner APIs, the bots, the escrow wallets, the OFAC engine, the reconciliation crons — feed into it or are managed from it.

[FIGURE 1: Command Center home screen — Brain Hub overview with tenant tiles + KPI cards]

The five tenants

Via One serves **five logical tenants**, each isolated from the others by row-level security and tenant-context middleware:

Code	Display name	Purpose
`OPTIMUS`	Latcom Horizons II (Optimus)	Telefónica MX distribution — the largest volume tenant, ~200K txn/mo target
`HAZ_GROUP`	HAZ Group	Latin-American multi-country distribution partner (HAZ Communications)
`RISE FINANCE`	RISE Holdings	DeFi remittance + wallet product (Movement chain primary, Aptos fallback)
`VIAONE`	Via One core	Direct-to-consumer Via One products (El Vecino, etc.)
`VIAONE_MASTER`	Via One Master	Cross-tenant admin tenant; system_admin role lives here

A given operator's view is automatically filtered by the tenants they have access to. A `system_admin` sees everything; a tenant-scoped admin sees only their tenant's data.

The 30+ provider integrations

Underneath the tenants, Via One calls a deep portfolio of upstream providers:

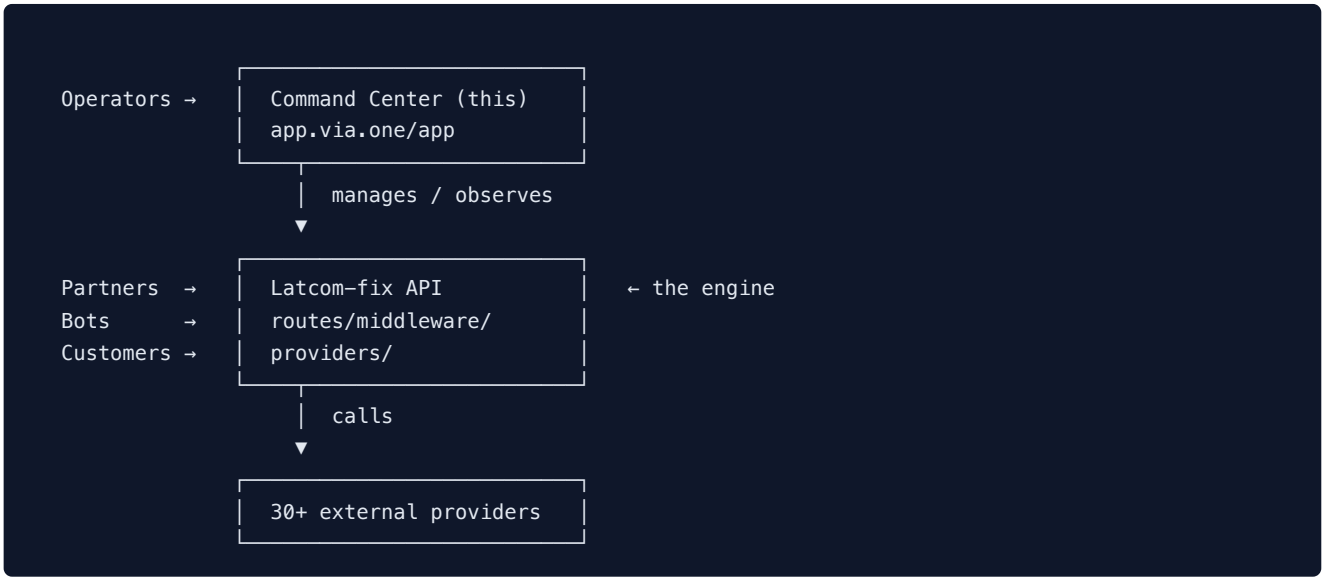
- **Mexico topup / paquete:** Telefónica SOAP, Telefónica ISO8583 (TEMM), Altamira, Codigo Arix, MUWE, Taecel, Altán
- **International topup:** PPN/ValueTopup, CellPay, Reloadly, NIHN (via PPN), Servipagos VZ

- **Payments:** Stripe, Pockyt (CashApp + APAC), MUWE bill pay
- **Blockchain wallets:** Privy (KMS), Movement, Aptos, Tron, Polygon
- **Messaging:** WhatsApp Cloud (Meta), Twilio, uContact, Telegram, Vonage carrier lookup, Tlaloc carrier detection
- **Compliance / data / AI:** treasury.gov OFAC, ExchangeRate-API, Anthropic Claude, SendGrid email
- **Storage:** AWS S3, DigitalOcean Spaces

Each is monitored by the Command Center's provider-health surface (Section 10) and exposed through the Live Transaction Console (Section 28).

[FIGURE 2: Architecture diagram — tenants on top, Command Center in middle, providers as a fan-out at the bottom]

Where the Command Center fits



Section last reviewed: 2026-05-06 — v1.0

---

2. Who uses it, and how

Roles

The Command Center supports five distinct roles. Each maps to a `users.role_code` value in the database.

Role	Code	Cross-tenant view?	Typical user
System Admin	`system_admin`	✅ Yes — all 5 tenants	Via One platform engineers, on-call, founders
Distributor	`distributor`	❌ Single tenant only	Distributor company admins (e.g. HAZ ops)
Reseller	`reseller`	❌ Single tenant only	Sub-resellers under a distributor
Support Agent	`support_agent`	✅ Read-only across tenants	Internal customer support
Compliance Officer	`compliance_officer`	✅ Read-only across tenants + OFAC write	Compliance / KYB review

Permission boundaries

A `system_admin` can do anything. The other roles have strict guardrails enforced server-side:

- **Distributor / Reseller** — sees only their own tenant's customers, transactions, balances. Cannot create or modify other tenants' data.
- **Support Agent** — read-only across all tenants. Can view customer details and transaction history but cannot edit balances, flip flags, or approve KYB.
- **Compliance Officer** — read-only across all tenants, plus write access to OFAC screening logs and KYB approvals.

These boundaries are enforced at the route level in `latcom-fix` (search `requireRole`, `requireSystemAdmin` middleware). Bypassing them requires changing the role on the user account, not just the UI.

Recommended Day-1 walkthrough for a new admin

If you're a new system_admin getting onboarded:

1. **Log in**, confirm you can see all 5 tenants on the Brain Hub home page.
2. **Open the Live Transaction Console** (Section 28) and watch live traffic for 5 minutes — get a feel for the rhythm of inbound/outbound/webhook activity.
3. **Click into one tenant** (start with OPTIMUS, the largest). Browse its customer roster (Section 7), one customer's transaction history (Section 18), the route mappings (Section 10).
4. **Open Provider Health** (Section 10). Confirm all primary providers are green. Note the smoke-test cadence.
5. **Open Testing Command Center** (Part III). Run one ISO8583 simulation (no real money) to learn the message inspector.
6. **Open NOC** (Part V). Read the most recent 10 alerts. If any are unresolved, ask which to acknowledge.
7. **Read Part VIII (Deployment & Release Ops)** before pushing any code.
8. **Bookmark Part X-A** (endpoint cheat sheet) and Part X-G (common SQL queries).

After Day 1, expect Days 2-7 to cover specific surfaces as you encounter them. Don't try to memorize the whole manual — use the Table of Contents.

[FIGURE 3: Role permission matrix — visual]

Section last reviewed: 2026-05-06 — v1.0

---

3. How to access

URLs

Environment	Command Center URL
Production	`https://app.via.one/app`
Staging	`https://latcom-fix-staging-staging.up.railway.app/app` (when needed for testing)

The same `app.via.one` host also serves several aliases mapped to the same underlying UI:

- `optimus.via.one` — branded Optimus view (same Command Center, OPTIMUS tenant context auto-set via subdomain)
- `dash.via.one`, `cmd.via.one`, `ops.via.one`, `my.via.one` — alternate alias entry points

Login flow

1. Navigate to the URL. You'll land on a login form.
2. Enter your `email` (or `customer_id` for legacy distributor accounts) and `portal_password`.
3. On success, the server issues a JWT (HS256, 1-hour expiry) stored in browser local storage.
4. The JWT is sent on every subsequent API call as `Authorization: Bearer `.

[FIGURE 4: Login screen with annotated fields]

2FA / IP whitelist

- **2FA** — currently optional, configurable per user. When enabled, after password the user is challenged for a TOTP code.
- **IP whitelist** — for distributor-tier accounts that opt in. When `customers.ip_whitelist_enabled = TRUE`, login attempts from IPs outside `customers.allowed_ips` are rejected with HTTP 403.

If you're a system_admin and your IP changed (working from a new café), the IP-whitelist setting on YOUR user account is unlikely to be set — system_admin accounts typically don't IP-whitelist. If you're locked out, see "Recovering a locked-out admin" below.

Recovering a locked-out admin

If you can't log in (forgot password, account disabled, etc.) and you're the only system_admin awake:

1. **Get DB access** — `psql $DATABASE_PUBLIC_URL` (the prod DB, see Appendix E for env vars).
2. **Reset your password hash:**

``` `sql UPDATE users SET portal_password_hash = '$2b$10$' WHERE email = 'you@viaone.com';` ``` Use any bcrypt tool to generate the hash; cost factor 10.

1. **Re-enable the account if disabled:**

``` `sql UPDATE users SET status = 'active' WHERE email = 'you@viaone.com';` ```

1. **Log in with the new password** within 5 minutes (tenant cache TTL).

If you're not the only system_admin, ask another admin to do it for you via the Brain Hub user-management surface (Section 24).

Section last reviewed: 2026-05-06 — v1.0

---

4. Glossary up front

You'll meet these terms throughout the manual. Memorize them now and the rest reads faster.

Term	Meaning
Tenant	A logical partner organization. The 5 tenants are OPTIMUS, HAZ_GROUP, RISE FINANCE, VIAONE, VIAONE_MASTER. Each has its own customers, route mappings, balances, branding.
Customer / distributor	An individual API consumer within a tenant. `latcomdigital` is a customer under OPTIMUS. The two terms are used interchangeably; the DB column is `customers.customer_id`.
Reseller	A sub-account under a distributor (rare; not all tenants use it).
Partner	Used informally as a synonym for tenant or customer, depending on context.
Provider	An external upstream we call to fulfill transactions (Telefónica, Altamira, PPN, etc.).
Operator	A telecom carrier (Telcel, Movistar, AT&T, Unefon, Bait, Altán). Mapped to providers via route_mappings.
SKU	A product identifier. We have multiple SKU namespaces: `latcom_sku_id` (catalog-facing), `operator_sku_id` (per-provider wire SKU), `vendor_sku` (override for cases where the wire differs from the operator default).
route_mapping	A row in `route_mappings` that says "for tenant X, operator Y, service Z, route to provider P (with failover to Q)." This is the routing brain.
F39	ISO8583 response code field. `00` = approved. Others = decline / error states. See Appendix B for the dictionary.
STAN	System Trace Audit Number. 6-digit ID Telefónica's gateway assigns each transaction.
RRN	Retrieval Reference Number. 12-digit reference for cross-system tracking.
MSISDN	Mobile Subscriber ISDN — phone number, in our codebase typically a 10-digit MX number with no country code.
TEMM	Telefónica Mobile México platform — the ISO8583 endpoint at `10.225.236.72:7903` (prod) or `10.225.244.79:7903` (cert).
LATJAVA	The on-premise gateway VM at `10.10.2.5` / `66.231.242.91` that proxies between Railway and Telefónica's intranet.
Cert mode	Telefónica's certification environment. When `PAYMENTS_CERT_MODE=true`, paquete attempts get rewritten at the wire to known-good cert SKU (`code=21, amount=150 MXN`) to enable end-to-end testing.
Idempotency key	A client-supplied value (`Idempotency-Key` header, or `dist_transid` for LATCOM-format calls) that prevents double-processing of retries. Stored in `provider_transactions.idempotency_key` and `idempotency_keys` table.
dist_transid	The distributor's transaction ID — unique per dist + per call. Used to derive the idempotency key when no header is supplied.
escrow	A held-funds account for non-custodial wallet flows (RISE primarily). Separate DB.
Kill-switch	A SQL UPDATE that reverts a feature flag. The most common is rolling back `payments_iso8583_enabled = FALSE` on flipped distributors. Documented in Section 33.
Brain Hub	The cross-tenant master view inside the Command Center. Section 5 onward.
NOC	Network Operations Center — the AI-powered incident diagnosis surface. Part V.
Bot	A WhatsApp / Telegram / Siri / RCS / iMessage conversational agent. They share a common orchestrator but have separate personas (El Vecino, ViaOne, request-bot).

Section last reviewed: 2026-05-06 — v1.0

PART II — BRAIN HUB

The cross-tenant master view. This is what the system_admin role sees when logging in. It's the most-used surface for daily ops.

---

5. Brain Hub overview screen

When you log in as system_admin, the first screen you see is the Brain Hub home.

[FIGURE 5: Brain Hub home — full screen]

Layout

The Brain Hub home is a 3-column layout:

Left column — Navigation rail (always visible)

- Brain Hub (you are here)
- Testing Command Center
- CRM
- Wallets
- Payouts
- Brain Hub > Compliance
- Brain Hub > KYB
- Brain Hub > Tenants
- Brain Hub > User Management
- NOC

Center — Tenant tiles + KPI cards The 5 tenants are shown as clickable tiles. Each tile shows:

- Tenant code + branded color
- Today's transaction count
- Today's transaction volume (USD)
- Provider health (green/yellow/red)
- Active alerts count

Clicking a tile drills into that tenant's view.

Below the tenant tiles, four global KPI cards:

- Total transactions (24h)
- Total volume (24h)
- Active customers (last login < 24h)
- Open NOC alerts

Right rail — Live alert feed Scrolling list of the last 24h of NOC alerts. Newest at the top. Color-coded by severity. Click an alert to expand and acknowledge.

Header KPI cards explained

Each KPI card auto-refreshes every 60 seconds.

Card	Source	What it tells you
Total transactions (24h)	`SELECT COUNT(*) FROM transactions WHERE created_at >= NOW() - INTERVAL '24 hours'`	How busy the platform has been
Total volume (24h)	`SUM(customer_amount)` for the same window	Real revenue traffic
Active customers (24h)	`COUNT DISTINCT customer_id` from `auth_logs` last 24h	Who's actually integrating today
Open NOC alerts	`noc_alerts WHERE acknowledged_at IS NULL`	Things that need eyes

How to read at a glance

- **All tiles green + low alert count** = platform healthy, no action needed.
- **One tile yellow** = some provider for that tenant is degraded; click in for detail.
- **Red KPI card** = something's spiking. Open the matching tenant first, then NOC for context.

[FIGURE 6: Tenant tiles closeup — annotated with what each number means]

Switching tenant context

Clicking a tenant tile sets your active tenant context. You'll see this reflected in:

- The header bar (now shows the tenant name + branded color stripe)
- Customer lists, route mappings, transactions — all auto-filtered to that tenant
- Provider health — shows only providers active for that tenant
- Returning to "All tenants" view: click the Via One logo top-left or the Brain Hub link.

Section last reviewed: 2026-05-06 — v1.0

---

6. Tenant management

Tenants are the top-level partition. Most operators won't create a new tenant often — it's a Lifecycle Event. But they will frequently edit settings within a tenant.

The `tenants` table — what each column means

Column	Type	What it controls
`id`	UUID	Internal primary key
`tenant_code`	VARCHAR(50) UNIQUE	Short stable code (`OPTIMUS` , `HAZ_GROUP`) — referenced everywhere
`company_name`	VARCHAR(255)	Display name shown in UIs
`subdomain`	VARCHAR(100) UNIQUE	First-label of hostname (e.g. `optimus` matches `optimus.via.one`)
`custom_domain`	VARCHAR(255)	Full custom domain for partners with their own DNS (e.g. `buprolat.latcom.co`). One per row.
`contact_email` , `contact_phone`	VARCHAR	Account contact info
`branding_config`	JSONB	Logo URL, primary/secondary colors, custom CSS — used to style the tenant's view
`settings`	JSONB	Per-tenant operational settings (timezone, currency, etc.)
`feature_flags`	JSONB	Booleans for tenant-level feature toggles (Section 25)
`plan`	VARCHAR	`standard` or `professional` — controls feature availability
`status`	VARCHAR	`active` / `deactivated`
`created_at` , `updated_at`	TIMESTAMPTZ	Audit

Creating a new tenant

[FIGURE 7: Tenant creation form — Brain Hub > Tenants > New]

Step-by-step:

1. **Brain Hub > Tenants > New Tenant**
2. Fill in:

- Tenant code (UPPERCASE, no spaces)
- Display name
- Subdomain (the first-label they'll use, e.g. `acme` for `acme.via.one`)
- Optional: custom domain (if they bring their own)
- Plan tier

1. Set initial branding (logo, colors) — can edit later
2. Set initial settings JSON (timezone, currency)
3. Click Create.
4. On creation, the system auto-provisions:

- Default route mappings (none — must be added per Section 10)
- Default feature flags (all off)
- An admin user record (you'll get the temp credentials)

After create, you'll need to add at least one customer (Section 7) and at least one route mapping (Section 10) before the tenant is operational.

Subdomain + custom_domain wiring

Tenant resolution at request time happens in `middleware/tenant-context.js`:

1. The middleware reads the incoming hostname.
2. It extracts the first label (`optimus` from `optimus.via.one`).
3. Queries `SELECT ... FROM tenants WHERE subdomain = $1 AND status = 'active'`.
4. If no match, falls back to: `WHERE custom_domain = $1`.
5. Sets `req.tenant` and `req.tenantId` on the request.

For `*.via.one` subdomains — just set `subdomain` on the tenant row. No DNS work needed (we already own the via.one wildcard).

For partner-bring-your-own-domain (e.g. `buprolat.latcom.co`):

1. Add the new domain as a Railway custom domain (via `railway` CLI or GraphQL — see Section 32 for the full procedure).
2. Railway returns a CNAME target (e.g. `61vobyc4.up.railway.app`) and a TXT verification value.
3. Partner adds the CNAME and TXT to their DNS.
4. Once Railway issues the cert (~1-3 min after DNS propagates), update the tenant row:

``` `sql UPDATE tenants SET custom_domain = 'buprolat.latcom.co' WHERE tenant_code = 'OPTIMUS';` ```

5. The tenant cache invalidates in 5 min OR after API restart.

⚠️ **Schema constraint:** `custom_domain` is a single column, not an array. If a tenant needs multiple custom domains, add additional tenant rows with the same downstream config OR (preferred) extend the resolver to support a `tenant_aliases` table. Not in v1.

Activating / deactivating

To deactivate a tenant:

```
UPDATE tenants SET status = 'deactivated' WHERE tenant_code = 'X';
```

Effects:

- Tenant resolution will skip the row (queries filter `status = 'active'`)
- New requests to that tenant's hostname → "Tenant context required" 400 error
- Existing JWTs from that tenant continue to work until they expire (the per-request DB lookup happens, but the row is excluded)

To restore: flip back to `active`.
```

Plan tiers

Plan	Enables
`standard`	Basic distributor API, single-currency wallet, default OFAC, manual reconciliation
`professional`	Dual-currency wallet (USD + MXN), webhook delivery queue, advanced OFAC, priority support, custom domain support

Setting `plan = 'professional'` is the gate; the actual feature availability depends on individual `feature_flags` (Section 25).

[FIGURE 8: Tenant edit screen — annotated]

Section last reviewed: 2026-05-06 — v1.0

7. Customer (distributor) management within a tenant

Customers are individual API consumers within a tenant. A tenant can have hundreds. Each has their own credentials, balances, IP whitelist, and feature flags.

The `customers` table

Column	What it does
`id`	Internal UUID
`customer_id`	The customer's stable identifier (`latcomdigital` , `HAZ_001` , `LATCONECTA_001`) — referenced in API calls
`company_name`	Display name
`api_key`	Long-form API key for x-api-key auth
`secret_key`	Password for `/login` and `/dislogin` flows
`tenant_id`	FK to `tenants.id`
`current_balance`	USD balance (the primary wallet)
`balance_mxn`	MXN balance (when dual-currency wallet is enabled)
`reserved_balance` , `reserved_balance_mxn`	Held funds for in-flight transactions
`credit_limit`	Maximum allowed negative balance (most distributors = 0)
`is_active`	Boolean — turn off to disable login
`discount_percentage`	Off-list discount applied to their pricing
`commission_percentage`	What we pay them as a sub-distributor (rare)
`allowed_ips`	JSONB array of IPs they can call from
`ip_whitelist_enabled`	Boolean — if FALSE, ignore `allowed_ips`
`auth_mode`	`api_key` or `jwt`
`payments_iso8583_enabled`	Per-customer flag to redirect Movistar traffic to the new payments-iso8583 path
`provider_routing`	JSONB — per-customer routing override (rarely used)
`allowed_countries`	JSONB array of ISO country codes they can transact for
`last_login_at`	Most recent successful login
`portal_password`	UI password (separate from API secret_key)
`minimum_alert_balance`	Below this, generate a low-balance alert

Customer roster table

[FIGURE 9: Customer roster table — Brain Hub > Customers]

Columns visible:

- `customer_id` , `company_name` , `is_active` (green/grey badge), `current_balance` , `balance_mxn` , `last_login_at` , action menu (•••)

Actions on each row:

- View — opens the customer detail page
- Edit — opens the edit form (Section 7.4)

- View transactions — drills into Section 18 filtered to this customer
- Top up balance — Section 20.3
- Deactivate / Reactivate
- Reset secret_key (regenerate)
- View auth logs

Creating a new customer (distributor)

[FIGURE 10: New customer creation form]

1. **Brain Hub > Customers > New Customer**
2. Fill in:

- `customer_id` (UPPERCASE convention, no spaces — `LATCOM_PERU_001`)
- `company_name` (display name)
- Initial `current_balance` (default 0)
- Initial `balance_mxn` (default 0; only relevant if `professional` plan)
- `discount_percentage` (default 0)
- `auth_mode` (default `api_key`)

1. **Generate keys:**

- Click "Generate api_key" → a 32+ char random string is created
- Click "Generate secret_key" → another random string for `/login` / `/dislogin`
- **Important:** these are shown ONCE. Copy them and send to the partner securely. Store yourself in 1Password.

1. **IP whitelist:** if the partner will call from known IPs, paste them now. Otherwise leave blank and disable the toggle.
2. **Feature flags:** leave default (all off). Toggle later as needed.
3. Click Create.

The customer can now log in via `POST /api/dislogin` (or `/login`) using their `customer_id` + `secret_key`.

Editing a customer (the safe vs dangerous fields)

[FIGURE 11: Customer edit screen — fields highlighted by danger level]

Safe to edit live:

- `company_name` (display only)
- `discount_percentage` (affects new transactions, not in-flight)
- `commission_percentage`
- `minimum_alert_balance`
- `allowed_ips` (toggle on)
- `ip_whitelist_enabled` (toggle off if rotating IPs)

Edit with caution:

- `current_balance` / `balance_mxn` — manual adjustment requires audit trail (Section 20.3 — never edit directly via SQL without a journal entry)

- `auth_mode` — switching forces all existing sessions to break
- `portal_password` — partner has to be informed of new credentials

Dangerous — coordinate with partner first:

- `is_active` (deactivating immediately blocks login)
- `secret_key` regeneration (existing scripts break instantly)
- `payments_iso8583_enabled` flip (changes routing path mid-flight; ensure no in-flight transactions before flipping)

Activating / deactivating / blocking

Three states for a customer:

State	Set via	Effect
Active	`is_active = TRUE`, `status = 'activated'`	Normal operation
Deactivated	`is_active = FALSE`	Login attempts return 403 immediately
Blocked	`is_active = FALSE`, `status = 'blocked'`	Same as deactivated, but the `blocked` status flag is used by NOC to suppress alerts (per the KKSQUARED kill pattern, May 1)

Blocking pattern (the KKSQUARED case): A customer that's generating noise (sparse traffic + alert spam) but you don't want to delete:

```
UPDATE customers
SET is_active = FALSE, status = 'blocked'
WHERE customer_id = 'KKSQUARED_001';
```

To re-enable:

```
UPDATE customers
SET is_active = TRUE, status = 'activated'
WHERE customer_id = 'KKSQUARED_001';
```

Topping up a customer's balance

Manual credits should always go through the audit trail (`balance_journal` table) rather than direct UPDATE. From the Command Center:

[FIGURE 12: Top-up balance modal — Brain Hub > Customer > Top up]

1. Open the customer
2. Click "Top up balance"
3. Choose currency (USD / MXN)
4. Enter amount and reason (free text)
5. Confirm
6. The system:

- Inserts a `balance_journal` row (kind=`manual_credit`, who=you, when=now, why=reason)
- Updates `customers.current_balance` (or `balance_mxn`)
- Generates an audit log entry visible in the customer's history

For programmatic credits (CRON, partner top-up flow), use the same internal journal helper — never raw UPDATE.

Section last reviewed: 2026-05-06 — v1.0

---

8. KYB onboarding queue

KYB (Know Your Business) is the partner intake flow. New partners apply via `onboarding.relier.group`, and the Command Center is where Via One operators review and approve.

The `kyb_applications` table

Column	What it tracks
`id`	UUID
`legal_name`, `dba`, `tax_id`	Business identity
`country`, `state`, `address`	Location
`contact_email`, `contact_phone`	Primary contact
`business_type`	LLC, SA de CV, etc.
`tenant_id_assigned`	If approved, which tenant the resulting customer is created under
`customer_id_assigned`	Created on approval
`status`	DRAFT / SUBMITTED / IN_REVIEW / APPROVED / REJECTED / REQUIRES_INFO
`submitted_at`, `reviewed_at`, `decided_at`	Audit timestamps
`reviewer_user_id`	Who decided
`decision_reason`	Free text
`documents`	JSONB array of uploaded doc references (S3 keys)
`ofac_screening_result`	JSONB — pre-screening result
`created_at`, `updated_at`	Audit

KYB statuses explained

Status	Meaning	Next step
`DRAFT`	Applicant started, hasn't submitted	None — wait or follow up
`SUBMITTED`	Applicant clicked Submit; documents uploaded	Reviewer should pick up
`IN_REVIEW`	A reviewer has claimed it	Reviewer continues
`REQUIRES_INFO`	Reviewer needs more docs/clarification	Applicant updates and resubmits
`APPROVED`	Tenant + customer record provisioned	Operational; partner can integrate
`REJECTED`	Decided against	Applicant notified; archive

Reviewing an application

[FIGURE 13: KYB application detail — Brain Hub > KYB > Application]

1. Open Brain Hub > KYB Onboarding queue
2. Filter to `SUBMITTED`
3. Click into the oldest one
4. The detail page shows:

- All form fields the applicant submitted
- Uploaded documents (click to download/preview)
- Pre-screening: OFAC match status, business registry validation, tax ID check
- Reviewer notes pane

1. Either:

- **Approve** → trigger tenant creation (or attach to existing tenant) + customer creation
- **Reject** → state reason; applicant notified
- **Requires info** → write what's needed; applicant gets emailed; status flips

1. On approve, the system auto-provisions:

- Customer record under the assigned tenant
- Initial balances at 0
- Welcome email with API credentials (system_admin sees credentials in the dashboard for handoff)

OFAC pre-screening

Every KYB application is auto-screened against OFAC the moment it's submitted. The result is stored in `kyb_applications.ofac_screening_result`.

If a match is found:

- The application status auto-flips to `IN_REVIEW`
- A red banner appears at the top of the application detail page
- The reviewer must explicitly note in `decision_reason` why they're approving despite the match (e.g., common name, false positive)

See Section 9 for full OFAC behavior.

[FIGURE 14: KYB application with OFAC match warning]

Section last reviewed: 2026-05-06 — v1.0

---

9. Compliance & OFAC

Via One operates its own OFAC screening engine — replacing the third-party `ofac-api.com` integration as of May 1 2026.

How it works

[FIGURE 15: OFAC compliance dashboard — Brain Hub > Compliance > OFAC]

Daily at **04:30 UTC**, a cron job pulls the latest OFAC sanctions lists directly from `treasury.gov`:

- SDN list (Specially Designated Nationals)
- Consolidated non-SDN list

These are loaded into local Postgres tables:

- `ofac_entries` — primary sanctioned individuals/entities
- `ofac_alt_names` — alternative names + aliases
- `ofac_addresses` — known addresses
- `ofac_ids` — known passport / SSN / national ID numbers
- `ofac_sync_metadata` — sync status (date, success, record count)
- `ofac_screening_log` — audit trail of every screening event

The 3-tier matcher

When a name is screened (via KYB submit, customer onboarding, or manual screen), the matcher runs three tiers in sequence:

1. **Exact match (score 100)** — input matches `primary_name_normalized` or `alt_name_normalized` after lowercasing + accent-stripping
2. **Token-set match (score 95)** — input has all the same name tokens as a sanction record, regardless of order ("Putin Vladimir" → matches "Vladimir Putin")
3. **Soundex + Levenshtein (score 50–85)** — phonetic match catches misspellings ("Vladimir Poutine" → high-score match to "Vladimir Putin")

A score ≥ 70 triggers a hold (manual review required); ≥ 90 triggers an automatic block.

When screening happens

Event	Where it fires
KYB submission	`kyb_applications` insert hook
Wallet creation	`wallets` / `ev_accounts` insert hook
Topup with new beneficiary	optional, configurable per tenant
Bill payment with new payee	optional, configurable per tenant
Manual screening	UI tool — Brain Hub > Compliance > Screen a name

Every screening event is logged in `ofac_screening_log` with: input name + country, context (wallet_signup, topup, manual), match count, top match score, blocked yes/no, timestamp.

Admin endpoints

Endpoint	What it does
`GET /admin/ofac/status`	Sync metadata for both lists (when last refreshed, record count, success/error)
`POST /admin/ofac/sync?list=sdn`	Manually re-pull one list (don't normally need to; cron runs daily)
`POST /admin/ofac/sync-all`	Refresh both lists
`POST /admin/ofac/screen`	Test-screen a name. Body: `{ name, country, context }`. Returns matches + scores. Audit-logged.
`GET /admin/ofac/screening-log?since=&limit=`	Recent screening events

Manual screening tool

[FIGURE 16: OFAC manual screening tool — annotated]

For a one-off check (a partner asks "are these 3 names cleared?"):

1. Brain Hub > Compliance > OFAC > Screen a Name
2. Enter the name (full name as it would appear on a doc)
3. Optionally narrow by country
4. Optionally specify the context (wallet_signup, topup, manual_search) — affects the audit log
5. Click Screen
6. Results appear:

- Top 5 matches with scores
- Per-match: full sanctioned name, list (SDN/non-SDN), programs, remarks
- Decision: APPROVED (no match ≥ 70), HOLD (70–89), BLOCKED (≥ 90)

When a match fires

What happens depends on context:

- **KYB application** — flips to `IN_REVIEW`, red banner, reviewer must explicitly approve in writing
- **Wallet signup** — wallet creation rejected; user sees "We were unable to verify your identity at this time"
- **Topup** — transaction blocked at the API layer; partner gets `error_code=OFAC_HOLD`
- **Manual** — informational only; reviewer decides next step

In all cases, an audit row is written to `ofac_screening_log` with `blocked=true` so a regulator can see the full record.

Section last reviewed: 2026-05-06 — v1.0

---

10. Provider health & smoke tests

Provider health is monitored on three levels: **liveness** (always running), **catalog audit** (drift detection), and **synthetic transactions** (real wire tests).

`/health/providers` overview

[FIGURE 17: Provider health gauge dashboard — full screen]

The provider health page is at `/health/providers`. It shows, per provider:

- **Status** — `ok` / `degraded` / `error`
- **Last check** — when we last got a successful health response
- **Catalog audit** — when we last validated the live catalog matches our DB
- **Smoke test** — last synthetic transaction result
- **Avg latency (5min)** — rolling latency profile

Providers monitored:

- CellPay (US topups, gift cards)
- CodigoArix (MX paquetes)
- PPN / ValueTopup (international)
- Reloadly (international fallback)
- Altamira / Telefónica MX (recargas + paquetes)
- NIHN (US Page Plus, MobileX)
- MUWE (MX bill pay, SPEI)
- Pockyt (CashApp, APAC)
- Stripe (cards, ACH)

Catalog audit — drift detection

Every hour at `:17` past, the `provider-catalog-audit` cron checks each provider's live catalog against our `latcom_products` / `operator_products` rows.

What counts as a "drift":

- A SKU exists in the live provider catalog but not in our DB (orphan in live)
- A SKU exists in our DB but not in the live catalog (orphan in DB — could be deprecated)
- The price differs (we have $7, live says $12 — bug pattern from May 1 NIHN issue)

Drifts are logged as warnings in the cron output and surface in the provider-health gauge as `degraded`. The audit also runs with a 15-minute drift detector (cron `3,18,33,48`) for tighter coverage.

Smoke tests — synthetic transactions

Two flavors:

Smoke type	What it does	Cost	Schedule
Soft (free)	Posts an invalid input (e.g., bogus phone) — provider responds with a known error code, proving the wire works	Free	Hourly
Hard ($1)	Real transaction with a real $1 amount — proves end-to-end including provider settlement	$1 per provider per run	On-demand only (gated)

Smoke results land in `provider_smoke_log` and feed the gauge.

To trigger a hard smoke manually:

- Brain Hub > Testing Command Center > Provider Smoke Test
- Pick provider + smoke type
- Click Run
- Result appears within ~10 seconds

Reading the gauge

[FIGURE 18: Provider health card — green/yellow/red states annotated]

- **Green** — last 5 checks all OK, last smoke green, no audit drift
- **Yellow** — last check OK but a recent failure happened, or audit drift detected
- **Red** — current check failing OR last 3+ failures in a row

If a provider is yellow/red, click into it for the full timeline. NOC alerts will already be firing for sustained reds.

When a provider goes dark

- The route_mappings layer auto-fails over to the failover provider (if one is configured)
- Customers calling that provider's primary route see the failover response
- An NOC alert is generated (Section 22)
- WhatsApp ops alert lands within ~60 seconds

Recovery is usually provider-side; we just observe. See Section 23 for the playbook.

Section last reviewed: 2026-05-06 — v1.0

---

11. Webhook delivery queue

Via One sends webhooks to partners whenever an order changes state (delivered, failed, partial, reversed). Delivery is durable — backed by the `webhook_deliveries` table — so a partner can be down for hours without losing notifications.

The state machine

```
pending → delivering → delivered ✅
                     → failed_retrying → (more attempts) → delivered ✅
                                       → dead 💀 (after 5 attempts)
```

State	Meaning
`pending`	Just enqueued; awaiting first attempt
`delivering`	Worker has picked it up, HTTP request in flight
`delivered`	Partner returned 2xx
`failed_retrying`	Last attempt failed; another scheduled
`dead`	All 5 attempts failed; dead-letter
`discarded`	Manually marked as not-to-retry (e.g., partner permanently offline)

Retry schedule

Five attempts total, spaced:

```
Attempt 1: immediately
Attempt 2: +2 seconds
Attempt 3: +8 seconds
Attempt 4: +30 seconds
Attempt 5: +2 minutes
After 5 failures: dead (no more retries unless manual replay)
```

The worker (`cron/webhook-delivery-worker.js`) drains the queue every 30 seconds, batching up to 25 deliveries per tick.

HMAC signature spec

Every webhook we send includes:

```
POST <partner-url>
Content-Type: application/json
X-ViaOne-Event: order.delivered
X-ViaOne-Signature: sha256=<hex>
X-ViaOne-Timestamp: 1716243600
X-ViaOne-Idempotency-Key: <uuid>

{ ...payload... }
```

The signature is `HMAC-SHA256(payload, endpoint.secret)`. Partners verify by recomputing and comparing.

Webhook admin surface

[FIGURE 19: Webhook deliveries dashboard — Brain Hub > Webhooks]

Admin sees a table of recent deliveries with filters:

- By customer/tenant
- By state (failed_retrying, dead, all)
- By endpoint name
- By time window

Per-row actions:

- View payload (full JSON)
- View attempt history (per-try response codes + bodies)
- Manual replay (resets state to `pending`)
- Mark discarded (won't retry; also won't surface in alerts)

Dead-letter management

Webhooks in `dead` state surface in two places:

- The Webhook tab of the Live Transaction Console (red badge)
- Brain Hub > Webhooks filtered to `dead`

Best practice: review the dead list weekly. For each one:

1. Identify the failure mode (404? 500? timeout?)
2. Confirm with partner if their endpoint is back online
3. If yes — manual replay
4. If permanently broken — mark discarded with a reason

Adding a new webhook endpoint

A partner says "send me webhooks at https://partner.example.com/cb":

1. Brain Hub > Webhooks > Endpoints > New
2. Choose customer (the partner's customer record)
3. Endpoint name (e.g. `order_fulfillment_callbacks`)
4. URL: `https://partner.example.com/cb`
5. Events to subscribe (multi-select):

- `order.delivered`
- `order.failed`
- `order.partial`
- `order.reversed`
- `cashapp.success` / `.failed`
- `oxxo.paid`, `enefevo.paid`
- etc.

1. Click Generate secret → unique HMAC signing secret (shown ONCE; copy and send to partner)

2. Activate

The first webhook fires within seconds of any matching event in the customer's traffic.

Section last reviewed: 2026-05-06 — v1.0

---

12. Alerts management

NOC alerts are generated by the platform automatically and routed to the operator team via WhatsApp, the Command Center alert feed, and (optionally) email.

Alert sources

Source	What triggers an alert
Provider Health	Sustained red status (3+ failures in a row)
Smoke tests	Repeated soft-smoke failures
Catalog audit	Persistent drift
Pool monitor	DB connection pool > 90% utilization
Failed-order alarms	`>10` orders stuck in `pending` state for 5+ min
Webhook deaths	A burst of dead-letter webhook deliveries
Balance alerts	Customer balance below `minimum_alert_balance`
OFAC matches	A high-score match was found
Custom NOC rules	AI-driven anomaly detection in the NOC monitor

WhatsApp ops bot

Alerts are delivered to the WhatsApp ops bot (configured recipients in `services/viaone-alert-service.js`).

To add yourself to the alert list:

1. Edit `services/viaone-alert-service.js` `RECIPIENTS` array
2. Add your number in international format (e.g. `15551234567`)
3. Push and redeploy

To temporarily mute (not on-call):

- Use the bot's `/mute 4h` command — quietens that recipient for the window
- Or remove from the recipients array

"Recent failures" filter (post-Apr 27 fix)

[FIGURE 20: Alert feed with "Recent failures" filter — last 24h only]

The alert feed has a critical filter: **"Recent failures (last 24h)"**.

Before April 27 2026, the alert feed showed all failures from the start of time — including 6-month-old test data. A KK Squared test from April was bubbling up as "current" alert noise. The fix scoped failures to last 24h, but the filter is

configurable: set window to 1h / 6h / 24h / 7d depending on what you're triaging.

Acknowledging vs resolving

- **Acknowledge** = "I've seen this; stop pinging me about it." The alert remains in the table but moves to "acknowledged" section. Future identical alerts won't re-page.
- **Resolve** = "The underlying issue is fixed." Marks the alert with `resolved_at` and `resolved_by`. New alerts of the same kind will page again as expected.

Best practice: acknowledge during triage, resolve when the cause is gone.

Snooze / mute patterns

Operators sometimes need to silence noisy alerts during a known-bad window (e.g., partner is doing a stress test that they told us about). Two patterns:

1. **Per-alert mute** — click "Mute for 1h" on an alert; subsequent same-kind alerts suppress for 1h
2. **Global quiet window** — Brain Hub > Alerts > Quiet Window — schedule a window during which all alerts of certain types are suppressed (still logged, just not paged)

Always document the reason in the snooze comment, so the next operator knows why it's quiet.

Section last reviewed: 2026-05-06 — v1.0

PART III — TESTING COMMAND CENTER

The carrier-grade testing harness. A separate UI surface inside the Command Center, built specifically for protocol-level diagnosis and pre-launch verification.

---

13. ISO8583 message inspector

[FIGURE 21: ISO8583 message inspector — annotated with bitmap, MTI, F39, F126]

The inspector is the visual debugger for Telefónica TEMM ISO8583 traffic. It can:

- **Parse** a captured raw ISO8583 byte stream into named fields
- **Build** a request from form input and show the byte stream
- **Diff** two messages (request vs response, or two attempts) field-by-field
- **Color-code** the bitmap to show which fields are present
- **Replay** a captured message as a new live request

Where to find it

Brain Hub > Testing Command Center > ISO8583 Inspector

Reading a parsed message

The inspector decodes:

- **MTI** (Message Type Identifier): `0200` request / `0210` response / `0800` logon / `0810` logon response
- **Bitmap** (primary + secondary): which fields are present
- **F2** (PAN) — destination phone number for our use case
- **F3** (Processing code): 6 digits, e.g. `650000` for cert paquete, `650300` for prod
- **F4** (Amount): integer, no decimal, no currency
- **F11** (STAN): system trace audit number
- **F12** (Local time)
- **F37** (RRN): retrieval reference number, set in response
- **F39** (Response code): `00` = approved, others = decline. See Appendix B
- **F41**, **F42** — terminal ID, acquirer ID
- **F126** (Auxiliary): `MOVIA` for airtime, `MOVIH` (paquete numeric 16-25) for paquete

Building a test request

1. Pick a profile from the dropdown:

- "Cert paquete" — pre-filled with cert SKU + amount
- "Cert TAE" — pre-filled airtime
- "Prod paquete" — uses real package codes (use sparingly!)
- "Custom" — fill all fields by hand

1. Override fields as needed
2. Click "Build" → byte stream rendered with hex + ASCII view
3. Click "Send" → fires the message at the configured endpoint
4. Response panel populates with the decoded `0210` response

Diff mode

Pick two messages (e.g. one that worked, one that didn't) — the inspector overlays them with green/red field-by-field diff. Useful for "this used to work yesterday, what changed?" investigations.

Section last reviewed: 2026-05-06 — v1.0

---

14. Provider test station — Telefónica / LATCOM / MUWE

[FIGURE 22: Provider test station — pick provider, send test transaction]

The test station sits one level above the ISO8583 inspector — instead of building bytes, you pick a provider + product + destination, and the station handles the wire format for you.

Modes

Mode	What it does	Use when
ISO8583 (direct TCP)	Opens a TCP connection straight to TEMM, bypassing LATJAVA proxy	Diagnosing LATJAVA issues
HTTP via LATJAVA	The normal path: request goes to LATJAVA proxy, which translates to ISO8583	Normal cert testing
Simulation	No real network call; uses a mock server	Load testing without burning quota

Picking simulation vs cert

⚠️ **Cert quota is shared and tight.** Per `feedback_temm_cert_quota.md`: cert env has ~1-3 transactions/day quota, persistent F51 once exhausted.

Recommended hierarchy:

1. **Simulation** for regression / functional / load tests
2. **Soft smoke (cert)** for daily liveness check
3. **Targeted cert burst** only when validating a specific change (max 3 calls per session)
4. **Prod** only after Telefónica peer-allowlist + full pre-flight clear

A typical cert verification run

1. Open Testing CC > Provider Test Station
2. Provider = Telefónica MX
3. Mode = HTTP via LATJAVA

4. Catalog = pick a known-good SKU (e.g. `GLO01B_6M2D_P1`)
5. Phone = enter a known-good cert phone (e.g. `5537686648`)
6. Click Send
7. Watch the Response panel:

- F39 = 00 → ✅ working
- F39 = 51 → cert quota exhausted (Telefónica side, wait for reset)
- F39 = 83 → phone not provisioned in cert (use a different known-good number)
- F39 = 05 → mock environment (you're hitting simulation, not real cert)
- Others → see Appendix B

MUWE test station

Same UX, but for MUWE bill pay + SPEI:

- Pick the biller (e.g. CFE, Telmex)
- Enter the customer's account number / barcode
- Send → returns the MUWE transaction reference

LATCOM-format test station

For testing the `/api/tn/latcom` endpoint as if you were a distributor:

- Pre-built request templates (TFE topup, GLO paquete, LATCOM_PAQ legacy)
- Sends through the full ingress flow (login → JWT → /tn/latcom)
- Useful for verifying middleware changes end-to-end

[FIGURE 23: LATCOM-format test station with pre-built templates]

Section last reviewed: 2026-05-06 — v1.0

---

15. Multi-provider smoke test

[FIGURE 24: Multi-provider smoke test results table]

Runs a single test across all configured providers, in parallel. Useful as a launch-readiness check or post-deploy sanity test.

How to run

Brain Hub > Testing CC > Multi-Provider Smoke

Options:

- **Soft (default)** — send invalid input to each provider; check that they return an expected error code. Costs nothing.
- **Hard** — send a real $1 transaction to each. Costs $1 × N providers.

Click Run. Results stream in over ~30 seconds:

Provider	Result	Latency	Notes
CellPay	✅	131ms	error_code as expected
CodigoArix	❌	240ms	Authentication failed: ...
Altamira	✅	89ms	OK
PPN	⚠️	192ms	ok but catalog drift
Reloadly	❌	0ms	not configured
...			

Reading the results

- ✅ — provider responded with the expected behavior
- ⚠️ — responded but with a soft warning (drift, slow, etc.)
- ❌ — failed completely (auth, network, config)

A single ❌ on a non-critical provider isn't an outage — it's an alert. Multiple ❌ on critical providers (Telefónica, Altamira, PPN) indicates an upstream incident.

Schedule

Smoke runs automatically:

- **Hourly soft smoke** — at `:17` past the hour (catalog audit + soft tests)
- **15-min smoke** — at `3, 18, 33, 48` past each hour (high-frequency drift detection)

Manual hard smoke is a launch-day or "did my deploy break anything?" tool.

Section last reviewed: 2026-05-06 — v1.0

---

16. Live Transaction Console (cross-reference)

The Live Transaction Console is documented in detail in its **own dedicated manual** (separate PDF in this folder).

For Command Center integration purposes, what you need to know:

- **URL:** `app.via.one/console/`
- **Access:** admin JWT (same JWT as Command Center)
- **Three tabs:**
- **Inbound** — every API request hitting Via One in real time
- **Outbound (Providers)** — every call we make to providers (Telefónica, Altamira, PPN, etc.)
- **Webhooks** — every callback we deliver to partners
- **Auto-refresh:** every 2 seconds
- **Click any row** → expand to full request/response JSON (with sensitive fields redacted)
- **Filters:** time window (15 min / 1h / 6h / 24h), search box, status (any / OK / errors only)

The console is the **live observation layer** that pairs naturally with the Testing Command Center: run a test, watch the row land in the console, click to inspect the full payload.

For full details — interface tour, redaction policy, common workflows, troubleshooting — see *Via One — Live Transaction Console (User Manual).pdf*.

Section last reviewed: 2026-05-06 — v1.0

PART IV — CRM & CUSTOMER OPERATIONS

Day-to-day support operations. This is where you spend the most time once the platform is running.

---

17. CRM dashboard

[FIGURE 25: CRM dashboard — customer list with notes pane]

The CRM is the customer relationship surface. Less a "sales CRM" and more a "operational customer record" view.

What it shows

- Customer roster (filterable by tenant, status, last_login_at)
- Per-customer notes timeline (free-text, written by support agents)
- Per-customer recent activity (transactions, alerts, KYB events)
- Per-customer balance + reservation state
- Per-customer assigned support agent

Adding a note

1. Open a customer
2. Click "Add note"
3. Type. The note gets stamped with your name + timestamp.
4. Visible to all support agents and system_admins; not visible to the customer

Segmentation

Customers can be tagged with arbitrary tags (`high_priority` , `slow_payer` , `compliance_watch`). Use tags to filter the roster.

Manual broadcast

CRM > Broadcast lets you send a templated WhatsApp or email to a customer or a tag group. Examples:

- "Service maintenance window tonight 2-4am UTC"
- "New SKU available — see updated catalog"
- "Your IP whitelist expires in 7 days, please contact us"

Broadcasts are logged in `crm_broadcasts` for audit.

Section last reviewed: 2026-05-06 — v1.0

---

18. Transaction lookup tools

The most-used support function. A partner says "what happened to txn X?" → use one of these:

By transaction ID

[FIGURE 26: Transaction lookup by ID — full lifecycle view]

`/api/admin/transaction/`. Shows:

- The customer
- Inbound request body
- The route_mapping decision (which provider was chosen)
- The provider call (via `provider_transactions`)
- Provider's response
- Settlement status (reconciled? credited?)
- Any webhook deliveries to the partner about this txn
- Any related orders (cash payment, OXXO barcode, etc.)

By customer ID

`/api/admin/transactions?customerId=X` — last 50 transactions for that customer. Filter by status, date range.

By dist_transid (idempotency key)

`/api/admin/transaction-by-dist?dist_transid=X` — looks up the transaction by the partner's idempotency key. Useful when the partner says "I sent ID X but you say you don't have it."

By destination phone

`/api/admin/transactions?destination=5512345678` — every transaction we've routed to that phone, across all customers. Useful for fraud investigation.

Reading the full transaction lifecycle

A complete transaction has stages:

```
1. Inbound API request received (api_requests row)
2. Auth verified, customer + tenant resolved
3. Idempotency check — was this dist_transid seen before?
4. Balance reservation — atomic UPDATE on customer balance
5. Route decision — route_mappings layer picks provider
6. Provider call (provider_transactions row)
7. Response received from provider
8. Balance commit on success / release on failure
9. Webhook enqueue (webhook_deliveries row)
10. Webhook delivery to partner
```

The transaction lookup tool walks you through each stage with the relevant DB rows. Anywhere a stage is missing, you've found the failure point.

[FIGURE 27: Transaction lifecycle stages — annotated]

Section last reviewed: 2026-05-06 — v1.0

---

19. Reconciliation views

Telefónica, MUWE, and Codigo Arix all send daily reconciliation files (SFTP for Telefónica, REST API for MUWE/CodigoArix). These are matched against our `provider_transactions` records to detect:

- Transactions we think succeeded but the provider didn't process
- Transactions the provider processed but we don't have a record of
- Amount mismatches

Telefónica nightly reconciliation

Cron at `0 23 *` (11 PM Mexico City). Process:

1. Connect via SFTP to Telefónica's recibidos folder
2. Pull the previous day's CDR file
3. Parse each line; match against `provider_transactions` by STAN/RRN
4. Log mismatches in `reconciliation_log`
5. Flag transactions for manual review where amounts disagree

[FIGURE 28: Reconciliation results dashboard]

Codigo Arix daily reconciliation

Similar pattern via REST API. Runs every morning.

Stuck-order processor

Runs every 5 minutes. Looks for orders in `pending` or `processing` state for >30 min. For each:

- Re-query the provider for the latest status
- If still pending → leave alone (track for next round)
- If failed → mark as failed, release reservation, send webhook
- If succeeded but we missed the response → mark as success, commit reservation, send webhook

Manual reversal flow

For when a transaction succeeded at the provider but our system marked it failed (or vice versa):

1. Brain Hub > Customers > [customer] > Transactions
2. Find the txn
3. Click "Manual reverse"

4. Pick reason (system_error / fraud / duplicate / partner_request)
5. Confirm
6. The system:

- Reverses the balance debit
- Posts a journal entry
- Sends a `reversal` webhook to the partner

⚠️ **Use sparingly.** Manual reversals are audited. Any pattern of frequent reversals to the same customer is a red flag.

Section last reviewed: 2026-05-06 — v1.0

---

20. Wallet & balance management

Every customer has up to two wallets: USD (`current_balance`) and MXN (`balance_mxn`). The dual-currency system was deployed Apr 25 2026 for `professional` plan tenants.

Reserved vs available

- `current_balance` = total funds in the USD wallet
- `reserved_balance` = funds held for in-flight transactions (debit happens at request, commit at success)
- `available` = `current_balance − reserved_balance`

When a transaction starts: `reserved_balance += amount`. When it succeeds: `current_balance −= amount, reserved_balance −= amount`. When it fails: `reserved_balance −= amount` (no debit).

This way, a customer with $100 can never start two $80 transactions concurrently — the second one sees `available = $20` and is rejected.

Manual credit (top-up)

Section 7 covered the UI flow. Internally:

```
INSERT INTO balance_journal (
  customer_id, kind, currency, amount, reason, actor_user_id, created_at
) VALUES (
  $customer_id, 'manual_credit', 'USD', 100.00, $reason, $you, NOW()
);

UPDATE customers
SET current_balance = current_balance + 100.00
WHERE customer_id = $customer_id;
```

Both must happen atomically. Use the helper `creditCustomerBalance(customer_id, amount, currency, reason, actor)` rather than raw SQL.

Manual debit

Mirror of credit — `kind = 'manual_debit'`, `amount` subtracted. Used for billing corrections, fraud rollback, etc.

Stuck reservations

Sometimes a transaction's reservation never gets cleaned up (process crashed mid-flight, partner abandoned). The stuck-reservation cron (`cron/stuck-order-processor.js`) finds these and releases:

- Looks for `provider_transactions` in `PENDING` status > 30 min old
- Re-queries the provider
- Releases the reservation if confirmed failed
- Marks the order as failed
- Sends a failure webhook

If the cron isn't firing, customer balances slowly creep into "all reserved" — they can't start new transactions even though they have plenty of `current_balance` . **Always verify the cron is firing post-deploy** (Section 24.1).

Section last reviewed: 2026-05-06 — v1.0

PART V — NETWORK OPERATIONS CENTER (NOC)

The AI-powered incident diagnosis surface. NOC is the layer between raw alerts and human triage.

---

21. NOC overview

[FIGURE 29: NOC dashboard with active incidents + AI diagnosis pane]

NOC is at `app.via.one/app/noc`. It surfaces:

- The current "platform health" score (a synthesized number)
- Active incidents (≥ 1 alert + still open)
- AI diagnosis for each incident — what Claude thinks is happening + suggested action
- Tools to query underlying data (provider_transactions, cash_payment_orders, etc.)

Manual trigger

`POST /api/noc/monitor/run` — runs a full sweep right now. Useful when you've just done a deploy and want immediate confirmation things are healthy.

Toggle

`/api/noc/monitor/toggle?action=start` (default: on) `/api/noc/monitor/toggle?action=stop` (rare; e.g. during a planned maintenance window)

How AI diagnosis works

When an incident is opened (alerts converging), the NOC monitor:

1. Pulls the last 1h of relevant data (provider_transactions, alerts, deploys)
2. Feeds it to Claude with a structured prompt: "Diagnose this. What's the most likely cause? What action would you recommend?"
3. The response is rendered in the incident detail pane
4. An on-call human reviews, decides whether to act

Claude is **suggesting**, not deciding. Always sanity-check.

Section last reviewed: 2026-05-06 — v1.0

---

22. Reading NOC alerts

[FIGURE 30: NOC alert detail with AI summary, evidence, suggested action]

An NOC alert has these fields:

Field	What it is
`severity`	INFO / WARN / ERROR / CRITICAL
`source`	Which subsystem fired it (provider_health, pool_monitor, etc.)
`title`	Short headline
`body`	What happened, with relevant facts (txn IDs, timings, error codes)
`acknowledged_at`, `resolved_at`	Audit
`acknowledged_by`, `resolved_by`	Who handled it
`evidence`	JSONB with the underlying rows that triggered (txn IDs, log snippets)
`ai_diagnosis`	Claude's analysis, if generated
`ai_suggested_action`	Claude's recommendation

Severity levels

Level	Page someone?	Examples
INFO	No	Catalog audit drift, customer balance below threshold
WARN	No, log only	Smoke test failure (single occurrence)
ERROR	Quiet ping	Provider degraded, repeated webhook failures
CRITICAL	Loud page	Pool exhaustion, full provider outage, mass transaction failures

The Recent failures filter

(Already covered in Section 12, repeated here for the NOC use case.) When triaging, always filter to "Last 24h" first. Older alerts can be misleading because:

- They may be from before a fix was deployed
- They may be from test environments leaking up
- They may be already resolved but never marked

Section last reviewed: 2026-05-06 — v1.0

---

23. Common incident playbooks

A library of "this happened, do these steps" patterns. The most-used playbooks:

Provider going dark

Symptoms: red gauge for one provider, smoke test failing repeatedly, NOC CRITICAL alert.

Steps:

1. Confirm via Live Transaction Console: are recent provider calls timing out / 5xx-ing?
2. Check our side first: is our service's egress IP whitelisted at the provider? (Sometimes their firewall changes.)
3. If our side looks fine, contact the provider's tech ops via the documented channel

4. Check `route_mappings` failover for that provider — is there one configured? If yes, traffic should already be auto-failing-over (verify via the live console)
5. If no failover and the provider is critical → consider per-customer kill-switch (see Section 33)

DB pool exhaustion (the Apr 28 incident)

Symptoms: `/health/db` returning 503, "pool exhausted" errors in logs, transaction latency climbs, NOC alert.

Steps:

1. Open `/health/db` — what's the utilization? Waiting connection count?
2. Check `database-config.js` singleton — is it the only pool, or are there multiple `new Pool(` instances?
3. Are there any long-running transactions stuck? Check `pg_stat_activity` for queries > 30s
4. If a runaway query: kill it (`SELECT pg_cancel_backend(pid)`)
5. If genuine load: scale `PG_POOL_MAX` (currently 30 per pool; bumping to 60 is the test-deploy plan)
6. WhatsApp alerts should already be firing per the singleton + alert wiring shipped Apr 28

LATJAVA proxy unreachable

Symptoms: Telefónica calls all timing out, MUWE OK, etc. Specifically `/api/temm/api/tae` returning 502.

Steps:

1. Confirm via Live Transaction Console — are the failures concentrated in Telefónica calls?
2. SSH to LATJAVA (`10.10.2.5`) — is it up? Service running?
3. If LATJAVA is up but unresponsive: restart the gateway service
4. If LATJAVA host is dead: `REVERT-apr30.sh` is the disaster recovery script (per Section 33)
5. While LATJAVA is down, fail customers back to Altamira: per-customer SQL `UPDATE customers SET payments_iso8583_enabled = FALSE WHERE ...`

Cert env quota exhausted (the May 5 F51 pattern)

Symptoms: Telefónica cert calls returning F39=51 "Fondos insuficientes" persistently.

Steps:

1. Confirm: is this happening only on cert env (`PAYMENTS_CERT_MODE=true`) or on prod?
2. If cert: that's Telefónica's quota — they reset it on their schedule. Wait or contact Telefónica to ask for a refresh.
3. **Do not burst-test** on cert. Per memory `feedback_temm_cert_quota.md` : ~1-3 transactions/day, persistent F51 once exhausted.

Mass kill-switch

A bad deploy is causing widespread failures. Need to roll back distributors who were flipped to the new path.

```
-- Roll back ALL flipped distributors:
UPDATE customers SET payments_iso8583_enabled = FALSE
WHERE payments_iso8583_enabled = TRUE
RETURNING customer_id, short_name;
```

Per the kill-switch runbook (Section 33), this takes effect on the next inbound request after the UPDATE commits — measured at 756ms in the G3 dry-run.

Fastly DNS swing

Per DR-4: if we need to swing the public DNS away from Railway (because Railway's edge is sick), the steps are in DR_RUNBOOK §5.C. Note: requires Fastly API access, currently single-pointed on Richard. DR-4 is the open task to fix that.

[FIGURE 31: Incident playbook decision tree — visual]

Section last reviewed: 2026-05-06 — v1.0

PART VI — ADMIN-ONLY TOOLS

These are surfaces only `system_admin` can reach. They're powerful — also dangerous if misused.

24. System health

`/health` overview

Three endpoints, all admin-readable:

Endpoint	Returns
`/health`	Basic alive ping (`{ status: "ok" }`) — Railway uses this for healthcheck
`/health/db`	Pool stats + probe latency. 503 if pool > 90% utilization or waiting > 5
`/health/providers`	Per-provider audit + smoke status (60s in-memory cache)

[FIGURE 32: /health/db response in pretty JSON]

Pool stats (`/health/db`)

```
{
  "ok": true,
  "pool": {
    "total": 12,
    "idle": 8,
    "in_use": 4,
    "waiting": 0,
    "utilization_pct": 33
  },
  "probe_latency_ms": 4,
  "uptime_sec": 3625,
  "ts": "2026-05-06T08:30:00Z"
}
```

Read it like:

- `total` = total connections currently open
- `idle` = sitting unused
- `in_use` = handed out to a request
- `waiting` = requests in line for a connection (this is the danger metric — anything > 0 sustained = pool starvation)
- `utilization_pct` = `in_use / total`

Redis cache state

The `redis-cache.js` resilience layer reports state to the NOC. If Redis goes offline:

- Sessions fall back to in-memory (no cross-replica sharing)
- Rate-limit state degrades gracefully
- Webhook delivery queue continues (DB-backed, doesn't depend on Redis)
- NOC alert fires

DR sync lag (when DR-2 ships)

Once the open DR-2 task completes (currently in launch-week backlog per `DR_REVIEW.md`), `/health/db` will include `dr_sync_lag_seconds`. Until then, you have to check the DO Spaces dashboard manually.

Section last reviewed: 2026-05-06 — v1.0

---

25. Feature flags

Per-tenant booleans stored in `tenants.feature_flags` JSONB.

[FIGURE 33: Feature flags toggle UI]

Common flags

Flag	Effect
`payments_iso8583_enabled` (per-customer, not tenant)	Redirect Movistar traffic from Altamira to the new payments-iso8583 path
`dual_currency_wallet`	Enable USD + MXN wallets (vs single USD)
`webhook_v2`	Use the new durable webhook delivery queue vs legacy in-memory
`ofac_strict_mode`	Block any match ≥ 70 instead of just holding
`cert_mode_global`	Force all this tenant's traffic through cert (testing only)
`email_verification_required`	KYB applicants must verify email before submission
`dr_mode`	When true, transactional crons are skipped (DR replica safety — DR-1 task)

Flipping a flag

UI:

1. Brain Hub > Tenants > [tenant] > Feature Flags
2. Toggle the flag
3. Click Save
4. Cache invalidates within 5 min OR API restart

SQL (for emergencies):

```
UPDATE tenants
SET feature_flags = jsonb_set(feature_flags, '{webhook_v2}', 'true')
WHERE tenant_code = 'OPTIMUS';
```

Risky flag flips

- `payments_iso8583_enabled` — affects routing path mid-flight. Always verify no in-flight transactions for the customer first.
- `cert_mode_global` — sends real customer money to cert env (which won't actually deliver). Never flip on prod customers.
- `dr_mode` — disables crons. Only use when actively in DR.

Section last reviewed: 2026-05-06 — v1.0

---

26. Debug routes

`/api/debug/*` — admin-only routes for forensic inspection.

[FIGURE 34: Debug routes index page]

Common debug endpoints

Endpoint	What it does
`/api/debug/iso8583/parse`	Paste raw ISO8583 bytes → get parsed fields
`/api/debug/iso8583/build`	Build a request from form input → get the byte stream
`/api/debug/altamira/topup`	Direct Altamira call (bypassing customer auth + balance)
`/api/debug/codigoarix-test`	Direct Codigo Arix call
`/api/debug/temm/test-paquete`	Direct TEMM cert paquete (hardcoded cert profile)
`/api/debug/provider//health`	Direct provider health probe
`/api/debug/customer//state`	Full state dump for one customer (balances, reservations, recent txns)
`/api/debug/route/decide`	Given (operator, country, service, amount, tenant), what would route_mappings choose?

Production-safe vs cert-only

⚠️ Some debug routes hit real providers and cost real money. The Testing CC (Part III) is the safer surface for most operators. Use `/api/debug/*` only when:

- You need a one-off test that the Testing CC doesn't cover
- You're root-causing a routing decision (use `/api/debug/route/decide`)
- You need raw protocol-level access (use `/api/debug/iso8583/*`)

Section last reviewed: 2026-05-06 — v1.0

---

27. Manual operations (bulk + emergency)

Some operations require SQL because the UI doesn't expose them. Document and audit when you do these.

Bulk customer flag flip

Flipping `payments_iso8583_enabled = TRUE` on a known set of customers (e.g., the Monday flip-set):

```
UPDATE customers
SET payments_iso8583_enabled = TRUE
WHERE customer_id IN ('latcomdigital', 'LATCOM_PERU_001', 'HAZ_001')
  AND tenant_id = (SELECT id FROM tenants WHERE tenant_code = 'OPTIMUS')
RETURNING customer_id;
```

Always include the tenant_id check to prevent cross-tenant accidents.

Bulk balance credit

For an investor bonus or one-off promotion (rare):

```
WITH bulk AS (
  INSERT INTO balance_journal (customer_id, kind, currency, amount, reason, actor_user_id)
  SELECT customer_id, 'bulk_credit', 'USD', 100.00, 'May promotion', $YOU
  FROM customers
  WHERE tenant_id = (SELECT id FROM tenants WHERE tenant_code = 'X')
    AND is_active = TRUE
)
UPDATE customers SET current_balance = current_balance + 100.00
WHERE tenant_id = (SELECT id FROM tenants WHERE tenant_code = 'X')
  AND is_active = TRUE
RETURNING customer_id, current_balance;
```

Always bracket in a single transaction; never run the UPDATE without the journal INSERT.

Mass kill-switch

(Section 23 covered this; repeated here for the manual-ops chapter.)

```
UPDATE customers SET payments_iso8583_enabled = FALSE
WHERE payments_iso8583_enabled = TRUE
RETURNING customer_id, short_name;
```

Effective on the next inbound request — measured 756ms reroute latency in the G3 dry-run.

One-off DBA scripts

Sometimes a partner-specific request requires a one-off script (e.g., "create user Ramon Garcia", "fix Luis Merayo's balance"). Per memory `feedback_one_off_script_credentials.md`: keep credentials in the operator's source-of-truth. For these scripts:

- Write under `scripts/`
- Use literal credentials (not env vars; the operator workflow needs them inline)
- Commit + push so the audit trail survives
- Run from the operator's machine, not the prod app

Section last reviewed: 2026-05-06 — v1.0

PART VII — OBSERVABILITY

How you see what's happening, both live and historical.

28. Live Transaction Console reference

(See Section 16 — the Live Console has its own dedicated manual.)

For Command Center integration: the Live Console is bookmarked in the left navigation rail. Use the same admin JWT. It's the primary observability surface for real-time investigation.

Section last reviewed: 2026-05-06 — v1.0

29. Logs

Logs are structured (where possible) and routed to multiple sinks.

Where logs go

Sink	Contents
Railway stdout	All `console.log` / `console.error` in real time. Tail via `railway logs` or the Railway dashboard.
`access.log` (file)	Every HTTP request via morgan combined format — historical access log
`viaone-alert-service`	High-severity errors → WhatsApp ops bot (live alerts)

Tailing live logs

```
railway logs --service Latcom-fix --environment production
```

Filters available via `--filter` (Railway CLI):

```
railway logs --filter "ERROR"
railway logs --filter "[OFAC]"
railway logs --filter "latcomdigital"
```

Log throttling

To avoid Railway log rate limits, several subsystems throttle (e.g., Redis logs only on state change, not per attempt). If you see fewer messages than expected, that's why.

Per-service log filters

Service	Filter prefix
Latcom-fix	(default)
viaone-support	`[support]`
relier-ppn	`[ppn]`
viaone-command	`[command]`

Section last reviewed: 2026-05-06 — v1.0

---

30. Health endpoints

Already covered in Section 24. Quick reference:

- `/health` — basic alive
- `/health/db` — DB pool + probe
- `/health/providers` — per-provider gauge

External monitoring (Uptime Robot, etc.) hits `/health` every minute.

Section last reviewed: 2026-05-06 — v1.0

PART VIII — DEPLOYMENT & RELEASE OPS

How code gets from a developer's laptop to production. Critical for anyone who touches deploys.

31. The launch flow

Staging-first rule

Per memory `feedback_staging_first.md`: **NEVER deploy major changes to production directly.** The Chinese Wall break on Apr 1 2026 was caused by a Telefónica routing change pushed straight to prod with no staging soak. Don't repeat.

What counts as "major":

- Middleware that runs on every request (e.g., the Live Console request-logger)
- Routing-layer changes (route_mappings, optimusRouter, tenant-context)
- Auth changes (adminAuth, ipWhitelist)
- New providers or provider-call shape changes
- Schema migrations that change existing tables

What's safe-direct-to-prod (debatable):

- New isolated routes (don't touch existing flow)
- UI-only changes
- Documentation
- Feature-flag-gated rollouts where the flag defaults off

When in doubt: staging first.

Tier 1 / 2 / 3 / 4 / 5 hierarchy (from Kyle's launch checklist)

Tier	Meaning	Example
Tier 1	Hard blocker — do not flip first distributor without this	Code fixes for the launch path
Tier 2	Must be true Monday AM	Verification, runbooks, env vars
Tier 3	Tuesday launch prerequisites	Altán + Bait/Walmart go-live
Tier 4	Post-launch week 1	Cleanup, F-items, DR launch-week
Tier 5	Later (P2/P3)	Sidebar cleanup, EV V1

The full checklist for the May 2026 Telefónica launch is `MONDAY_LAUNCH_CHECKLIST.md` — keep it as the canonical doc for that launch.

Pre-flight checklist (generic)

Before any production deploy:

- [ ] Code reviewed (PR approved)

- [ ] Staging deploy succeeded
- [ ] Staging soak ≥ 15 min (longer for behavioral changes)
- [ ] Migration applied to staging — verified via `\d not just pgmigrations row`
- [ ] Smoke tests green on staging
- [ ] Operational runbook updated for any new failure modes
- [ ] On-call notified of deploy window
- [ ] Rollback plan documented (commit SHA to revert to, kill-switch SQL ready)

Cutover windows

Major deploys happen at low-traffic windows:

- Telefónica volume peaks 9am–9pm CDMX
- Best deploy window: 3am–7am CDMX (low traffic, on-call awake)
- For the May 2026 launch: 00:00 Thursday May 7 (zero traffic)

Rollback decision tree

After a deploy, watch:

- Error rate per minute
- p95 latency
- NOC alert volume

Decision tree:

- Error rate > 5% over 5 min → **roll back deploy**
- Single-distributor errors > 20% → **flip back that distributor's flag** (Section 33)
- LATJAVA unreachable → **mass kill-switch**
- NOC alert volume × 3 normal → **investigate before next action**

Rollback steps:

- Revert via `git revert` + push
- OR redeploy previous commit via `railway up --detach` from the worktree of the prior commit
- Once revert lands, `kill-switch` any in-flight customer flags

Section last reviewed: 2026-05-06 — v1.0

---

32. Custom domain management

When a partner brings their own domain (e.g., `buprolat.latcom.co`), here's the full procedure.

The CNAME-not-IP rule

Railway does NOT issue static inbound IPs. The `Static Outbound IPs` setting (e.g. `162.220.234.15`) is **outbound-only** — Railway's UI explicitly says "cannot be used for inbound traffic."

Partners must **CNAME** to the Railway-issued target, not A-record to any IP. If a partner insists on an A record (some old DNS hosts), the only options are:

- Railway Pro Static Ingress IP (paid add-on)
- A proxy in front (CDN, Cloudflare, etc.) that gives them a static A — they CNAME the proxy, proxy CNAMEs us

Adding a partner subdomain step-by-step

[FIGURE 35: Custom domain setup wizard — Brain Hub > Domains > Add]

1. **Get the partner's hostname** (e.g. `buprolat.latcom.co`).
2. **Add as Railway custom domain** via GraphQL (CLI has the TXT-prefix bug per `feedback_railway_cli_txt_bug.md`):

```
TOKEN=$(...)
curl -s -X POST https://backboard.railway.com/graphql/v2 \
  -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
  -d '{
    "query": "mutation cdc($input: CustomDomainCreateInput!) {
      customDomainCreate(input: $input) {
        id domain
        status {
          certificateStatus
          dnsRecords { recordType requiredValue currentValue status }
        }
      }
    }",
    "variables": {
      "input": {
        "domain": "buprolat.latcom.co",
        "environmentId": "<prod env id>",
        "projectId": "<project id>",
        "serviceId": "<latcom-fix service id>"
      }
    }
  }'
```

1. **Railway returns:**
   - A CNAME target (e.g. `61vobyc4.up.railway.app`)
   - A TXT verification record host + value
1. **Partner adds two DNS records** at their DNS provider:

`` CNAME buprolat → 61vobyc4.up.railway.app TXT _railway-verify.buprolat → railway-verify=29f884af964b... ``

1. **Wait for DNS propagation** (~5 min for most providers; up to 1 hour for slow ones).
1. **Railway issues SSL cert** automatically once it sees both records. Cert state: `VALIDATING_OWNERSHIP` → `ISSUING` → `VALID`.
1. **UPDATE the tenant row:**

``sql UPDATE tenants SET custom_domain = 'buprolat.latcom.co' WHERE tenant_code = 'OPTIMUS';``

1. **Test with curl:**

``bash curl -s -o /dev/null -w "HTTP=%{http_code} ssl_ok=%{ssl_verify_result}\n" \ https://buprolat.latcom.co/api/dislogin # Expected: HTTP=405 (method not allowed for GET on a POST-only route) or HTTP=200, ssl_ok=0 (success)``

1. **Document in the tenant row's `settings`** what hostname maps where, for future operators.

TXT verification gotcha

Per memory `feedback_railway_cli_txt_bug.md`: the `railway domain` CLI command **prints the TXT verify with a double `railway-verify=` prefix.** Do NOT add another prefix to the TXT value. The GraphQL `verificationToken` field gives the correctly-formed value — paste it verbatim.

DNS propagation expectations

- **Same-DNS-provider as their main records** (most common): 1–5 min
- **Different DNS provider for the subdomain:** 5–30 min
- **Slow providers (some legacy):** up to 1 hour

If after 1 hour the cert still hasn't issued and DNS is verifiably propagated everywhere external (`dig` from multiple resolvers), Railway's verifier may be flapping. Open a ticket with Railway support — sometimes they need to manually nudge.

Multiple domains per tenant

Currently: one row in `tenants` can hold one `custom_domain`. If a tenant needs 3 domains all pointing to the same backend:

- **Option A (v1):** Add 3 separate tenant rows (different tenant_code each, but same downstream config). Confusing but works.
- **Option B (planned):** Extend to a `tenant_aliases` table. Not yet implemented.

For most cases, one domain per tenant is enough.

Section last reviewed: 2026-05-06 — v1.0

---

33. Kill-switch runbook

The fastest way to revert a deploy that's misbehaving in the wild.

Per-customer kill-switch

```
-- Roll back ONE distributor:
UPDATE customers
SET payments_iso8583_enabled = FALSE
WHERE customer_id = 'latcomdigital'
RETURNING customer_id, short_name;
```

Effective on the very next inbound request after commit (auth-jwt re-fetches customer per request — no JWT/Redis cache to flush).

Mass kill-switch

```
-- Roll back ALL flipped distributors:
UPDATE customers
SET payments_iso8583_enabled = FALSE
WHERE payments_iso8583_enabled = TRUE
RETURNING customer_id, short_name;
```

Cache invalidation timing

The G3 kill-switch dry-run (April 2026) measured 756ms from the SQL UPDATE commit to the next request being routed via the new path. So:

- T+0: SQL commit
- T+~750ms: next inbound request lands on the OLD path

If you need faster: API restart. The customer cache is in-process; restart clears it instantly. But restart has its own cost (in-flight transactions during restart get aborted).

Kill-switch decision matrix

Scenario	Action
Single distributor showing high errors	Per-customer kill-switch on that distributor
Multiple distributors in same tenant failing	Mass kill-switch for that tenant (filter by tenant_id in the SQL)
All customers across tenants failing	Mass kill-switch globally
LATJAVA proxy itself dead	Mass kill-switch + restart LATJAVA
Bad code in production	Revert the deploy AND mass kill-switch (defense in depth)

Rollback decision tree (repeat from Section 31)

1. Error rate > 5% over 5 min → roll back deploy
2. Single distributor errors > 20% → flip back that distributor only
3. LATJAVA unreachable → mass kill-switch
4. Provider auth/firewall issues → contact provider, no code rollback needed

Section last reviewed: 2026-05-06 — v1.0

PART IX — INTEGRATIONS REFERENCE

A directory of every active integration, in case the reader needs to know "where does X live."

34. Provider directory

Mexico topup / paquete

Provider	Purpose	Endpoint	Auth	Notes
Telefónica MX (TEMM)	ISO8583 high-volume payments	`10.225.236.72:7903` (prod), `10.225.244.79:7903` (cert) via LATJAVA proxy	Client cert (cert env), peer-allowlist (prod)	The May 2026 launch target
Altamira	SOAP recargas + paquetes (Movistar IVA)	proxy at `10.225.236.116:9303`	App ID + user code	IVA-adjusted: send `ceil(amount/1.16)`
Codigo Arix	MX paquetes	`https://api.codigoarix.com/arix/api`	ID cliente + usuario + clave	Telcel/AT&T routes here
MUWE	Bill pay + SPEI + topup	`https://test.sipelatam.mx` (sandbox)	MD5 signature	OXXO/Enefevo cash via webhook
Taecel	Utility bills	`${TAECEL_BASE_URL}`	API key	
Altán	MVNO multi-brand	`${ALTAN_BASE_URL}`	API key auto-loaded from `altan_devapps` table	BE 347 + others

International topup

Provider	Purpose	Endpoint	Auth
PPN / ValueTopup	100+ countries	`${PPN_BASE_URL}`	HTTP Basic + IP whitelist
CellPay	US topups + gift cards	`${CELLPAY_BASE_URL}`	API key
Reloadly	50+ countries (fallback)	`https://api.reloadly.com`	API key
NIHN	Page Plus + MobileX (US)	via PPN proxy	API key
Servipagos VZ	Venezuelan bill pay	`${SERVIPAGOS_BASE_URL}`	Merchant ID + API token

Card / hosted checkout

Provider	Purpose	Endpoint	Auth
Stripe	Cards + ACH	`https://api.stripe.com`	Secret key
Pockyt	Alipay/WeChat/Apple/Google Pay/Cash App	`https://mapi.yuansferdev.com` (sandbox)	MerchantNo + StoreNo + API token + MD5 sig

Blockchain / wallets

Provider	Purpose	Endpoint	Auth
Privy	KMS for wallet keys	`https://api.privy.io`	App ID + app secret
Movement	Primary RISE chain	`https://mainnet.movementnetwork.xyz/v1`	None (public RPC)
Aptos	Fallback chain	`https://fullnode.mainnet.aptoslabs.com/v1`	None
Tron	USDT (TRC-20)	`https://api.tronstack.com`	None
Polygon	Legacy USDC	Alchemy / Quicknode	API key

Messaging

Provider	Purpose
WhatsApp Cloud API (Meta)	Primary partner-facing channel
Twilio	SMS, voice, legacy WA
uContact	Alternate SMS
Telegram	Secondary channel via grammY
Vonage	US carrier number lookup
Tlaloc	Internal carrier detection

Compliance / data / AI

Provider	Purpose
treasury.gov OFAC	Sanctions list source (daily 04:30 UTC sync)
ExchangeRate-API	USD/MXN FX (daily 9am refresh)
Anthropic Claude	NLP (intent classification, NOC analysis)
SendGrid	Transactional email

Storage

Provider	Purpose
AWS S3	KYC docs, reconciliation reports
DigitalOcean Spaces	Daily encrypted PG + LATJAVA backups

Section last reviewed: 2026-05-06 — v1.0

35. Tenant directory

Tenant	Description	Plan	Distinguishing details
OPTIMUS	Latcom Horizons II — Telefónica MX distribution	professional	Largest volume, May 2026 launch, custom domain `buprolat.latcom.co`
HAZ_GROUP	HAZ Communications — multi-LATAM	professional	Multi-country (HAZ_GROUP customers have allowed_countries on each row)
RISE FINANCE	RISE Holdings — DeFi remittance	professional	Movement chain primary, Aptos fallback, non-custodial
VIAONE	Direct-to-consumer Via One products	professional	El Vecino, EV bot, Via One web app
VIAONE_MASTER	Cross-tenant admin tenant	n/a	system_admin role lives here

Section last reviewed: 2026-05-06 — v1.0

36. Customer directory (high-level)

Active distributors per tenant with their integration shape:

OPTIMUS

- `latcomdigital` — LatCom Test Barcelona (LATCOM-format `/api/dislogin` + `/api/tn/latcom`)
- `LATCOM_PERU_001` — Latcom Peru
- `LATCONECTA_001` — Latconecta Digital (IP-whitelisted)

- (12 others — see `SELECT customer_id, company_name FROM customers WHERE tenant_id = (SELECT id FROM tenants WHERE tenant_code = 'OPTIMUS')` for live list)

HAZ_GROUP

- `HAZ_001` — HAZ Group main distributor

RISE FINANCE

- `RISE_HOLDINGS_001` — RISE Holdings

VIAONE

- `013141` — El Vecino
- (others)

VIAONE_MASTER

- system_admin user records

Section last reviewed: 2026-05-06 — v1.0

PART X — APPENDICES

Appendix A — Endpoint cheat sheet

Public (no auth)

```
GET  /health
GET  /health/db
GET  /health/providers
```

Distributor API

```
POST /api/distributor/login          (or /api/login)        JSON: {customer_id, secret_key}
POST /api/dislogin                   (LATCOM-compat)        JSON: {username, password,
dist_api?, user_uid?}
POST /api/distributor/product_purchase                      JWT, Idempotency-Key
POST /api/distributor/tn/fast        (or /api/tn)            JWT
POST /api/distributor/tn/latcom      (or /api/tn/latcom)     JWT
GET  /api/distributor/get_balance                            JWT
GET  /api/distributor/product_list                            JWT
```

Customer API (Via One v1)

```
POST /api/v1/topup                   JWT
POST /api/v1/bill-payment            JWT
POST /api/v1/spei                    JWT
POST /api/v1/catalog/topup           JWT
GET  /api/v1/balance                 JWT
```

Console (admin)

```
GET /api/console/summary
GET /api/console/inbound
GET /api/console/outbound
GET /api/console/webhooks
GET /api/console/{type}/:id
```

Admin

```
GET  /api/admin/transaction/:id
GET  /api/admin/transactions?customerId=
GET  /api/admin/all-transactions
POST /api/admin/test-downtime-alert
POST /api/admin/add-credit
GET  /admin/ofac/status
POST /admin/ofac/sync?list=sdn
POST /admin/ofac/screen
GET  /admin/ofac/screening-log
GET  /api/brain-hub/overview
GET  /api/brain-hub/kyb/applications
```

Debug

```
POST /api/debug/iso8583/parse
POST /api/debug/iso8583/build
POST /api/debug/altamira/topup
POST /api/debug/codigoarix-test
POST /api/debug/temm/test-paquete
GET  /api/debug/customer/:id/state
POST /api/debug/route/decide
```

Webhooks (we receive)

```
POST /webhook/meta/whatsapp
POST /webhook/meta/messenger
POST /webhook/telegram
POST /webhook/imessage
POST /webhook/rcs
POST /webhook/muwe/oxxo
POST /webhook/muwe/enefevo
POST /webhook/payment/stripe/:methodKey
POST /webhook/pockyt
POST /webhook/twilio/sms
POST /webhook/twilio/status
POST /webhook/uptime
```

Appendix B — F39 response code dictionary

(ISO8583 response codes as Telefónica TEMM uses them.)

F39	Meaning	Action
00	Approved	Success
01	Refer to issuer	Investigate; partner-side issue
05	Do not honor	Generic decline; check destination eligibility
12	Invalid transaction	Malformed request; check fields
13	Invalid amount	Amount out of range
14	Invalid card / account	Wrong destination
30	Format error	Malformed wire format
41	Lost card	n/a for our use case
43	Stolen card	n/a
51	**Insufficient funds**	**Cert env: quota exhausted; Prod: customer balance issue**
54	Expired	n/a
55	Invalid PIN	Auth failure
57	Transaction not permitted to issuer	Carrier doesn't accept this op
58	Transaction not permitted to terminal	Routing or terminal config issue
61	Exceeds withdrawal amount limit	Amount too high
62	Restricted card	Carrier-side block
63	Security violation	Security issue (rare)
65	Exceeds withdrawal frequency	Velocity limit hit
68	Response received too late	Timeout — provider may have processed; reconcile
75	Invalid PIN entries	Auth
76	Invalid product	SKU not allowed
82	Time-out at issuer	Provider timeout
83	**Cannot complete, violation of law**	TEMM cert: phone not provisioned in cert env
87	Purchase only — no cash advance	Carrier-side restriction
91	Issuer or switch inoperative	Provider down
92	Financial institution not found	Routing error
94	Duplicate transmission	Idempotency hit
96	System malfunction	Provider error

Appendix C — SKU naming convention

Knowing the prefix tells you a lot:

Prefix	Meaning	Example
`TFE_`	Telefónica MX topup (TAE-style, open-range MXN)	`TFE_150_MXN`, `TFE_MXN_20_TO_2000`
`GLO01T_`	Telefónica MX topup (open-range, P1=MXN-priced, D1=USD-priced)	`GLO01T_RAMXN_P1`, `GLO01T_RAUSD_D1`
`GLO01B_`	Telefónica MX paquete (P1=MXN-priced, D1=USD-priced)	`GLO01B_6M2D_P1`, `GLO01B_6M2D_D1`
`LATCOM_PAQ`	Legacy LATCOM Barcelona paquete (hardcoded mapping)	`LATCOM_PAQ16`, `LATCOM_PAQ17`, `LATCOM_PAQ18`
`GCA09T_`	Telcel topup via Codigo Arix	`GCA09T_080MXN_C1`
`GCA09B_`	Movistar paquete via Codigo Arix	`GCA09B_100MXN_C1`
`RMP`	Movistar paquete code (Codigo Arix wire)	`RMP100`, `RMP150`, `RMP200`
`PQRI`	Telefónica MX paquete wire SKU (TEMM operator_sku_id)	`PQRI6M2DP`, `PQRI1G4DP`, `PQRIIPM`
`TEMXN_`	Telefónica MX (newer naming)	`TEMXN_RECARGA_30_TEMM`, `TEMXN_PQRI6M2D_2_DAYS`
`TAETELCEL`	Tiempo Aire Electrónico Telcel (open-range)	`TAETELCEL`
`INT`, `PA`, `PAA`	Telcel data plans	`INT`, `PA`
`MUWE_`	MUWE biller	`MUWE_4052305028`
`SPVZ-`	Servipagos VZ biller	`SPVZ-MOV-POS`

The carrier is encoded in the prefix; the channel is encoded in the suffix (`_P1` = MXN-priced point-1, `_D1` = USD-priced).

Appendix D — Tenant-context middleware behavior

The full algorithm is in `middleware/tenant-context.js`. Key behaviors:

1. **Hostname → tenant resolution:**
   - Extract first label of hostname (`optimus` from `optimus.via.one`)
   - Query: `SELECT * FROM tenants WHERE subdomain = $1 AND status = 'active'`
   - Fallback: `WHERE custom_domain = $1`
1. **5-minute LRU cache** (max 200 entries) — eliminates DB hit on every request.
1. **Negative caching** — unknown hosts cached as `tenant: null` for the same TTL, preventing repeated failed lookups.
1. **Circuit breaker** — opens after 3 consecutive DB failures, 30-second cooldown, falls back to negative cache during outage.

1. **PostgreSQL session variable** — once a tenant is resolved, the middleware sets `app.current_tenant = $tenant_id` on the connection. RLS policies on transactions / customers / route_mappings filter by this variable.

Diagnostic queries

If tenant resolution seems off, try:

```
-- What does the resolver see for this hostname?
SELECT id, tenant_code, subdomain, custom_domain, status
FROM tenants
WHERE subdomain = 'optimus' OR custom_domain = 'optimus.via.one';

-- Is RLS active for this tenant?
SET app.current_tenant = '<uuid-from-tenants.id>';
SELECT COUNT(*) FROM customers; -- should match expected for that tenant
RESET app.current_tenant;
```

Appendix E — Critical environment variables

Var	Default	Purpose
`JWT_SECRET`	(required)	HS256 secret for admin/distributor JWTs
`DATABASE_URL`	(required)	Postgres connection string (Railway internal)
`DATABASE_PUBLIC_URL`	(auto)	Postgres public URL (for psql from outside)
`REDIS_URL`	(required for v2 webhooks + circuit breaker hydration)	Redis connection
`NODE_ENV`	`production` (must!)	When `staging`, providers default to mock mode
`PAYMENTS_CERT_MODE`	`false`	When `true`, paquete attempts get rewritten to cert pair (code=21, amount=150)
`PAYMENTS_PROXY_URL`	`https://66.231.242.91`	LATJAVA proxy URL
`PAYMENTS_GATEWAY_API_KEY`	(required for prod)	Gateway auth key
`PAYMENTS_TIMEOUT_MS`	45000	Telefónica timeout — consider 30000 in prod for faster reversal
`PAYMENTS_IVA_ADJUST`	`false`	Enable IVA targeting
`PAYMENTS_CERT_PAQUETE_CODE`	`21`	Cert override package code
`PAYMENTS_CERT_PAQUETE_AMOUNT`	`150`	Cert override amount
`TELEFONICA_GATEWAY_URL`	`https://66.231.242.91`	Same as PAYMENTS_PROXY_URL
`PG_POOL_MAX`	30	Max connections per pool (each service has its own pool)
`FORCE_REAL_PROVIDERS`	unset	Staging-only override to call real providers
`DR_MODE`	unset	When set + read by code (DR-1 task), skips transactional crons on the DR replica
`RAILWAY_TOKEN`	(set in shell)	Token for Railway CLI in non-interactive mode
`OFAC_API_KEY`	unused	Legacy — we self-host now
`PASSWORD_RESET_EMAILS_ENABLED`	`false`	Set `true` once SMTP env vars are configured

Appendix F — Database table directory

By domain. One-line purpose each.

Identity & tenancy

- `tenants` — top-level partition
- `customers` — distributors / API consumers within a tenant
- `users` — UI/dashboard logins (separate from customers)
- `api_keys` — per-customer auth tokens

Transactions & ledger

- `transactions` — primary transaction ledger (customer-facing view)
- `provider_transactions` — every provider call we've made (with full payloads)
- `operator_balance` — per-provider held balance
- `balance_journal` — audit trail for manual credits/debits
- `idempotency_keys` — dedup for retries

Wallets & blockchain

- `wallet_addresses` — per-user, per-chain crypto addresses
- `wallet_transfers` — chain transactions
- `ev_accounts` — El Vecino consumer wallet records

Compliance

- `ofac_entries`, `ofac_alt_names`, `ofac_addresses`, `ofac_ids` — sanctioned name corpus
- `ofac_screening_log` — audit of every screening event
- `ofac_sync_metadata` — sync status
- `kyc_documents` — uploaded KYC docs (S3 key references)
- `kyb_applications` — partner intake forms
- `audit_log` — generic admin action audit

Catalog

- `latcom_products` — canonical SKU catalog
- `operator_products` — per-provider SKU mapping
- `route_mappings` — operator + tenant + service → provider routing
- `vendors` — provider credential registry
- `altan_devapps` — Altán BE-specific credentials

Webhooks

- `webhook_endpoints` — partner-configured callback URLs
- `webhook_deliveries` — durable delivery queue with state + retry history

Sessions & conversations

- `sessions` — UI sessions
- `conversations` — bot conversation threads
- `siri_sessions` — Siri integration state
- `auth_logs` — login attempts (success + failure)

Bots / messaging

- `whatsapp_messages` — inbound + outbound WA log
- `rcs_users`, `channel_identities` — multi-channel identity mapping

Console (this manual)

- `api_requests` — inbound API hit log (powers the Live Transaction Console)

Other

- `dr_sync_log` — DR replica sync state
- `operators` — telecom carrier registry
- `feature_flags` (table; also lives in tenants.feature_flags JSONB) — global feature gates

---

Appendix G — Common SQL queries

"Last 24h transactions for tenant X"

```
SELECT t.created_at, c.customer_id, t.product_type, t.customer_amount, t.status
FROM transactions t
JOIN customers c ON c.id = t.customer_id
WHERE c.tenant_id = (SELECT id FROM tenants WHERE tenant_code = 'OPTIMUS')
  AND t.created_at >= NOW() - INTERVAL '24 hours'
ORDER BY t.created_at DESC LIMIT 200;
```

"Customers with payments_iso8583 flipped"

```
SELECT customer_id, company_name, current_balance, balance_mxn
FROM customers
WHERE payments_iso8583_enabled = TRUE;
```

"Recent OFAC matches"

```
SELECT screened_at, name_input, country_input, context, top_match_score, blocked
FROM ofac_screening_log
WHERE screened_at >= NOW() - INTERVAL '7 days'
  AND top_match_score >= 70
ORDER BY screened_at DESC;
```

"Webhook deaths in last week"

```
SELECT d.created_at, e.customer_id, e.url, d.event, d.attempts, d.last_error
FROM webhook_deliveries d
JOIN webhook_endpoints e ON e.id = d.endpoint_id
WHERE d.state = 'dead'
  AND d.dead_at >= NOW() - INTERVAL '7 days'
ORDER BY d.dead_at DESC;
```

"Provider success rate last 1h"

```
SELECT provider,
       COUNT(*) AS total,
       COUNT(*) FILTER (WHERE status = 'SUCCESS') AS ok,
       COUNT(*) FILTER (WHERE status IN ('FAILED','TIMEOUT')) AS err,
       ROUND(100.0 * COUNT(*) FILTER (WHERE status = 'SUCCESS') / COUNT(*), 1) AS success_pct,
       AVG(latency_ms)::int AS avg_latency_ms
FROM provider_transactions
WHERE created_at >= NOW() - INTERVAL '1 hour'
GROUP BY provider
ORDER BY total DESC;
```

"Customer balance below alert threshold"

```
SELECT customer_id, company_name, current_balance, minimum_alert_balance
FROM customers
WHERE current_balance < minimum_alert_balance
  AND minimum_alert_balance > 0
  AND is_active = TRUE
ORDER BY current_balance ASC;
```

Appendix H — Document history

Version	Date	Notes
1.0	2026-05-06	Initial release — 10 parts + 8 appendices, English. Spanish edition published as separate PDF.

End of manual.

For the Live Transaction Console, see *Via One — Live Transaction Console (User Manual).pdf*.

For partner-specific integration docs, see *Existing Partner Manuals/* in this folder.

For Telefónica wire-protocol specs, see *Reference Specs/* in this folder.

Updates and corrections: please contribute via pull request to the markdown source file — the PDF regenerates from there.