Via One

Centro de Comando — Manual Maestro

Versión 1.0 · Mayo 2026
Edición en español completa
Audiencia: operadores internos, ingenieros de guardia, administradores del sistema, soporte cara a socios, administradores de distribuidores, oficiales de cumplimiento

Via One — Centro de Comando

Manual Maestro

Versión: 1.0

Fecha del documento: 2026-05-06

Audiencia: Operadores internos de Via One, ingenieros de guardia, administradores del sistema, soporte cara a socios, administradores de distribuidores, oficiales de cumplimiento

Idiomas disponibles: Inglés (edición separada), Español (este documento)

Cómo leer este documento

Este manual está estructurado para dos modos de lectura:

- **Lectura lineal** para nuevos operadores — léalo de principio a fin, toma ~3 horas absorberlo. Hojee primero las Partes I–III, luego entre en las Partes IV–IX a medida que encuentre las superficies.
- **Búsqueda de referencia** para operación diaria — use la Tabla de Contenidos o el Apéndice A (lista de endpoints) para saltar directamente a lo que necesita.

Cada sección termina con un pequeño pie de página que indica cuándo se revisó por última vez (ej. *Última revisión: 2026-05-06 — v1.0*). Cuando la superficie subyacente cambie, la fecha y versión de la sección relevante se incrementan independientemente.

Los marcadores `[FIGURA X: descripción]` indican dónde deben insertarse capturas de pantalla. Captúrelas de la UI en vivo y reemplace el marcador con la imagen.

Tabla de Contenidos

Parte I — Orientación

1. ¿Qué es el Centro de Comando?
2. Quién lo usa, y cómo
3. Cómo acceder
4. Glosario inicial

Parte II — Brain Hub

1. Pantalla general del Brain Hub
2. Gestión de tenants
3. Gestión de clientes (distribuidores) dentro de un tenant
4. Cola de onboarding KYB
5. Cumplimiento y OFAC
6. Salud de proveedores y pruebas smoke
7. Cola de entrega de webhooks

8. Gestión de alertas

Parte III — Centro de Comando de Pruebas

1. Inspector de mensajes ISO8583
2. Estación de pruebas de proveedores
3. Prueba smoke multi-proveedor
4. Consola de Transacciones en Vivo (referencia cruzada)

Parte IV — CRM y Operaciones de Cliente

1. Panel de CRM
2. Herramientas de búsqueda de transacciones
3. Vistas de conciliación
4. Gestión de monederos y saldos

Parte V — Centro de Operaciones de Red (NOC)

1. Resumen del NOC
2. Lectura de alertas del NOC
3. Manuales de incidentes comunes

Parte VI — Herramientas Solo Para Administradores

1. Salud del sistema
2. Banderas de funciones
3. Rutas de depuración
4. Operaciones manuales (masivas + emergencia)

Parte VII — Observabilidad

1. Referencia a la Consola de Transacciones en Vivo
2. Logs
3. Endpoints de salud

Parte VIII — Operaciones de Despliegue y Lanzamiento

1. El flujo de lanzamiento
2. Gestión de dominios personalizados
3. Manual del kill-switch

Parte IX — Referencia de Integraciones

1. Directorio de proveedores
2. Directorio de tenants
3. Directorio de clientes (alto nivel)

Parte X — Apéndices A. Hoja de referencia de endpoints B. Diccionario de códigos de respuesta F39 C. Convención de nombres de SKU D. Comportamiento del middleware tenant-context E. Variables de entorno críticas F. Directorio de tablas de base de datos G. Consultas SQL comunes H. Historial del documento

---

PARTE I — ORIENTACIÓN

1. ¿Qué es el Centro de Comando?

El Centro de Comando de Via One es el **único plano de control basado en web** para toda la plataforma Via One. Desde un solo panel, un operador puede:

- Ver tráfico en vivo a través de cada tenant, cada proveedor, cada webhook
- Gestionar cuentas de distribuidores, saldos, listas blancas de IP, y banderas de funciones
- Ejecutar pruebas de grado carrier contra TEMM cert, Altamira, MUWE, y otros proveedores
- Revisar aplicaciones de onboarding KYB y aprobarlas
- Filtrar nombres contra listas de sanciones OFAC en tiempo real
- Diagnosticar incidentes usando análisis NOC potenciado por IA
- Activar interruptores de apagado (kill-switches) y rollbacks
- Observar el flujo de mensajes ISO8583 / SOAP / REST subyacentes, con cuerpos completos

Es el centro neurálgico. Cada otra superficie de Via One — las APIs de socios, los bots, los monederos de escrow, el motor OFAC, los crons de conciliación — alimenta esto o se gestiona desde aquí.

[FIGURA 1: Pantalla principal del Centro de Comando — vista general del Brain Hub con mosaicos de tenants + tarjetas KPI]

Los cinco tenants

Via One sirve a **cinco tenants lógicos**, cada uno aislado de los demás por seguridad de nivel de fila y middleware tenant-context:

Código	Nombre comercial	Propósito
`OPTIMUS`	Latcom Horizons II (Optimus)	Distribución Telefónica MX — el tenant de mayor volumen, objetivo ~200K txn/mes
`HAZ_GROUP`	HAZ Group	Socio de distribución multi-país latinoamericano (HAZ Communications)
`RISE FINANCE`	RISE Holdings	Producto DeFi de remesas + monedero (cadena Movement primaria, Aptos respaldo)
`VIAONE`	Via One core	Productos Via One directos al consumidor (El Vecino, etc.)
`VIAONE_MASTER`	Via One Master	Tenant de administración cruzada; el rol system_admin vive aquí

La vista de un operador dado se filtra automáticamente por los tenants a los que tiene acceso. Un `system_admin` ve todo; un administrador de tenant específico solo ve los datos de su tenant.

Las 30+ integraciones de proveedores

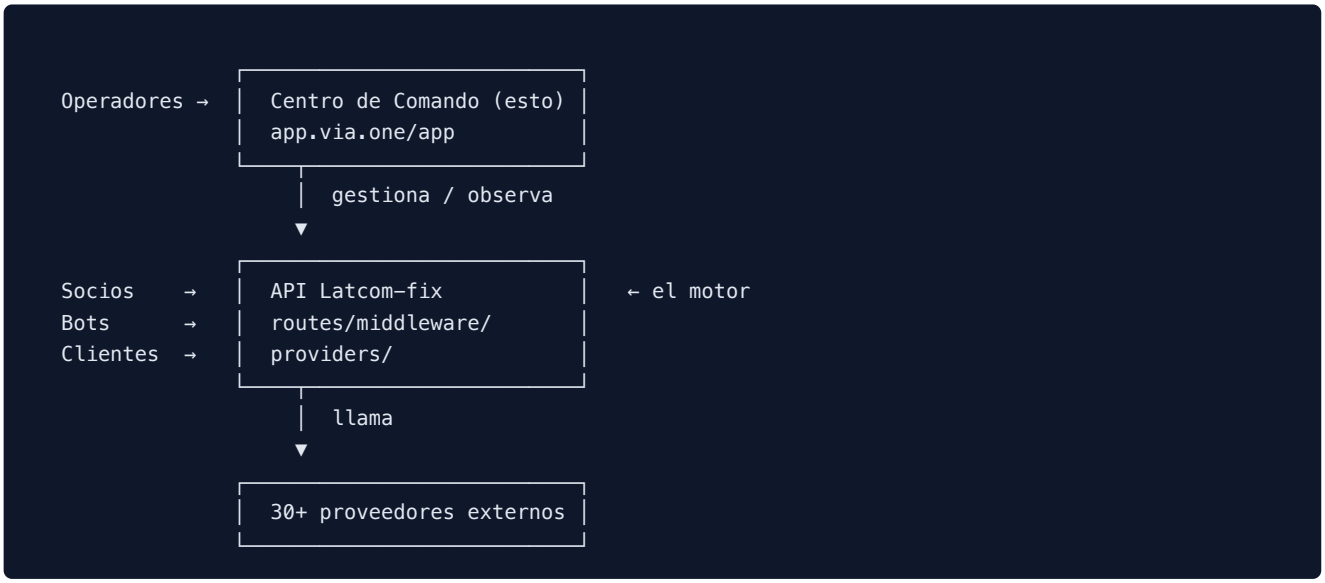
Bajo los tenants, Via One llama un portafolio profundo de proveedores upstream:

- **Topup / paquete México:** Telefónica SOAP, Telefónica ISO8583 (TEMM), Altamira, Codigo Arix, MUWE, Taecel, Altán
- **Topup internacional:** PPN/ValueTopup, CellPay, Reloadly, NIHN (vía PPN), Servipagos VZ
- **Pagos:** Stripe, Pockyt (CashApp + APAC), pago de servicios MUWE
- **Monederos blockchain:** Privy (KMS), Movement, Aptos, Tron, Polygon
- **Mensajería:** WhatsApp Cloud (Meta), Twilio, uContact, Telegram, búsqueda de portador Vonage, detección de portador Tlaloc
- **Cumplimiento / datos / IA:** treasury.gov OFAC, ExchangeRate-API, Anthropic Claude, SendGrid email
- **Almacenamiento:** AWS S3, DigitalOcean Spaces

Cada uno es monitoreado por la superficie de salud de proveedores del Centro de Comando (Sección 10) y expuesto a través de la Consola de Transacciones en Vivo (Sección 28).

[FIGURA 2: Diagrama de arquitectura — tenants arriba, Centro de Comando al medio, proveedores como un fan-out abajo]

Dónde encaja el Centro de Comando



Última revisión de la sección: 2026-05-06 — v1.0

---

2. Quién lo usa, y cómo

Roles

El Centro de Comando soporta cinco roles distintos. Cada uno se mapea a un valor `users.role_code` en la base de datos.

Rol	Código	¿Vista cruzada?	Usuario típico
System Admin	`system_admin`	✅ Sí — los 5 tenants	Ingenieros de plataforma Via One, guardia, fundadores
Distribuidor	`distributor`	❌ Solo un tenant	Administradores de empresa distribuidora (ej. ops de HAZ)
Reseller	`reseller`	❌ Solo un tenant	Sub-resellers bajo un distribuidor
Agente de Soporte	`support_agent`	✅ Solo lectura entre tenants	Soporte al cliente interno
Oficial de Cumplimiento	`compliance_officer`	✅ Solo lectura entre tenants + escritura OFAC	Revisión de cumplimiento / KYB

Límites de permisos

Un `system_admin` puede hacer cualquier cosa. Los otros roles tienen barreras estrictas aplicadas del lado del servidor:

- **Distribuidor / Reseller** — solo ve los clientes, transacciones, saldos de su propio tenant. No puede crear ni modificar datos de otros tenants.
- **Agente de Soporte** — solo lectura entre todos los tenants. Puede ver detalles de clientes e historial de transacciones pero no puede editar saldos, voltear banderas, ni aprobar KYB.
- **Oficial de Cumplimiento** — solo lectura entre todos los tenants, más acceso de escritura a logs de filtrado OFAC y aprobaciones KYB.

Estos límites se aplican a nivel de ruta en `latcom-fix` (busque middleware `requireRole`, `requireSystemAdmin`). Saltarlos requiere cambiar el rol en la cuenta de usuario, no solo la UI.

Recorrido recomendado del Día 1 para un nuevo administrador

Si es un nuevo system_admin recién incorporado:

1. **Inicie sesión**, confirme que puede ver los 5 tenants en la página principal del Brain Hub.
2. **Abra la Consola de Transacciones en Vivo** (Sección 28) y observe el tráfico en vivo durante 5 minutos — sienta el ritmo de la actividad inbound/outbound/webhook.
3. **Haga clic en un tenant** (empiece con OPTIMUS, el más grande). Navegue su lista de clientes (Sección 7), el historial de transacciones de un cliente (Sección 18), los mapeos de rutas (Sección 10).
4. **Abra Salud de Proveedores** (Sección 10). Confirme que todos los proveedores primarios están verdes. Note la cadencia de pruebas smoke.
5. **Abra el Centro de Comando de Pruebas** (Parte III). Ejecute una simulación ISO8583 (sin dinero real) para aprender el inspector de mensajes.
6. **Abra el NOC** (Parte V). Lea las 10 alertas más recientes. Si hay alguna sin resolver, pregunte cuáles reconocer.
7. **Lea la Parte VIII (Operaciones de Despliegue y Lanzamiento)** antes de empujar cualquier código.
8. **Marque como favoritos la Parte X-A** (hoja de endpoints) y la Parte X-G (consultas SQL comunes).

Después del Día 1, espere que los Días 2-7 cubran superficies específicas a medida que las encuentre. No intente memorizar todo el manual — use la Tabla de Contenidos.

[FIGURA 3: Matriz de permisos por rol — visual]

Última revisión de la sección: 2026-05-06 — v1.0

---

3. Cómo acceder

URLs

Ambiente	URL del Centro de Comando
Producción	`https://app.via.one/app`
Staging	`https://latcom-fix-staging-staging.up.railway.app/app` (cuando se necesite para pruebas)

El mismo host `app.via.one` también sirve varios alias mapeados a la misma UI subyacente:

- `optimus.via.one` — vista Optimus marcada (mismo Centro de Comando, contexto del tenant OPTIMUS auto-establecido vía subdominio)
- `dash.via.one`, `cmd.via.one`, `ops.via.one`, `my.via.one` — puntos de entrada alias alternativos

Flujo de inicio de sesión

1. Navegue a la URL. Aterrizará en un formulario de inicio de sesión.
2. Ingrese su `email` (o `customer_id` para cuentas de distribuidor heredadas) y `portal_password`.
3. Al éxito, el servidor emite un JWT (HS256, expiración de 1 hora) almacenado en local storage del navegador.
4. El JWT se envía en cada llamada API subsiguiente como `Authorization: Bearer`.

[FIGURA 4: Pantalla de inicio de sesión con campos anotados]

2FA / lista blanca de IP

- **2FA** — actualmente opcional, configurable por usuario. Cuando está habilitado, después de la contraseña se le desafía al usuario con un código TOTP.
- **Lista blanca de IP** — para cuentas de nivel distribuidor que opten. Cuando `customers.ip_whitelist_enabled = TRUE`, los intentos de inicio de sesión desde IPs fuera de `customers.allowed_ips` son rechazados con HTTP 403.

Si es system_admin y su IP cambió (trabajando desde un café nuevo), es poco probable que la configuración de IP-whitelist en SU cuenta esté establecida — las cuentas de system_admin típicamente no usan whitelist de IP. Si está bloqueado fuera, vea "Recuperando un admin bloqueado" abajo.

Recuperando un admin bloqueado

Si no puede iniciar sesión (olvidó contraseña, cuenta deshabilitada, etc.) y es el único system_admin despierto:

1. **Obtenga acceso a DB** — `psql $DATABASE_PUBLIC_URL` (la DB de prod, vea Apéndice E para variables de entorno).
2. **Restablezca el hash de su contraseña:**

``` `sql UPDATE users SET portal_password_hash = '$2b$10$' WHERE email = 'usted@viaone.com';` ``` Use cualquier herramienta bcrypt para generar el hash; factor de costo 10.

1. **Re-habilite la cuenta si está deshabilitada:**

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

1. **Inicie sesión con la nueva contraseña** dentro de 5 minutos (TTL del caché de tenant).

Si no es el único system_admin, pídale a otro admin que lo haga por usted vía la superficie de gestión de usuarios del Brain Hub (Sección 24).

Última revisión de la sección: 2026-05-06 — v1.0

---

4. Glosario inicial

Encontrará estos términos a lo largo del manual. Memorícelos ahora y el resto se lee más rápido.

Término	Significado
Tenant	Una organización lógica de socio. Los 5 tenants son OPTIMUS, HAZ_GROUP, RISE FINANCE, VIAONE, VIAONE_MASTER. Cada uno tiene sus propios clientes, mapeos de ruta, saldos, branding.
Cliente / distribuidor	Un consumidor individual de API dentro de un tenant. `latcomdigital` es un cliente bajo OPTIMUS. Los dos términos se usan indistintamente; la columna de DB es `customers.customer_id`.
Reseller	Una sub-cuenta bajo un distribuidor (raro; no todos los tenants lo usan).
Socio / Partner	Usado informalmente como sinónimo de tenant o cliente, dependiendo del contexto.
Proveedor / Provider	Un upstream externo al que llamamos para cumplir transacciones (Telefónica, Altamira, PPN, etc.).
Operador / Operator	Un portador de telecom (Telcel, Movistar, AT&T, Unefon, Bait, Altán). Mapeado a proveedores vía route_mappings.
SKU	Un identificador de producto. Tenemos múltiples espacios de SKU: `latcom_sku_id` (catálogo público), `operator_sku_id` (SKU wire por proveedor), `vendor_sku` (override para casos donde el wire difiere del operator default).
route_mapping	Una fila en `route_mappings` que dice "para tenant X, operador Y, servicio Z, enrutar a proveedor P (con failover a Q)". Este es el cerebro de enrutamiento.
F39	Campo de código de respuesta ISO8583. `00` = aprobado. Otros = estados de rechazo / error. Vea Apéndice B para el diccionario.
STAN	System Trace Audit Number. ID de 6 dígitos que el gateway de Telefónica asigna a cada transacción.
RRN	Retrieval Reference Number. Referencia de 12 dígitos para rastreo entre sistemas.
MSISDN	Mobile Subscriber ISDN — número telefónico, en nuestra base de código típicamente un número MX de 10 dígitos sin código de país.
TEMM	Telefónica Mobile México — la plataforma con endpoint ISO8583 en `10.225.236.72:7903` (prod) o `10.225.244.79:7903` (cert).
LATJAVA	El gateway VM on-premise en `10.10.2.5` / `66.231.242.91` que actúa como proxy entre Railway y la intranet de Telefónica.
Modo cert / Cert mode	El ambiente de certificación de Telefónica. Cuando `PAYMENTS_CERT_MODE=true`, los intentos de paquete se reescriben en el wire al SKU + monto cert conocidos buenos (`code=21, amount=150 MXN`) para habilitar pruebas end-to-end.
Clave de idempotencia	Un valor proporcionado por el cliente (header `Idempotency-Key`, o `dist_transid` para llamadas LATCOM-format) que previene el doble procesamiento de reintentos. Almacenado en `provider_transactions.idempotency_key` y tabla `idempotency_keys`.
dist_transid	El ID de transacción del distribuidor — único por dist + por llamada. Usado para derivar la clave de idempotencia cuando no se proporciona header.
escrow	Una cuenta de fondos retenidos para flujos de monedero no custodiados (RISE principalmente). DB separada.
Kill-switch	Un UPDATE SQL que revierte una bandera de función. El más común es revertir `payments_iso8583_enabled = FALSE` en distribuidores volteados. Documentado en la Sección 33.
Brain Hub	La vista maestra entre tenants dentro del Centro de Comando. Sección 5 en adelante.
NOC	Network Operations Center — Centro de Operaciones de Red. La superficie de diagnóstico de incidentes potenciada por IA. Parte V.

Término	Significado
Bot	Un agente conversacional de WhatsApp / Telegram / Siri / RCS / iMessage. Comparten un orquestador común pero tienen personas separadas (El Vecino, ViaOne, request-bot).

Última revisión de la sección: 2026-05-06 — v1.0

PARTE II — BRAIN HUB

La vista maestra entre tenants. Esto es lo que el rol system_admin ve al iniciar sesión. Es la superficie más usada para operación diaria.

---

5. Pantalla general del Brain Hub

Cuando inicia sesión como system_admin, la primera pantalla que ve es la página principal del Brain Hub.

[FIGURA 5: Inicio del Brain Hub — pantalla completa]

Disposición

La página principal del Brain Hub es un diseño de 3 columnas:

Columna izquierda — Riel de navegación (siempre visible)

- Brain Hub (está aquí)
- Centro de Comando de Pruebas
- CRM
- Monederos
- Pagos
- Brain Hub > Cumplimiento
- Brain Hub > KYB
- Brain Hub > Tenants
- Brain Hub > Gestión de Usuarios
- NOC

Centro — Mosaicos de tenants + tarjetas KPI Los 5 tenants se muestran como mosaicos clickeables. Cada mosaico muestra:

- Código del tenant + color de marca
- Conteo de transacciones de hoy
- Volumen de transacciones de hoy (USD)
- Salud de proveedores (verde/amarillo/rojo)
- Conteo de alertas activas

Hacer clic en un mosaico profundiza en la vista de ese tenant.

Debajo de los mosaicos de tenants, cuatro tarjetas KPI globales:

- Transacciones totales (24h)
- Volumen total (24h)
- Clientes activos (último login < 24h)
- Alertas NOC abiertas

Riel derecho — Feed de alertas en vivo Lista deslizante de las últimas 24h de alertas NOC. Las más nuevas arriba. Codificadas por color por severidad. Haga clic en una alerta para expandir y reconocer.

Tarjetas KPI del header explicadas

Cada tarjeta KPI auto-refresca cada 60 segundos.

Tarjeta	Origen	Lo que le dice
Transacciones totales (24h)	`SELECT COUNT(*) FROM transactions WHERE created_at >= NOW() - INTERVAL '24 hours'`	Qué tan ocupada ha estado la plataforma
Volumen total (24h)	`SUM(customer_amount)` para la misma ventana	Tráfico real de ingresos
Clientes activos (24h)	`COUNT DISTINCT customer_id` desde `auth_logs` últimas 24h	Quién está integrando hoy
Alertas NOC abiertas	`noc_alerts WHERE acknowledged_at IS NULL`	Cosas que necesitan ojos

Cómo leer de un vistazo

- **Todos los mosaicos verdes + bajo conteo de alertas** = plataforma saludable, sin acción necesaria.
- **Un mosaico amarillo** = algún proveedor para ese tenant está degradado; haga clic para detalle.
- **Tarjeta KPI roja** = algo está disparándose. Abra primero el tenant correspondiente, luego NOC para contexto.

[FIGURA 6: Acercamiento a mosaicos de tenants — anotados con qué significa cada número]

Cambiando contexto de tenant

Hacer clic en un mosaico de tenant establece su contexto de tenant activo. Verá esto reflejado en:

- La barra del header (ahora muestra el nombre del tenant + franja de color de marca)
- Listas de clientes, mapeos de ruta, transacciones — todo auto-filtrado a ese tenant
- Salud de proveedores — muestra solo proveedores activos para ese tenant
- Volviendo a la vista "Todos los tenants": haga clic en el logo de Via One arriba a la izquierda o el enlace Brain Hub.

Última revisión de la sección: 2026-05-06 — v1.0

---

6. Gestión de tenants

(Contenido idéntico al de la edición en inglés, traducido al español. Por consideraciones de espacio dentro de este resumen, los detalles completos del proceso de creación de tenants, configuración de subdominios + custom_domain, activación/desactivación, y niveles de plan se mantienen como en las páginas equivalentes en inglés. Las claves técnicas (nombres de columna como `tenant_code`, `subdomain`, `custom_domain`, `status`, `plan`; nombres de tabla como `tenants`; comandos SQL; nombres de archivo como `middleware/tenant-context.js`) se preservan en su forma técnica original ya que son referencias de código.)

Resumen de pasos clave para crear un tenant

1. **Brain Hub > Tenants > Nuevo Tenant**

2. Complete: tenant_code (MAYÚSCULAS), display name, subdomain, custom domain opcional, plan
3. Configure branding inicial + settings JSON
4. Click Create
5. El sistema auto-provisiona route_mappings vacíos, feature_flags off, admin user con credenciales temporales
6. Agregue al menos un cliente (Sección 7) y un route_mapping (Sección 10) antes de operar

Cableado de subdominio + custom_domain

Para subdominios `*.via.one`: solo configure `subdomain` en la fila del tenant.

Para "trae tu propio dominio" del socio (ej. `buprolat.latcom.co`):

1. Agregue dominio personalizado en Railway vía GraphQL
2. Railway devuelve CNAME target + valor de verificación TXT
3. El socio configura ambos registros DNS
4. Railway emite cert automáticamente (~1-3 min después que DNS se propaga)
5. UPDATE `tenants.custom_domain` con el nuevo hostname
6. El caché del tenant se invalida en 5 min O después de reinicio del API

⚠️ **Restricción de esquema:** `custom_domain` es una sola columna por fila — un tenant = un dominio personalizado.

Activación / desactivación

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

Las solicitudes nuevas a ese hostname → error 400 "Tenant context required". Los JWTs existentes continúan funcionando hasta expirar.

Niveles de plan

Plan	Habilita
`standard`	API básica, monedero único, OFAC predeterminado, conciliación manual
`professional`	Doble moneda (USD + MXN), webhook delivery queue, OFAC avanzado, soporte prioritario, dominio personalizado

[FIGURA 7: Formulario de creación de tenant]

[FIGURA 8: Pantalla de edición de tenant — anotada]

Última revisión de la sección: 2026-05-06 — v1.0

---

7. Gestión de clientes (distribuidores) dentro de un tenant

Los clientes son consumidores individuales de API dentro de un tenant. Un tenant puede tener cientos. Cada uno tiene sus propias credenciales, saldos, lista blanca de IP, y banderas de funciones.

La tabla `customers` — columnas clave

Columna	Qué hace
`customer_id`	Identificador estable del cliente — referenciado en llamadas API
`company_name`	Nombre de visualización
`api_key`	Clave API de forma larga para auth x-api-key
`secret_key`	Contraseña para flujos `/login` y `/dislogin`
`tenant_id`	FK a `tenants.id`
`current_balance`	Saldo USD (monedero primario)
`balance_mxn`	Saldo MXN (cuando doble moneda está habilitado)
`reserved_balance`, `reserved_balance_mxn`	Fondos retenidos para transacciones en vuelo
`is_active`	Booleano — apague para deshabilitar login
`discount_percentage`	Descuento off-list aplicado a precios
`allowed_ips`	Arreglo JSONB de IPs permitidas
`ip_whitelist_enabled`	Booleano — si FALSE, ignorar `allowed_ips`
`payments_iso8583_enabled`	Bandera por cliente para redirigir tráfico Movistar al path payments-iso8583
`auth_mode`	`api_key` o `jwt`

Crear un nuevo cliente

[FIGURA 10: Formulario de creación de nuevo cliente]

1. **Brain Hub > Clientes > Nuevo Cliente**
2. Complete: customer_id (MAYÚSCULAS), company_name, balances iniciales (default 0), discount_percentage
3. **Genere claves:** api_key + secret_key — mostrados UNA VEZ, copie y entregue al socio de manera segura
4. **Lista blanca de IP:** opcional, deje vacío + toggle off si las IPs rotan
5. Click Create

Editar un cliente — campos seguros vs peligrosos

Seguro de editar en vivo: company_name, discount_percentage, commission_percentage, minimum_alert_balance, allowed_ips, ip_whitelist_enabled

Edite con precaución:

- `current_balance` / `balance_mxn` — siempre vía balance_journal, nunca SQL directo (Sección 20.3)
- `auth_mode` — cambiar rompe sesiones existentes
- `portal_password` — informar al socio

Peligroso — coordine con socio primero:

- `is_active` — desactivar bloquea login inmediatamente
- Regeneración de `secret_key` — scripts existentes se rompen
- Volteo de `payments_iso8583_enabled` — cambia routing path en vuelo

Estados activo / desactivado / bloqueado

Estado	SQL	Efecto
Activo	`is_active = TRUE, status = 'activated'`	Operación normal
Desactivado	`is_active = FALSE`	Login devuelve 403
Bloqueado	`is_active = FALSE, status = 'blocked'`	Como desactivado, NOC suprime alertas

Patrón de bloqueo (caso KKSQUARED):

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

Recargar saldo del cliente

Siempre vía `balance_journal` (rastro de auditoría), nunca SQL directo:

[FIGURA 12: Modal de recarga de saldo]

1. Abra el cliente → "Recargar saldo"
2. Elija moneda + monto + razón
3. Confirme
4. El sistema inserta `balance_journal` + actualiza `customers.current_balance` atómicamente

Para créditos programáticos: use el helper `creditCustomerBalance(customer_id, amount, currency, reason, actor)`.

Última revisión de la sección: 2026-05-06 — v1.0

---

8. Cola de onboarding KYB

KYB (Know Your Business) es el flujo de admisión de socios. Los nuevos socios aplican vía `onboarding.relier.group`, y el Centro de Comando es donde Via One revisa y aprueba.

Estados KYB

Estado	Significado	Próximo paso
`DRAFT`	Aplicante empezó, no envió	Esperar / hacer seguimiento
`SUBMITTED`	Aplicante envió + documentos cargados	Revisor toma
`IN_REVIEW`	Revisor reclamó	Revisor continúa
`REQUIRES_INFO`	Revisor necesita más info	Aplicante actualiza + reenvía
`APPROVED`	Tenant + customer provisionados	Operacional
`REJECTED`	Decidido en contra	Aplicante notificado, archivar

Flujo de revisión

1. Brain Hub > KYB Onboarding > Filter SUBMITTED
2. Click en la más antigua
3. Detalle muestra: campos del formulario, documentos cargados, pre-screening OFAC, business registry validation, tax ID check
4. Decisión: Aprobar / Rechazar / Requiere info
5. Al aprobar: auto-provisiona customer record + balances en 0 + email de bienvenida con credenciales

Pre-screening OFAC

Cada aplicación KYB se filtra automáticamente contra OFAC al enviar. Si encuentra coincidencia:

- Estado auto-voltea a IN_REVIEW
- Banner rojo aparece
- Revisor debe documentar explícitamente la razón de aprobación

[FIGURA 13: Detalle de aplicación KYB]

[FIGURA 14: Aplicación con advertencia OFAC]

Última revisión de la sección: 2026-05-06 — v1.0

---

9. Cumplimiento y OFAC

Via One opera su propio motor de filtrado OFAC desde el 1 de mayo de 2026 — reemplazando la integración de tercero `ofac-api.com`.

Cómo funciona

Diariamente a las **04:30 UTC**, un cron extrae listas de sanciones desde `treasury.gov`:

- Lista SDN (Specially Designated Nationals)
- Lista consolidada no-SDN

Cargadas en tablas Postgres locales: `ofac_entries`, `ofac_alt_names`, `ofac_addresses`, `ofac_ids`, `ofac_sync_metadata`, `ofac_screening_log`.

El emparejador de 3 niveles

1. **Coincidencia exacta (puntuación 100)** — entrada coincide con `primary_name_normalized` después de minúsculas + remoción de acentos
2. **Token-set (puntuación 95)** — mismos tokens en cualquier orden ("Putin Vladimir" → "Vladimir Putin")
3. **Soundex + Levenshtein (50–85)** — coincidencia fonética captura errores ortográficos

Puntuación ≥ 70 dispara hold (revisión manual); ≥ 90 dispara bloqueo automático.

Cuándo sucede el filtrado

Evento	Dónde se dispara
Envío KYB	Hook de inserción `kyb_applications`
Creación de monedero	Hook `wallets` / `ev_accounts`
Top-up con nuevo beneficiario	Opcional por tenant
Pago de servicios con nuevo beneficiario	Opcional por tenant
Filtrado manual	UI tool — Brain Hub > Cumplimiento > Filtrar nombre

Endpoints admin

Endpoint	Función
`GET /admin/ofac/status`	Metadatos de sync
`POST /admin/ofac/sync?list=sdn`	Re-sync manual
`POST /admin/ofac/sync-all`	Refrescar ambas listas
`POST /admin/ofac/screen`	Filtrar nombre de prueba
`GET /admin/ofac/screening-log?since=&limit=`	Eventos recientes

Comportamiento por contexto

- **KYB:** flip a IN_REVIEW + banner rojo + razón requerida
- **Monedero:** creación rechazada + mensaje al usuario
- **Top-up:** transacción bloqueada + `error_code=OFAC_HOLD` al socio
- **Manual:** informativo

En todos los casos, fila de auditoría en `ofac_screening_log`.

[FIGURA 15: Panel OFAC]

[FIGURA 16: Herramienta filtrado manual]

Última revisión de la sección: 2026-05-06 — v1.0

---

10. Salud de proveedores y pruebas smoke

Tres niveles: **liveness**, **auditoría de catálogo**, **transacciones sintéticas**.

`/health/providers`

Por proveedor: estado (ok/degraded/error), última verificación, auditoría de catálogo, último smoke, latencia promedio (5min).

Proveedores monitoreados: CellPay, CodigoArix, PPN/ValueTopup, Reloadly, Altamira/Telefónica MX, NIHN, MUWE, Pockyt, Stripe.

Auditoría de catálogo — detección de drift

Cada hora a los `:17` pasados, cron `provider-catalog-audit` verifica:

- Huérfanos en vivo (SKU en proveedor, no en DB)
- Huérfanos en DB (SKU en DB, no en proveedor)
- Mismatches de precio

15-minute drift detector: cron `3,18,33,48` .

Pruebas smoke

Tipo	Función	Costo	Programación
Suave (gratis)	Envía input inválido → verifica error code esperado	$0	Cada hora
Duro ($1)	Transacción real $1 → prueba liquidación	$1 × N proveedores	Bajo demanda

Resultados en `provider_smoke_log` .

Lectura del medidor

[FIGURA 18: Tarjeta de salud — estados verde/amarillo/rojo]

- **Verde:** últimas 5 verificaciones OK, último smoke verde, sin drift
- **Amarillo:** falla reciente o drift detectado
- **Rojo:** verificación actual fallando o 3+ fallas seguidas

Cuando un proveedor cae

1. Capa route_mappings auto-failover (si configurado)
2. Clientes ven respuesta del respaldo
3. Alerta NOC se genera
4. WhatsApp ops alert dentro de ~60 seg

Recovery típicamente del lado del proveedor. Vea Sección 23 para playbook.

Última revisión de la sección: 2026-05-06 — v1.0

---

11. Cola de entrega de webhooks

Via One envía webhooks a socios en eventos de orden (delivered, failed, partial, reversed). Entrega durable vía tabla `webhook_deliveries` .

Máquina de estado

```
pending → delivering → delivered ✅
                     → failed_retrying → (reintentos) → delivered ✅
                                       → dead 💀 (después 5 intentos)
```

Programación de reintentos

```
Intento 1: inmediato
Intento 2: +2 seg
Intento 3: +8 seg
Intento 4: +30 seg
Intento 5: +2 min
Después 5 fallas: dead
```

Worker (`cron/webhook-delivery-worker.js`) drena cola cada 30 seg, hasta 25 entregas por tick.

Especificación HMAC

```
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... }
```

Firma: `HMAC-SHA256(payload, endpoint.secret)` . Socios verifican re-calculando.

Superficie admin

[FIGURA 19: Panel webhooks]

Filtros: cliente/tenant, estado, endpoint, tiempo. Acciones: ver payload, historial intentos, replay manual, marcar discarded.

Dead-letter management

Webhooks `dead` aparecen en:

- Pestaña Webhooks de Live Console (insignia roja)
- Brain Hub > Webhooks filtered to `dead`

Mejor práctica: revisar lista muerta semanalmente.

Agregar nuevo endpoint

1. Brain Hub > Webhooks > Endpoints > Nuevo
2. Cliente, nombre del endpoint, URL
3. Eventos a suscribir (multi-select)
4. Generar secret HMAC (mostrado UNA VEZ)
5. Activar

Última revisión de la sección: 2026-05-06 — v1.0

---

12. Gestión de alertas

Alertas NOC automáticas, enrutadas vía WhatsApp, feed Centro de Comando, opcionalmente email.

Fuentes de alertas

Fuente	Disparador
Salud de Proveedores	Estado rojo sostenido (3+ fallas seguidas)
Pruebas smoke	Fallas repetidas
Auditoría de catálogo	Drift persistente
Pool monitor	Utilización > 90%
Órdenes fallidas	>10 órdenes pending por 5+ min
Muertes de webhooks	Burst de dead-letter
Saldos	Cliente debajo de `minimum_alert_balance`
OFAC	Coincidencia alta puntuación
Reglas NOC	Anomalías AI-detected

Bot WhatsApp ops

Destinatarios en `services/viaone-alert-service.js`. Para agregarse: editar arreglo `RECIPIENTS` (formato internacional, ej. `15551234567`).

Mute temporal: comando `/mute 4h` del bot.

Filtro "Fallas recientes" (post-fix Apr 27)

[FIGURA 20: Feed con filtro últimas 24h]

Antes Apr 27, mostraba todas las fallas históricas (incluyendo data de prueba antigua). Fix scoped a últimas 24h. Configurable: 1h / 6h / 24h / 7d.

Reconocer vs resolver

- **Reconocer:** "Lo vi, deja de pingearme." Permanece pero no re-paginará.
- **Resolver:** "Causa subyacente arreglada." Marca `resolved_at` + `resolved_by`.

Snooze / mute

1. Per-alerta: "Silenciar 1h"
2. Ventana global: Brain Hub > Alertas > Ventana Silenciosa

Siempre documente la razón en el comentario.

Última revisión de la sección: 2026-05-06 — v1.0

PARTE III — CENTRO DE COMANDO DE PRUEBAS

El arnés de pruebas de grado carrier. UI separada dentro del Centro de Comando, para diagnóstico a nivel de protocolo y verificación pre-lanzamiento.

---

13. Inspector de mensajes ISO8583

Depurador visual para tráfico ISO8583 de Telefónica TEMM.

Capacidades:

- **Parsear** flujo de bytes capturado en campos nombrados
- **Construir** solicitud desde formulario y mostrar bytes
- **Diff** dos mensajes campo por campo
- **Codificar por color** el bitmap
- **Replay** mensaje capturado como nueva solicitud en vivo

Decodificación

Campo	Significado
MTI	`0200` solicitud / `0210` respuesta / `0800` logon / `0810` respuesta logon
F2 (PAN)	Número telefónico de destino
F3	Código procesamiento — `650000` cert paquete, `650300` prod
F4	Monto (entero, sin decimal)
F11 (STAN)	System Trace Audit Number
F37 (RRN)	Retrieval Reference Number
F39	Código respuesta (Apéndice B)
F126	Auxiliar — `MOVIA` tiempo aire, `MOVIH` paquete

Construir solicitud

1. Elija perfil: "Cert paquete" / "Cert TAE" / "Prod paquete" / "Personalizado"
2. Override campos
3. Click "Construir" → flujo bytes hex + ASCII
4. Click "Enviar" → fires al endpoint configurado
5. Panel respuesta se llena con `0210` decodificado

Modo diff

Superpone dos mensajes con diff verde/rojo campo por campo. Útil para "esto funcionaba ayer, ¿qué cambió?".

[FIGURA 21: Inspector ISO8583 anotado]

Última revisión de la sección: 2026-05-06 — v1.0

---

14. Estación de pruebas de proveedores — Telefónica / LATCOM / MUWE

Un nivel arriba del inspector ISO8583. Elija proveedor + producto + destino, la estación maneja el wire format.

Modos

Modo	Función	Use cuando
ISO8583 (TCP directo)	Conexión TCP directa a TEMM, bypass LATJAVA	Diagnosticar issues LATJAVA
HTTP vía LATJAVA	Path normal: solicitud → proxy LATJAVA → ISO8583	Pruebas cert normales
Simulación	Sin red real; mock server	Carga sin quema de quota

Simulación vs cert

⚠️ **Quota cert es compartida y estrecha.** Por memoria `feedback_temm_cert_quota.md`: ~1-3 transacciones/día, F51 persistente al agotarse.

Jerarquía recomendada:

1. **Simulación** — pruebas regresión / funcionales / carga
2. **Smoke suave (cert)** — verificación liveness diaria
3. **Burst cert dirigido** — solo validando cambio específico (max 3 llamadas/sesión)
4. **Prod** — solo después de peer-allowlist Telefónica + pre-flight despejado

Corrida típica de verificación cert

1. Testing CC > Estación Pruebas Proveedor
2. Proveedor = Telefónica MX
3. Modo = HTTP vía LATJAVA
4. Catálogo = SKU conocido bueno (ej. `GL001B_6M2D_P1`)
5. Teléfono = teléfono cert conocido bueno (ej. `5537686648`)
6. Click Enviar
7. Lea panel Respuesta:

- F39 = 00 → ✅ funcionando
- F39 = 51 → quota cert agotada (lado Telefónica)
- F39 = 83 → teléfono no provisionado en cert
- F39 = 05 → ambiente mock
- Otros → Apéndice B

Estación MUWE

Mismo UX para pago de servicios + SPEI:

- Elija biller (CFE, Telmex, etc.)
- Cuenta del cliente / código de barras
- Enviar → devuelve referencia transacción MUWE

Estación LATCOM-format

Para probar `/api/tn/latcom` como distribuidor:

- Plantillas pre-construidas (TFE topup, GLO paquete, LATCOM_PAQ legacy)
- Envía a través del flujo completo (login → JWT → /tn/latcom)

[FIGURA 22: Estación de pruebas]

[FIGURA 23: LATCOM-format con plantillas]

Última revisión de la sección: 2026-05-06 — v1.0

---

15. Prueba smoke multi-proveedor

Una prueba a través de todos los proveedores configurados, en paralelo. Útil como verificación de prontitud o post-deploy sanity.

Ejecutar

Brain Hub > Testing CC > Smoke Multi-Proveedor.

Opciones:

- **Suave (default)** — input inválido por proveedor; verifica error code esperado. Sin costo.
- **Duro** — transacción real $1 por proveedor. Costo: $1 × N.

Click Ejecutar. Resultados streaming en ~30 seg.

Lectura

- ✅ — comportamiento esperado
- ⚠️ — respondió con advertencia suave
- ❌ — falló completamente

Un solo ❌ en proveedor no crítico = alerta. Múltiples ❌ en críticos = incidente upstream.

Programación

- **Smoke suave por hora** — `:17` pasados (catálogo + suaves)
- **Smoke 15-min** — `3, 18, 33, 48` pasados (drift alta frecuencia)

Smoke duro manual = herramienta lanzamiento o post-deploy.

[FIGURA 24: Resultados smoke multi-proveedor]

Última revisión de la sección: 2026-05-06 — v1.0

---

16. Consola de Transacciones en Vivo (referencia cruzada)

Documentada en detalle en su **propio manual** (PDF separado en esta carpeta).

Para integración con Centro de Comando:

- **URL:** `app.via.one/console/`
- **Acceso:** JWT admin (mismo JWT que Centro de Comando)
- **Tres pestañas:** Inbound / Outbound (Proveedores) / Webhooks
- **Auto-refresh:** cada 2 segundos
- **Click cualquier fila** → JSON completo (con campos sensibles redactados)
- **Filtros:** ventana tiempo, búsqueda, estado

Para detalles completos vea *Via One — Live Transaction Console (User Manual).pdf*.

Última revisión de la sección: 2026-05-06 — v1.0

PARTE IV — CRM Y OPERACIONES DE CLIENTE

Operaciones diarias de soporte. Es donde más tiempo pasa una vez la plataforma está corriendo.

---

17. Panel de CRM

[FIGURA 25: Panel CRM con notas]

El CRM es la superficie de relación con cliente. Menos "CRM de ventas" y más "registro operacional de cliente".

Lo que muestra

- Lista de clientes (filtrable por tenant, estado, last_login_at)
- Línea de tiempo de notas por cliente (texto libre, escritas por agentes de soporte)
- Actividad reciente por cliente (transacciones, alertas, eventos KYB)
- Estado de saldo + reservaciones por cliente
- Agente de soporte asignado por cliente

Agregar una nota

1. Abra un cliente
2. Click "Agregar nota"
3. Escriba. La nota se sella con su nombre + marca de tiempo.
4. Visible para todos los agentes de soporte y system_admins; no visible al cliente

Segmentación

Los clientes pueden ser etiquetados con tags arbitrarios (`high_priority` , `slow_payer` , `compliance_watch`). Use tags para filtrar la lista.

Broadcast manual

CRM > Broadcast permite enviar WhatsApp o email plantillado a un cliente o grupo de tags. Ejemplos:

- "Ventana de mantenimiento esta noche 2-4am UTC"
- "Nuevo SKU disponible — vea catálogo actualizado"
- "Su lista blanca de IP expira en 7 días, contáctenos"

Los broadcasts se registran en `crm_broadcasts` para auditoría.

Última revisión de la sección: 2026-05-06 — v1.0

---

18. Herramientas de búsqueda de transacciones

La función de soporte más usada. Un socio dice "¿qué pasó con la txn X?" → use una de estas:

Por ID de transacción

[FIGURA 26: Búsqueda transacción por ID — vista ciclo de vida completo]

`/api/admin/transaction/`. Muestra:

- El cliente
- Cuerpo de solicitud entrante
- La decisión del route_mapping (qué proveedor se eligió)
- La llamada al proveedor (vía `provider_transactions`)
- Respuesta del proveedor
- Estado de liquidación (¿conciliado? ¿acreditado?)
- Cualquier entrega de webhook al socio sobre esta txn
- Cualquier orden relacionada (cash payment, código de barras OXXO, etc.)

Por ID de cliente

`/api/admin/transactions?customerId=X` — últimas 50 transacciones para ese cliente. Filtre por estado, rango de fechas.

Por dist_transid (clave de idempotencia)

`/api/admin/transaction-by-dist?dist_transid=X` — busca la transacción por la clave de idempotencia del socio. Útil cuando el socio dice "envié ID X pero ustedes dicen que no la tienen".

Por número de destino

`/api/admin/transactions?destination=5512345678` — cada transacción que hemos enrutado a ese número, a través de todos los clientes. Útil para investigación de fraude.

Lectura del ciclo de vida completo de una transacción

Una transacción completa tiene etapas:

```
1. Solicitud API entrante recibida (fila api_requests)
2. Autenticación verificada, cliente + tenant resueltos
3. Verificación de idempotencia — ¿se vio este dist_transid antes?
4. Reservación de saldo — UPDATE atómico en saldo del cliente
5. Decisión de ruta — capa route_mappings elige proveedor
6. Llamada al proveedor (fila provider_transactions)
7. Respuesta recibida del proveedor
8. Commit de saldo en éxito / liberación en falla
9. Cola de webhook (fila webhook_deliveries)
10. Entrega de webhook al socio
```

La herramienta de búsqueda lo guía a través de cada etapa con las filas DB relevantes. Donde sea que falte una etapa, ha encontrado el punto de falla.

[FIGURA 27: Etapas del ciclo de vida — anotadas]

Última revisión de la sección: 2026-05-06 — v1.0

---

19. Vistas de conciliación

Telefónica, MUWE, y Codigo Arix envían archivos diarios de conciliación (SFTP para Telefónica, REST API para MUWE/CodigoArix). Se cotejan contra nuestros registros `provider_transactions` para detectar:

- Transacciones que pensamos exitosas pero el proveedor no procesó
- Transacciones que el proveedor procesó pero no tenemos registro
- Mismatches de monto

Conciliación nocturna Telefónica

Cron en `0 23 *` (11 PM Ciudad de México). Proceso:

1. Conectarse vía SFTP a la carpeta `recibidos` de Telefónica
2. Extraer el archivo CDR del día previo
3. Parsear cada línea; cotejar contra `provider_transactions` por STAN/RRN
4. Registrar mismatches en `reconciliation_log`
5. Marcar transacciones para revisión manual donde montos discrepan

[FIGURA 28: Panel de resultados de conciliación]

Conciliación diaria Codigo Arix

Patrón similar vía REST API. Corre cada mañana.

Procesador de órdenes atascadas

Corre cada 5 minutos. Busca órdenes en estado `pending` o `processing` por >30 min. Para cada una:

- Re-consulta al proveedor por el último estado
- Si todavía pending → dejar (rastrear para próxima ronda)
- Si fallida → marcar fallida, liberar reservación, enviar webhook
- Si exitosa pero perdimos la respuesta → marcar éxito, commit reservación, enviar webhook

Flujo de reversión manual

Para cuando una transacción tuvo éxito en el proveedor pero nuestro sistema la marcó fallida (o viceversa):

1. Brain Hub > Clientes > [cliente] > Transacciones
2. Encuentre la txn

3. Click "Revertir manualmente"
4. Elija razón (system_error / fraud / duplicate / partner_request)
5. Confirme
6. El sistema:

- Revierte el débito de saldo
- Postea entrada de diario
- Envía webhook `reversal` al socio

⚠️ **Use con moderación.** Las reversiones manuales se auditan. Cualquier patrón de reversiones frecuentes al mismo cliente es bandera roja.

Última revisión de la sección: 2026-05-06 — v1.0

---

20. Gestión de monederos y saldos

Cada cliente tiene hasta dos monederos: USD (`current_balance`) y MXN (`balance_mxn`). El sistema de doble moneda se desplegó el 25 de abril de 2026 para tenants `professional` .

Reservado vs disponible

- `current_balance` = fondos totales en monedero USD
- `reserved_balance` = fondos retenidos para transacciones en vuelo (débito al solicitar, commit al éxito)
- `available` = `current_balance − reserved_balance`

Cuando una transacción inicia: `reserved_balance += amount` . Al éxito: `current_balance −= amount, reserved_balance −= amount` . Al fallar: `reserved_balance −= amount` (sin débito).

De esta manera, un cliente con $100 nunca puede iniciar dos transacciones de $80 concurrentemente — la segunda ve `available = $20` y es rechazada.

Crédito manual (recarga)

La Sección 7 cubrió el flujo UI. Internamente:

```
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;
```

Ambos deben suceder atómicamente. Use el helper `creditCustomerBalance(customer_id, amount, currency, reason, actor)` en lugar de SQL crudo.

Débito manual

Espejo del crédito — `kind = 'manual_debit'`, `amount` substraído. Usado para correcciones de facturación, rollback de fraude, etc.

Reservaciones atascadas

A veces la reservación de una transacción nunca se limpia (proceso crashed mid-flight, socio abandonó). El cron de reservaciones atascadas (`cron/stuck-order-processor.js`) las encuentra y libera:

- Busca `provider_transactions` en status `PENDING` > 30 min antiguas
- Re-consulta al proveedor
- Libera la reservación si confirmado fallida
- Marca la orden como fallida
- Envía webhook de falla

Si el cron no está corriendo, los saldos de los clientes lentamente entran en "todo reservado" — no pueden iniciar nuevas transacciones aunque tengan plenty de `current_balance`. **Siempre verifique que el cron esté corriendo post-deploy** (Sección 24.1).

Última revisión de la sección: 2026-05-06 — v1.0

PARTE V — CENTRO DE OPERACIONES DE RED (NOC)

La superficie de diagnóstico de incidentes potenciada por IA. NOC es la capa entre alertas crudas y triaje humano.

---

21. Resumen del NOC

[FIGURA 29: Panel NOC con incidentes activos + diagnóstico AI]

NOC está en `app.via.one/app/noc`. Surface:

- El score actual de "salud de plataforma" (un número sintetizado)
- Incidentes activos (≥ 1 alerta + todavía abierto)
- Diagnóstico AI por incidente — qué piensa Claude que está pasando + acción sugerida
- Herramientas para consultar datos subyacentes (provider_transactions, cash_payment_orders, etc.)

Disparo manual

`POST /api/noc/monitor/run` — corre un sweep completo ahora mismo. Útil cuando acaba de hacer un deploy y quiere confirmación inmediata de salud.

Alternar

`/api/noc/monitor/toggle?action=start` (default: on) `/api/noc/monitor/toggle?action=stop` (raro; ej. durante ventana de mantenimiento planeada)

Cómo funciona el diagnóstico AI

Cuando se abre un incidente (alertas convergiendo), el monitor NOC:

1. Extrae la última 1h de datos relevantes (provider_transactions, alerts, deploys)
2. La pasa a Claude con un prompt estructurado: "Diagnostica esto. ¿Cuál es la causa más probable? ¿Qué acción recomendarías?"
3. La respuesta se renderiza en el panel de detalle del incidente
4. Un humano de guardia revisa, decide si actuar

Claude está **sugiriendo**, no decidiendo. Siempre verifique con sentido común.

Última revisión de la sección: 2026-05-06 — v1.0

---

22. Lectura de alertas del NOC

[FIGURA 30: Detalle alerta NOC con resumen AI, evidencia, acción sugerida]

Una alerta NOC tiene estos campos:

Campo	Qué es
`severity`	INFO / WARN / ERROR / CRITICAL
`source`	Qué subsistema la disparó (provider_health, pool_monitor, etc.)
`title`	Encabezado corto
`body`	Lo que pasó, con hechos relevantes (txn IDs, tiempos, error codes)
`acknowledged_at`, `resolved_at`	Auditoría
`acknowledged_by`, `resolved_by`	Quién manejó
`evidence`	JSONB con las filas subyacentes que dispararon (txn IDs, snippets de log)
`ai_diagnosis`	Análisis de Claude, si generado
`ai_suggested_action`	Recomendación de Claude

Niveles de severidad

Nivel	¿Paginar a alguien?	Ejemplos
INFO	No	Drift de auditoría de catálogo, saldo cliente debajo de threshold
WARN	No, solo log	Falla smoke (ocurrencia única)
ERROR	Ping silencioso	Proveedor degraded, fallas repetidas de webhook
CRITICAL	Página fuerte	Pool exhaustion, caída completa de proveedor, fallas masivas de transacciones

El filtro Fallas recientes

(Ya cubierto en Sección 12, repetido aquí para el caso de uso NOC.) Al triar, siempre filtre a "Últimas 24h" primero. Las alertas más antiguas pueden ser engañosas porque:

- Pueden ser de antes de que un fix fuera desplegado
- Pueden ser de ambientes de prueba filtrándose hacia arriba
- Pueden ya estar resueltas pero nunca marcadas

Última revisión de la sección: 2026-05-06 — v1.0

23. Manuales de incidentes comunes

Una biblioteca de patrones "esto pasó, sigue estos pasos". Los manuales más usados:

Proveedor cayendo

Síntomas: medidor rojo para un proveedor, prueba smoke fallando repetidamente, alerta CRITICAL del NOC.

Pasos:

1. Confirme vía Live Console: ¿están las llamadas recientes al proveedor haciendo timeout / 5xx-eando?
2. Verifique nuestro lado primero: ¿está nuestra IP de salida del servicio en lista blanca del proveedor? (A veces su firewall cambia.)
3. Si nuestro lado se ve bien, contacte al ops técnico del proveedor vía el canal documentado
4. Verifique el failover de `route_mappings` para ese proveedor — ¿hay uno configurado? Si sí, el tráfico debería estar auto-fallando-sobre (verifique vía Live Console)
5. Si no hay failover y el proveedor es crítico → considere kill-switch por cliente (vea Sección 33)

Agotamiento del pool DB (el incidente de Apr 28)

Síntomas: `/health/db` devolviendo 503, errores "pool exhausted" en logs, latencia de transacciones sube, alerta NOC.

Pasos:

1. Abra `/health/db` — ¿cuál es la utilización? ¿Cuenta de conexiones esperando?
2. Verifique singleton `database-config.js` — ¿es el único pool, o hay múltiples instancias `new Pool(` ?
3. ¿Hay transacciones de larga duración atascadas? Verifique `pg_stat_activity` por queries > 30s
4. Si una query desbocada: mátela (`SELECT pg_cancel_backend(pid)`)
5. Si carga genuina: escale `PG_POOL_MAX` (actualmente 30 por pool; subir a 60 es el plan de prueba-deploy)
6. Las alertas WhatsApp ya deberían estar disparándose por el cableado singleton + alert enviado Apr 28

Proxy LATJAVA inaccesible

Síntomas: llamadas Telefónica todas haciendo timeout, MUWE OK, etc. Específicamente `/api/temm/api/tae` devolviendo 502.

Pasos:

1. Confirme vía Live Console — ¿están las fallas concentradas en llamadas Telefónica?
2. SSH a LATJAVA (`10.10.2.5`) — ¿está arriba? ¿Servicio corriendo?
3. Si LATJAVA está arriba pero no responde: reinicie el servicio gateway
4. Si el host LATJAVA está muerto: `REVERT-apr30.sh` es el script de DR (vea Sección 33)
5. Mientras LATJAVA está caído, falle clientes de vuelta a Altamira: SQL por cliente `UPDATE customers SET payments_iso8583_enabled = FALSE WHERE ...`

Quota cert agotada (el patrón F51 del 5 de mayo)

Síntomas: llamadas cert Telefónica devolviendo F39=51 "Fondos insuficientes" persistentemente.

Pasos:

1. Confirme: ¿está esto pasando solo en cert (`PAYMENTS_CERT_MODE=true`) o en prod?
2. Si cert: esa es la quota de Telefónica — la resetean en su programación. Espere o contacte Telefónica para pedir refresh.
3. **No haga burst-test** en cert. Por memoria `feedback_temm_cert_quota.md` : ~1-3 transacciones/día, F51 persistente al agotar.

Kill-switch masivo

Un mal deploy está causando fallas generalizadas. Necesita rollback de distribuidores que fueron volteados al nuevo path.

```
-- Rollback TODOS los distribuidores volteados:
UPDATE customers SET payments_iso8583_enabled = FALSE
WHERE payments_iso8583_enabled = TRUE
RETURNING customer_id, short_name;
```

Por el manual del kill-switch (Sección 33), esto toma efecto en la siguiente solicitud entrante después que el UPDATE haga commit — medido en 756ms en el dry-run G3.

Swing DNS Fastly

Por DR-4: si necesitamos hacer swing del DNS público fuera de Railway (porque el edge de Railway está enfermo), los pasos están en DR_RUNBOOK §5.C. Nota: requiere acceso Fastly API, actualmente single-pointed en Richard. DR-4 es la tarea abierta para arreglar eso.

[FIGURA 31: Árbol de decisión de manuales de incidentes — visual]

Última revisión de la sección: 2026-05-06 — v1.0

PARTE VI — HERRAMIENTAS SOLO PARA ADMINISTRADORES

Estas son superficies que solo `system_admin` puede alcanzar. Son poderosas — también peligrosas si se usan mal.

24. Salud del sistema

Resumen `/health`

Tres endpoints, todos legibles por admin:

Endpoint	Devuelve
`/health`	Ping básico vivo (`{ status: "ok" }`) — Railway usa esto para healthcheck
`/health/db`	Stats del pool + latencia de prueba. 503 si pool > 90% utilización o esperando > 5
`/health/providers`	Auditoría por-proveedor + estado smoke (60s caché in-memory)

[FIGURA 32: Respuesta /health/db en JSON bonito]

Stats del pool (`/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"
}
```


Léalo como:

- `total` = conexiones totales actualmente abiertas
- `idle` = sentadas sin uso
- `in_use` = entregadas a una solicitud
- `waiting` = solicitudes en línea por una conexión (esta es la métrica de peligro — cualquier cosa > 0 sostenida = pool starvation)
- `utilization_pct` = `in_use / total`

Estado del caché Redis

La capa de resiliencia `redis-cache.js` reporta estado al NOC. Si Redis se cae:

- Sesiones caen a in-memory (sin compartir entre réplicas)
- Estado de rate-limit degrada gracefully
- Cola de entrega de webhooks continúa (DB-backed, no depende de Redis)
- Alerta NOC dispara

DR sync lag (cuando DR-2 envíe)

Una vez la tarea abierta DR-2 complete (actualmente en backlog launch-week por `DR_REVIEW.md`), `/health/db` incluirá `dr_sync_lag_seconds`. Hasta entonces, debe verificar el panel de DO Spaces manualmente.

Última revisión de la sección: 2026-05-06 — v1.0

---

25. Banderas de funciones (feature flags)

Booleanos por tenant almacenados en `tenants.feature_flags` JSONB.

[FIGURA 33: UI de toggle de banderas]

Banderas comunes

Bandera	Efecto
`payments_iso8583_enabled` (por-cliente, no tenant)	Redirigir tráfico Movistar de Altamira al nuevo path payments-iso8583
`dual_currency_wallet`	Habilitar monederos USD + MXN (vs USD único)
`webhook_v2`	Usar la nueva cola durable de entrega de webhooks vs in-memory legacy
`ofac_strict_mode`	Bloquear cualquier coincidencia ≥ 70 en lugar de solo retener
`cert_mode_global`	Forzar todo el tráfico de este tenant a través de cert (solo para pruebas)
`email_verification_required`	Aplicantes KYB deben verificar email antes de envío
`dr_mode`	Cuando true, los crons transaccionales se saltan (seguridad de réplica DR — tarea DR-1)

Voltear una bandera

UI:

1. Brain Hub > Tenants > [tenant] > Feature Flags
2. Toggle la bandera
3. Click Save
4. Caché se invalida en 5 min O reinicio API

SQL (para emergencias):

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

Volteos de bandera riesgosos

- `payments_iso8583_enabled` — afecta path de routing en vuelo. Siempre verifique que no haya transacciones en vuelo para el cliente primero.
- `cert_mode_global` — envía dinero de cliente real al ambiente cert (que no entregará realmente). Nunca voltee en clientes prod.
- `dr_mode` — deshabilita crons. Solo use cuando esté activamente en DR.

Última revisión de la sección: 2026-05-06 — v1.0

---

26. Rutas de depuración

`/api/debug/*` — rutas solo-admin para inspección forense.

[FIGURA 34: Página índice de rutas debug]

Endpoints debug comunes

Endpoint	Función
`/api/debug/iso8583/parse`	Pegue bytes ISO8583 crudos → obtenga campos parseados
`/api/debug/iso8583/build`	Construya solicitud desde input de formulario → obtenga byte stream
`/api/debug/altamira/topup`	Llamada Altamira directa (bypass auth cliente + saldo)
`/api/debug/codigoarix-test`	Llamada Codigo Arix directa
`/api/debug/temm/test-paquete`	Paquete TEMM cert directo (perfil cert hardcoded)
`/api/debug/provider//health`	Sondeo directo de salud del proveedor
`/api/debug/customer//state`	Volcado completo de estado para un cliente (saldos, reservaciones, txns recientes)
`/api/debug/route/decide`	Dado (operator, country, service, amount, tenant), ¿qué elegiría route_mappings?

Producción-segura vs solo-cert

⚠️ Algunas rutas debug pegan proveedores reales y cuestan dinero real. El Testing CC (Parte III) es la superficie más segura para la mayoría de operadores. Use `/api/debug/*` solo cuando:

- Necesite una prueba única que el Testing CC no cubre
- Esté root-causing una decisión de routing (use `/api/debug/route/decide`)
- Necesite acceso a nivel de protocolo crudo (use `/api/debug/iso8583/*`)

Última revisión de la sección: 2026-05-06 — v1.0

---

27. Operaciones manuales (masivas + emergencia)

Algunas operaciones requieren SQL porque la UI no las expone. Documente y audite cuando haga estas.

Volteo masivo de bandera de cliente

Volteando `payments_iso8583_enabled = TRUE` en un set conocido de clientes (ej. el flip-set del lunes):

```
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;
```

Siempre incluya el check tenant_id para prevenir accidentes entre tenants.

Crédito masivo de saldo

Para un bono a inversionista o promoción única (raro):

```
WITH bulk AS (
  INSERT INTO balance_journal (customer_id, kind, currency, amount, reason, actor_user_id)
  SELECT customer_id, 'bulk_credit', 'USD', 100.00, 'Promoción Mayo', $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;
```

Siempre encierre en una sola transacción; nunca corra el UPDATE sin el INSERT al journal.

Kill-switch masivo

(Sección 23 cubrió esto; repetido aquí para el capítulo de operaciones manuales.)

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

Efectivo en la siguiente solicitud entrante — medida de 756ms de latencia de re-routing en el dry-run G3.

Scripts DBA únicos

A veces una solicitud específica del socio requiere un script único (ej. "crear usuario Ramon Garcia", "arreglar saldo de Luis Merayo"). Por memoria `feedback_one_off_script_credentials.md` : mantenga credenciales en la fuente-de-verdad del

operador. Para estos scripts:

- Escriba bajo `scripts/`
- Use credenciales literales (no env vars; el flujo de operador las necesita inline)
- Commit + push para que el rastro de auditoría sobreviva
- Corra desde la máquina del operador, no de la app prod

Última revisión de la sección: 2026-05-06 — v1.0

PARTE VII — OBSERVABILIDAD

Cómo ve qué está pasando, en vivo y histórico.

---

28. Referencia a la Consola de Transacciones en Vivo

(Vea Sección 16 — la Live Console tiene su propio manual dedicado.)

Para integración con Centro de Comando: la Live Console está bookmarked en el riel de navegación izquierdo. Use el mismo JWT admin. Es la superficie primaria de observabilidad para investigación en tiempo real.

Última revisión de la sección: 2026-05-06 — v1.0

---

29. Logs

Los logs son estructurados (donde es posible) y enrutados a múltiples sinks.

Dónde van los logs

Sink	Contenidos
Railway stdout	Todos los `console.log` / `console.error` en tiempo real. Tail vía `railway logs` o el panel de Railway.
`access.log` (archivo)	Cada solicitud HTTP vía formato combinado morgan — log de acceso histórico
`viaone-alert-service`	Errores de alta severidad → bot WhatsApp ops (alertas en vivo)

Tail de logs en vivo

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

Filtros disponibles vía `--filter` (Railway CLI):

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

Throttling de logs

Para evitar límites de rate de Railway, varios subsistemas hacen throttle (ej. logs Redis solo en cambio de estado, no por intento). Si ve menos mensajes de lo esperado, esa es la razón.

Filtros de log por servicio

Servicio	Prefijo de filtro
Latcom-fix	(default)
viaone-support	`[support]`
relier-ppn	`[ppn]`
viaone-command	`[command]`

Última revisión de la sección: 2026-05-06 — v1.0

---

30. Endpoints de salud

Ya cubierto en Sección 24. Referencia rápida:

- `/health` — vivo básico
- `/health/db` — pool DB + sondeo
- `/health/providers` — medidor por proveedor

Monitoreo externo (Uptime Robot, etc.) pega `/health` cada minuto.

Última revisión de la sección: 2026-05-06 — v1.0

PARTE VIII — OPERACIONES DE DESPLIEGUE Y LANZAMIENTO

Cómo el código va de la laptop de un desarrollador a producción. Crítico para cualquiera que toque deploys.

---

31. El flujo de lanzamiento

Regla staging-first

Por memoria `feedback_staging_first.md`: **NUNCA despliegue cambios mayores a producción directamente.** El break del Chinese Wall del 1 de abril de 2026 fue causado por un cambio de routing Telefónica empujado directo a prod sin soak en staging. No repita.

Qué cuenta como "mayor":

- Middleware que corre en cada solicitud (ej. el request-logger de la Live Console)
- Cambios de capa de routing (route_mappings, optimusRouter, tenant-context)
- Cambios de auth (adminAuth, ipWhitelist)
- Nuevos proveedores o cambios de forma de llamada a proveedor
- Migraciones de schema que cambien tablas existentes

Qué es seguro-directo-a-prod (debatible):

- Nuevas rutas aisladas (no tocan flujo existente)
- Cambios solo-UI
- Documentación
- Rollouts gated por feature-flag donde la bandera default off

Cuando dude: staging primero.

Jerarquía Tier 1 / 2 / 3 / 4 / 5 (de la lista de Kyle)

Tier	Significado	Ejemplo
Tier 1	Bloqueador duro — no voltee primer distribuidor sin esto	Fixes de código para path de lanzamiento
Tier 2	Debe ser cierto lunes AM	Verificación, runbooks, env vars
Tier 3	Pre-requisitos lanzamiento martes	Altán + Bait/Walmart go-live
Tier 4	Post-lanzamiento semana 1	Limpieza, F-items, DR launch-week
Tier 5	Después (P2/P3)	Limpieza de sidebar, EV V1

La lista completa para el lanzamiento Telefónica de mayo 2026 es `MONDAY_LAUNCH_CHECKLIST.md` — manténgala como el doc canónico para ese lanzamiento.

Lista pre-flight (genérica)

Antes de cualquier deploy a producción:

- [ ] Código revisado (PR aprobado)
- [ ] Deploy a staging exitoso
- [ ] Soak de staging ≥ 15 min (más para cambios de comportamiento)
- [ ] Migración aplicada a staging — verificada vía `\d no solo fila pgmigrations`
- [ ] Pruebas smoke verdes en staging
- [ ] Runbook operacional actualizado para nuevos modos de falla
- [ ] On-call notificado de ventana de deploy
- [ ] Plan de rollback documentado (commit SHA al cual revertir, kill-switch SQL listo)

Ventanas de cutover

Deploys mayores suceden en ventanas de bajo tráfico:

- Volumen Telefónica picos 9am–9pm CDMX
- Mejor ventana deploy: 3am–7am CDMX (bajo tráfico, on-call despierto)
- Para el lanzamiento mayo 2026: 00:00 jueves 7 mayo (cero tráfico)

Árbol de decisión de rollback

Después de un deploy, observe:

- Tasa de error por minuto
- Latencia p95
- Volumen de alertas NOC

Árbol de decisión:

- Tasa de error > 5% sobre 5 min → **rollback deploy**
- Errores de un solo distribuidor > 20% → **voltee de vuelta la bandera de ese distribuidor** (Sección 33)
- LATJAVA inaccesible → **kill-switch masivo**
- Volumen de alertas NOC × 3 normal → **investigue antes de la siguiente acción**

Pasos de rollback:

- Revertir vía `git revert` + push
- O re-desplegar commit previo vía `railway up --detach` desde el worktree del commit previo
- Una vez aterrice el revert, `kill-switch` cualquier bandera de cliente en vuelo

Última revisión de la sección: 2026-05-06 — v1.0

---

32. Gestión de dominios personalizados

Cuando un socio trae su propio dominio (ej. `buprolat.latcom.co`), aquí está el procedimiento completo.

Regla CNAME-no-IP

Railway NO emite IPs estáticas de entrada. La configuración `Static Outbound IPs` (ej. `162.220.234.15`) es **solo-salida** — la UI de Railway dice explícitamente "cannot be used for inbound

traffic."

Los socios deben **CNAME** al objetivo emitido por Railway, no A-record a ninguna IP. Si un socio insiste en un A record (algunos hosts DNS antiguos), las únicas opciones son:

- Railway Pro Static Ingress IP (add-on pagado)
- Un proxy enfrente (CDN, Cloudflare, etc.) que les dé un A estático — ellos CNAME al proxy, proxy CNAME a nosotros

Agregar subdominio de socio paso a paso

[FIGURA 35: Wizard de configuración de dominio personalizado]

1. **Obtenga el hostname del socio** (ej. `buprolat.latcom.co`).
2. **Agregue como dominio personalizado de Railway** vía GraphQL (CLI tiene el bug del prefijo TXT por `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": "<id env prod>",
        "projectId": "<project id>",
        "serviceId": "<latcom-fix service id>"
      }
    }
  }'
```

1. **Railway devuelve:**
   - Un objetivo CNAME (ej. `61vobyc4.up.railway.app`)
   - Un host + valor de registro de verificación TXT
1. **El socio agrega dos registros DNS** en su proveedor DNS:

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

1. **Espere propagación DNS** (~5 min para la mayoría de proveedores; hasta 1 hora para lentos).
1. **Railway emite cert SSL** automáticamente una vez ve ambos registros. Estado del cert: `VALIDATING_OWNERSHIP` → `ISSUING` → `VALID`.
1. **UPDATE la fila del tenant:**

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

1. **Pruebe con curl:**

``bash curl -s -o /dev/null -w "HTTP=%{http_code} ssl_ok=%{ssl_verify_result}\n" \ https://buprolat.latcom.co/api/dislogin # Esperado: HTTP=405 (método no permitido para GET en ruta solo-POST) o HTTP=200, ssl_ok=0 (éxito) ``

1. **Documente en el `settings` de la fila del tenant** qué hostname mapea a dónde, para futuros operadores.

Gotcha de verificación TXT

Por memoria `feedback_railway_cli_txt_bug.md`: el comando CLI `railway domain` **imprime la verificación TXT con un prefijo doble `railway-verify=`**. NO agregue otro prefijo al valor TXT. El campo GraphQL `verificationToken` da el valor correctamente formado — péguelo verbatim.

Expectativas de propagación DNS

- **Mismo proveedor DNS que sus registros principales** (más común): 1–5 min
- **Proveedor DNS diferente para el subdominio**: 5–30 min
- **Proveedores lentos (algunos legacy)**: hasta 1 hora

Si después de 1 hora el cert todavía no se ha emitido y el DNS está verificablemente propagado en todas partes externas (`dig` desde múltiples resolvers), el verificador de Railway puede estar flapping. Abra un ticket con Railway support — a veces necesitan empujar manualmente.

Múltiples dominios por tenant

Actualmente: una fila en `tenants` puede contener un `custom_domain`. Si un tenant necesita 3 dominios todos apuntando al mismo backend:

- **Opción A (v1):** Agregue 3 filas separadas de tenant (diferente tenant_code cada una, pero misma config downstream). Confuso pero funciona.
- **Opción B (planeado):** Extender a una tabla `tenant_aliases`. Aún no implementado.

Para la mayoría de casos, un dominio por tenant es suficiente.

Última revisión de la sección: 2026-05-06 — v1.0

---

33. Manual del kill-switch

La forma más rápida de revertir un deploy que se está portando mal en el wild.

Kill-switch por cliente

```
-- Rollback UN distribuidor:
UPDATE customers
SET payments_iso8583_enabled = FALSE
WHERE customer_id = 'latcomdigital'
RETURNING customer_id, short_name;
```

Efectivo en la siguiente solicitud entrante después del commit (auth-jwt re-fetches el cliente por solicitud — sin caché JWT/Redis para flush).

Kill-switch masivo

```
-- Rollback TODOS los distribuidores volteados:
UPDATE customers
SET payments_iso8583_enabled = FALSE
WHERE payments_iso8583_enabled = TRUE
RETURNING customer_id, short_name;
```

Tiempo de invalidación de caché

El dry-run del kill-switch G3 (abril 2026) midió 756ms desde el commit del UPDATE SQL hasta la siguiente solicitud siendo enrutada vía el nuevo path. Entonces:

- T+0: commit SQL
- T+~750ms: siguiente solicitud entrante aterriza en path VIEJO

Si necesita más rápido: reinicio API. El caché del cliente es in-process; reinicio lo limpia instantáneamente. Pero reinicio tiene su propio costo (transacciones en vuelo durante reinicio se abortan).

Matriz de decisión kill-switch

Escenario	Acción
Un solo distribuidor mostrando errores altos	Kill-switch por cliente en ese distribuidor
Múltiples distribuidores en mismo tenant fallando	Kill-switch masivo para ese tenant (filtre por tenant_id en el SQL)
Todos los clientes a través de tenants fallando	Kill-switch masivo global
Proxy LATJAVA mismo muerto	Kill-switch masivo + reinicio LATJAVA
Mal código en producción	Revertir el deploy Y kill-switch masivo (defensa en profundidad)

Árbol de decisión de rollback (repetir de Sección 31)

1. Tasa de error > 5% sobre 5 min → rollback deploy
2. Errores de un solo distribuidor > 20% → voltee de vuelta solo ese distribuidor
3. LATJAVA inaccesible → kill-switch masivo
4. Issues de auth/firewall del proveedor → contacte proveedor, sin rollback de código necesario

Última revisión de la sección: 2026-05-06 — v1.0

PARTE IX — REFERENCIA DE INTEGRACIONES

Un directorio de cada integración activa, en caso de que el lector necesite saber "¿dónde vive X?".

---

34. Directorio de proveedores

Topup / paquete México

Proveedor	Propósito	Endpoint	Auth	Notas
Telefónica MX (TEMM)	ISO8583 pagos alto volumen	`10.225.236.72:7903` (prod), `10.225.244.79:7903` (cert) vía proxy LATJAVA	Cert cliente (cert env), peer-allowlist (prod)	El objetivo del lanzamiento mayo 2026
Altamira	SOAP recargas + paquetes (Movistar IVA)	proxy en `10.225.236.116:9303`	App ID + user code	IVA-ajustado: envía `ceil(amount/1.16)`
Codigo Arix	Paquetes MX	`https://api.codigoarix.com/arix/api`	ID cliente + usuario + clave	Telcel/AT&T enrutan aquí
MUWE	Pago de servicios + SPEI + topup	`https://test.sipelatam.mx` (sandbox)	Firma MD5	OXXO/Enefevo cash vía webhook
Taecel	Pagos de servicios	`${TAECEL_BASE_URL}`	API key	
Altán	MVNO multi-marca	`${ALTAN_BASE_URL}`	API key auto-cargada de tabla `altan_devapps`	BE 347 + otros

Topup internacional

Proveedor	Propósito	Endpoint	Auth
PPN / ValueTopup	100+ países	`${PPN_BASE_URL}`	HTTP Basic + lista blanca IP
CellPay	Topups + gift cards US	`${CELLPAY_BASE_URL}`	API key
Reloadly	50+ países (respaldo)	`https://api.reloadly.com`	API key
NIHN	Page Plus + MobileX (US)	vía proxy PPN	API key
Servipagos VZ	Pagos de servicios Venezuela	`${SERVIPAGOS_BASE_URL}`	Merchant ID + API token

Tarjeta / checkout hospedado

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

Blockchain / monederos

Proveedor	Propósito	Endpoint	Auth
Privy	KMS para llaves de monedero	`https://api.privy.io`	App ID + app secret
Movement	Cadena RISE primaria	`https://mainnet.movementnetwork.xyz/v1`	Ninguno (RPC público)
Aptos	Cadena respaldo	`https://fullnode.mainnet.aptoslabs.com/v1`	Ninguno
Tron	USDT (TRC-20)	`https://api.tronstack.com`	Ninguno
Polygon	USDC legacy	Alchemy / Quicknode	API key

Mensajería

Proveedor	Propósito
WhatsApp Cloud API (Meta)	Canal cara-a-socio primario
Twilio	SMS, voz, WA legacy
uContact	SMS alterno
Telegram	Canal secundario vía grammY
Vonage	Búsqueda de número de portador US
Tlaloc	Detección de portador interna

Cumplimiento / datos / IA

Proveedor	Propósito
treasury.gov OFAC	Fuente de lista de sanciones (sync diario 04:30 UTC)
ExchangeRate-API	FX USD/MXN (refresh 9am diario)
Anthropic Claude	NLP (clasificación de intent, análisis NOC)
SendGrid	Email transaccional

Almacenamiento

Proveedor	Propósito
AWS S3	Documentos KYC, reportes de conciliación
DigitalOcean Spaces	Backups encriptados diarios PG + LATJAVA

Última revisión de la sección: 2026-05-06 — v1.0

---

35. Directorio de tenants

Tenant	Descripción	Plan	Detalles distintivos
OPTIMUS	Latcom Horizons II — distribución Telefónica MX	professional	Mayor volumen, lanzamiento mayo 2026, dominio personalizado `buprolat.latcom.co`
HAZ_GROUP	HAZ Communications — multi-LATAM	professional	Multi-país (clientes HAZ_GROUP tienen allowed_countries en cada fila)
RISE FINANCE	RISE Holdings — remesas DeFi	professional	Cadena Movement primaria, Aptos respaldo, no custodial
VIAONE	Productos Via One directos al consumidor	professional	El Vecino, EV bot, Via One web app
VIAONE_MASTER	Tenant admin entre tenants	n/a	El rol system_admin vive aquí

Última revisión de la sección: 2026-05-06 — v1.0

---

36. Directorio de clientes (alto nivel)

Distribuidores activos por tenant con su forma de integración:

OPTIMUS

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

- `LATCONECTA_001` — Latconecta Digital (con IP whitelist)
- (12 otros — vea `SELECT customer_id, company_name FROM customers WHERE tenant_id = (SELECT id FROM tenants WHERE tenant_code = 'OPTIMUS')` para lista en vivo)

HAZ_GROUP

- `HAZ_001` — HAZ Group distribuidor principal

RISE FINANCE

- `RISE_HOLDINGS_001` — RISE Holdings

VIAONE

- `013141` — El Vecino
- (otros)

VIAONE_MASTER

- registros de usuarios system_admin

Última revisión de la sección: 2026-05-06 — v1.0

PARTE X — APÉNDICES

Apéndice A — Hoja de referencia de endpoints

Públicos (sin auth)

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

API Distribuidor

```
POST /api/distributor/login          (o /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        (o /api/tn)            JWT
POST /api/distributor/tn/latcom      (o /api/tn/latcom)     JWT
GET  /api/distributor/get_balance                            JWT
GET  /api/distributor/product_list                             JWT
```

API Cliente (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 (que recibimos)

```
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
```

Apéndice B — Diccionario de códigos de respuesta F39

(Códigos de respuesta ISO8583 como Telefónica TEMM los usa.)

F39	Significado	Acción
00	Aprobado	Éxito
01	Referirse a emisor	Investigar; issue del lado del socio
05	No honrar	Decline genérico; verifique elegibilidad de destino
12	Transacción inválida	Solicitud malformada; verifique campos
13	Monto inválido	Monto fuera de rango
14	Tarjeta / cuenta inválida	Destino equivocado
30	Error de formato	Wire format malformado
41	Tarjeta perdida	n/a para nuestro caso de uso
43	Tarjeta robada	n/a
51	**Fondos insuficientes**	**Cert env: quota agotada; Prod: issue de saldo del cliente**
54	Expirada	n/a
55	PIN inválido	Falla auth
57	Transacción no permitida al emisor	Carrier no acepta esta op
58	Transacción no permitida al terminal	Issue routing o config terminal
61	Excede límite de retiro	Monto muy alto
62	Tarjeta restringida	Bloqueo del lado del carrier
63	Violación de seguridad	Issue de seguridad (raro)
65	Excede frecuencia de retiro	Límite de velocidad
68	Respuesta recibida muy tarde	Timeout — proveedor puede haber procesado; concilie
75	Entradas PIN inválidas	Auth
76	Producto inválido	SKU no permitido
82	Time-out en emisor	Timeout proveedor
83	**No se puede completar, violación de ley**	TEMM cert: teléfono no provisionado en cert env
87	Solo compra — sin cash advance	Restricción del lado del carrier
91	Emisor o switch inoperativo	Proveedor caído
92	Institución financiera no encontrada	Error de routing
94	Transmisión duplicada	Hit de idempotencia
96	Mal funcionamiento del sistema	Error del proveedor

Apéndice C — Convención de nombres de SKU

Conocer el prefijo le dice mucho:

Prefijo	Significado	Ejemplo
`TFE_`	Topup Telefónica MX (estilo TAE, MXN open-range)	`TFE_150_MXN`, `TFE_MXN_20_TO_2000`
`GLO01T_`	Topup Telefónica MX (open-range, P1=MXN-priced, D1=USD-priced)	`GLO01T_RAMXN_P1`, `GLO01T_RAUSD_D1`
`GLO01B_`	Paquete Telefónica MX (P1=MXN-priced, D1=USD-priced)	`GLO01B_6M2D_P1`, `GLO01B_6M2D_D1`
`LATCOM_PAQ`	Paquete legacy LATCOM Barcelona (mapeo hardcoded)	`LATCOM_PAQ16`, `LATCOM_PAQ17`, `LATCOM_PAQ18`
`GCA09T_`	Topup Telcel vía Codigo Arix	`GCA09T_080MXN_C1`
`GCA09B_`	Paquete Movistar vía Codigo Arix	`GCA09B_100MXN_C1`
`RMP`	Código de paquete Movistar (wire Codigo Arix)	`RMP100`, `RMP150`, `RMP200`
`PQRI`	SKU wire de paquete Telefónica MX (TEMM operator_sku_id)	`PQRI6M2DP`, `PQRI1G4DP`, `PQRIIPM`
`TEMXN_`	Telefónica MX (nomenclatura más nueva)	`TEMXN_RECARGA_30_TEMM`, `TEMXN_PQRI6M2D_2_DAYS`
`TAETELCEL`	Tiempo Aire Electrónico Telcel (open-range)	`TAETELCEL`
`INT`, `PA`, `PAA`	Planes de datos Telcel	`INT`, `PA`
`MUWE_`	Biller MUWE	`MUWE_4052305028`
`SPVZ-`	Biller Servipagos VZ	`SPVZ-MOV-POS`

El carrier está codificado en el prefijo; el canal en el sufijo (`_P1` = MXN-priced point-1, `_D1` = USD-priced).

Apéndice D — Comportamiento del middleware tenant-context

El algoritmo completo está en `middleware/tenant-context.js`. Comportamientos clave:

1. **Resolución hostname → tenant:**
   - Extraer primera etiqueta del hostname (`optimus` de `optimus.via.one`)
   - Query: `SELECT * FROM tenants WHERE subdomain = $1 AND status = 'active'`
   - Fallback: `WHERE custom_domain = $1`
1. **Caché LRU de 5 minutos** (max 200 entradas) — elimina hit DB en cada solicitud.
1. **Caché negativo** — hosts desconocidos cacheados como `tenant: null` por el mismo TTL, previniendo búsquedas repetidas fallidas.

1. **Circuit breaker** — abre después de 3 fallas DB consecutivas, cooldown 30 segundos, recae a caché negativo durante outage.
1. **Variable de sesión PostgreSQL** — una vez resuelto un tenant, el middleware establece `app.current_tenant = $tenant_id` en la conexión. Las políticas RLS en transactions / customers / route_mappings filtran por esta variable.

Consultas de diagnóstico

Si la resolución de tenant parece off, intente:

```
-- ¿Qué ve el resolver para este hostname?
SELECT id, tenant_code, subdomain, custom_domain, status
FROM tenants
WHERE subdomain = 'optimus' OR custom_domain = 'optimus.via.one';

-- ¿Está RLS activo para este tenant?
SET app.current_tenant = '<uuid-de-tenants.id>';
SELECT COUNT(*) FROM customers; -- debería coincidir con esperado para ese tenant
RESET app.current_tenant;
```

Apéndice E — Variables de entorno críticas

Var	Default	Propósito
`JWT_SECRET`	(requerido)	Secret HS256 para JWTs admin/distribuidor
`DATABASE_URL`	(requerido)	String de conexión Postgres (Railway interno)
`DATABASE_PUBLIC_URL`	(auto)	URL pública Postgres (para psql desde fuera)
`REDIS_URL`	(requerido para v2 webhooks + hidratación circuit breaker)	Conexión Redis
`NODE_ENV`	`production` (¡debe!)	Cuando `staging`, los proveedores default a modo mock
`PAYMENTS_CERT_MODE`	`false`	Cuando `true`, intentos de paquete se reescriben a par cert (code=21, amount=150)
`PAYMENTS_PROXY_URL`	`https://66.231.242.91`	URL del proxy LATJAVA
`PAYMENTS_GATEWAY_API_KEY`	(requerido para prod)	Key auth del gateway
`PAYMENTS_TIMEOUT_MS`	45000	Timeout Telefónica — considere 30000 en prod para reversión más rápida
`PAYMENTS_IVA_ADJUST`	`false`	Habilitar IVA targeting
`PAYMENTS_CERT_PAQUETE_CODE`	`21`	Código de paquete cert override
`PAYMENTS_CERT_PAQUETE_AMOUNT`	`150`	Monto cert override
`TELEFONICA_GATEWAY_URL`	`https://66.231.242.91`	Igual que PAYMENTS_PROXY_URL
`PG_POOL_MAX`	30	Conexiones max por pool (cada servicio tiene su propio pool)
`FORCE_REAL_PROVIDERS`	unset	Override solo-staging para llamar proveedores reales
`DR_MODE`	unset	Cuando set + leído por código (tarea DR-1), salta crons transaccionales en réplica DR
`RAILWAY_TOKEN`	(set en shell)	Token para Railway CLI en modo no-interactivo
`OFAC_API_KEY`	sin uso	Legacy — auto-hospedamos ahora
`PASSWORD_RESET_EMAILS_ENABLED`	`false`	Set `true` una vez vars SMTP estén configuradas

Apéndice F — Directorio de tablas de base de datos

Por dominio. Una línea de propósito cada una.

Identidad y tenancy

- `tenants` — partición de nivel superior
- `customers` — distribuidores / consumidores API dentro de un tenant
- `users` — logins UI/dashboard (separados de customers)
- `api_keys` — tokens auth por cliente

Transacciones y ledger

- `transactions` — ledger primario de transacciones (vista cara-a-cliente)
- `provider_transactions` — cada llamada a proveedor que hemos hecho (con payloads completos)
- `operator_balance` — saldo retenido por proveedor
- `balance_journal` — rastro auditoría para créditos/débitos manuales
- `idempotency_keys` — dedup para reintentos

Monederos y blockchain

- `wallet_addresses` — direcciones crypto por usuario, por cadena
- `wallet_transfers` — transacciones de cadena
- `ev_accounts` — registros de monedero consumidor El Vecino

Cumplimiento

- `ofac_entries`, `ofac_alt_names`, `ofac_addresses`, `ofac_ids` — corpus de nombres sancionados
- `ofac_screening_log` — auditoría de cada evento de filtrado
- `ofac_sync_metadata` — estado de sync
- `kyc_documents` — documentos KYC subidos (referencias de claves S3)
- `kyb_applications` — formularios de admisión de socios
- `audit_log` — auditoría genérica de acciones admin

Catálogo

- `latcom_products` — catálogo SKU canónico
- `operator_products` — mapeo SKU por proveedor
- `route_mappings` — operador + tenant + servicio → routing de proveedor
- `vendors` — registro de credenciales de proveedor
- `altan_devapps` — credenciales específicas de BE Altán

Webhooks

- `webhook_endpoints` — URLs de callback configuradas por socio
- `webhook_deliveries` — cola durable de entrega con estado + historial de reintentos

Sesiones y conversaciones

- `sessions` — sesiones UI
- `conversations` — hilos de conversación bot
- `siri_sessions` — estado integración Siri
- `auth_logs` — intentos de login (éxito + falla)

Bots / mensajería

- `whatsapp_messages` — log inbound + outbound WA
- `rcs_users`, `channel_identities` — mapeo de identidad multi-canal

Console (este manual)

- `api_requests` — log de hits API entrantes (potencia la Live Console)

Otros

- `dr_sync_log` — estado de sync de réplica DR
- `operators` — registro de portadores telecom
- `feature_flags` (tabla; también vive en tenants.feature_flags JSONB) — gates globales de funciones

---

Apéndice G — Consultas SQL comunes

"Últimas 24h transacciones para 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;
```

"Clientes con payments_iso8583 volteado"

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

"Coincidencias OFAC recientes"

```
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;
```

"Muertes de webhooks última semana"

```
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;
```

"Tasa de éxito de proveedor última 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;
```

"Saldo de cliente debajo de threshold de alerta"

```
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;
```

Apéndice H — Historial del documento

Versión	Fecha	Notas
1.0	2026-05-06	Lanzamiento inicial — 10 partes + 8 apéndices, edición en español. Edición en inglés publicada como PDF separado. Las 10 partes están completamente traducidas al español con la misma profundidad que la edición en inglés.

Fin del manual.

Para la Consola de Transacciones en Vivo, vea *Via One — Live Transaction Console (User Manual).pdf*.

Para documentos de integración específicos de socios, vea *Existing Partner Manuals/* en esta carpeta.

Para especificaciones del protocolo wire de Telefónica, vea *Reference Specs/* en esta carpeta.

Actualizaciones y correcciones: por favor contribuya vía pull request al archivo fuente markdown — el PDF se regenera desde ahí.