

RecargameYa — API Pública v1

Documentación de integración para clientes B2B.

Base URL: <https://recargameya.com/api/v1/public>

Modos: [sandbox](#) (pruebas sin ejecución real) y [live](#) (operaciones reales), identificados por el prefijo de la API Key.

1. Autenticación

Cada request debe incluir 3 headers:

Header	Descripción
X-API-Key	Public key: pk_live_xxx o pk_sandbox_xxx
X-API-Timestamp	Unix epoch en segundos (± 300 s del reloj del servidor)
X-API-Signature	HMAC-SHA256 (hex minúsculas) del signing string usando tu secret

Signing string

```
<timestamp>\n<HTTP_METHOD>\n<request_path>\n<sha256(body_bytes)>
```

- [request_path](#) : path completo sin querystring, sin host (ej: [/api/v1/public/recharge](#))
- [body_bytes](#) : cuerpo raw del request. Para GET → string vacío (hash: [e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855](#))
- [\n](#) : LF (byte [0x0A](#)), nunca [CRLF](#)
- Separador entre los 4 componentes: un solo [\n](#)

Ejemplo cURL (bash)

```
API=https://recargameya.com
KEY="pk_sandbox_xxxxx"
SECRET="sk_sandbox_xxxxx"

TS=$(date +%s)
METHOD="POST"
PATH="/api/v1/public/recharge"
BODY='{"client_reference_id":"MY-REF-001","carrier":"claro","phone":"8091234567","amount":100}'
BODY_HASH=$(printf '%s' "$BODY" | sha256sum | awk '{print $1}')
SIGNING=$(printf "%s\n%s\n%s\n%s" "$TS" "$METHOD" "$PATH" "$BODY_HASH")
SIG=$(printf '%s' "$SIGNING" | openssl dgst -sha256 -hmac "$SECRET" | awk '{print $2}')

curl -X POST "$API$PATH" \
  -H "X-API-Key: $KEY" \
  -H "X-API-Timestamp: $TS" \
  -H "X-API-Signature: $SIG" \
  -H "Content-Type: application/json" \
  -d "$BODY"
```

Ejemplo Python

```
import hashlib, hmac, json, time, requests

KEY = "pk_sandbox_XXXXX"
SECRET = "sk_sandbox_XXXXX"
BASE = "https://recargameya.com"

def signed_request(method, path, body=None):
    body_bytes = json.dumps(body).encode() if body else b""
    ts = str(int(time.time()))
    body_hash = hashlib.sha256(body_bytes).hexdigest()
    signing = f"{ts}\n{method}\n{path}\n{body_hash}"
    sig = hmac.new(SECRET.encode(), signing.encode(), hashlib.sha256).hexdigest()
    headers = {
        "X-API-Key": KEY,
        "X-API-Timestamp": ts,
        "X-API-Signature": sig,
        "Content-Type": "application/json",
    }
    return requests.request(method, BASE + path, headers=headers, data=body_bytes)

r = signed_request("POST", "/api/v1/public/recharge", {
    "client_reference_id": "MY-REF-001",
    "carrier": "claro",
    "phone": "8091234567",
    "amount": 100,
})
print(r.json())
```

Ejemplo Node.js

```
const crypto = require("crypto");

const KEY = "pk_sandbox_xxxxx";
const SECRET = "sk_sandbox_xxxxx";
const BASE = "https://recargameya.com";

async function signedRequest(method, path, body = null) {
  const bodyStr = body ? JSON.stringify(body) : "";
  const bodyHash = crypto.createHash("sha256").update(bodyStr).digest("hex");
  const ts = Math.floor(Date.now() / 1000).toString();
  const signing = `${ts}\n${method}\n${path}\n${bodyHash}`;
  const sig = crypto.createHmac("sha256", SECRET).update(signing).digest("hex");

  return fetch(BASE + path, {
    method,
    headers: {
      "X-API-Key": KEY,
      "X-API-Timestamp": ts,
      "X-API-Signature": sig,
      "Content-Type": "application/json",
    },
    body: method === "GET" ? undefined : bodyStr,
  });
}
```

Diagnóstico sin firma — GET /health

```
curl https://recargameya.com/api/v1/public/health
```

Útil para verificar conectividad y sincronizar reloj. Compara `data.server_time` con tu Unix epoch local. Si el drift es >300s, las requests firmadas serán rechazadas.

Troubleshooting HTTP 401

Mensaje	Causa	Solución
Falta el header X-API-Key	Header ausente	Añade X-API-Key
API Key inválida	Mal escrita o deshabilitada	Verifica prefijo completo pk_sandbox_ o pk_live_
X-API-Timestamp inválido	No numérico	Usa Unix epoch en segundos
X-API-Timestamp fuera de rango. Drift=Xs	Reloj desincronizado	Sincroniza NTP; mensaje indica drift
Firma inválida	Signing string incorrecto	Revisa path completo con /api/v1/public/ , \n como LF, hex minúsculas, secret correcto
IP no autorizada	Whitelist activa	Solicitar añadir IP a soporte

2. Sandbox vs Live

Cada cliente API recibe 2 pares de keys:

Modo	Prefijo	Comportamiento
sandbox	pk_sandbox_* / sk_sandbox_*	Simula respuestas (~90-95% éxito). Balance sandbox separado. No ejecuta operaciones reales.
live	pk_live_* / sk_live_*	Ejecuta operaciones reales. Descuenta del balance prepago live.

Ambos modos comparten los mismos endpoints — solo cambia la key.

3. Idempotencia

Todo POST requiere client_reference_id en el body:

- String único generado por el cliente (UUID recomendado), máx 64 chars.
- Si repites el request con el mismo client_reference_id, retornamos la misma respuesta de la primera vez con idempotent_replay: true. No re-ejecutamos la operación.
- Esencial para retries ante timeouts de red: garantiza no cobrar dos veces.

4. Rate Limiting

- Default: 60 requests/minuto por API key.
 - Al superar: `HTTP 429` con `code: rate_limit_exceeded`.
 - Contactar soporte para ampliar.
-

5. Endpoints Fase 1 — Recargas y Facturas

5.1 `GET /balance` — consultar balance

Response 200:

```
{
  "ok": true,
  "data": {
    "user_id": "00000XXX",
    "business_name": "Nombre del cliente",
    "mode": "live",
    "balance": 15420.50,
    "currency": "DOP"
  }
}
```

5.2 `GET /balance/history?limit=50` — audit trail de movimientos

Devuelve el historial de cambios de balance. Útil para cuadrar contabilidad.

⚠ La query string **NO** se firma. Solo firma el path `/api/v1/public/balance/history`.

Response 200:

```
{
  "ok": true,
  "data": {
    "user_id": "00000XXX",
    "mode": "live",
    "current_balance": 15420.50,
    "total": 2,
    "items": [
      {
        "id": "uuid-...",
        "delta": -50.00,
        "balance_before": 15470.50,
        "balance_after": 15420.50,
        "cause_type": "api_operation",
        "cause_id": "TXN-API-...",
        "note": "recarga_movil claro ref=MI-REF-001",
        "created_at": "2026-10-03T06:16:58.689Z"
      }
    ]
  }
}
```

Valores posibles de `cause_type` :

Cause	Significado
<code>api_operation</code>	Deducción por operación ejecutada
<code>api_refund</code>	Reversión por fallo del operador o excepción interna
<code>admin_adjust</code>	Ajuste manual desde el panel admin
<code>admin_reset</code>	Reset del balance por admin (excepcional)
<code>initial_credit</code>	Crédito inicial de sandbox

5.3 GET /catalog/carriers — carriers móviles disponibles

```
{
  "ok": true,
  "data": {
    "carriers": [
      {"code": "claro", "name": "Claro", "country": "DO", "min_amount": 5, "max_amount": 5000},
      {"code": "altice", "name": "Altice", "country": "DO", "min_amount": 5, "max_amount": 5000},
      {"code": "digicel", "name": "Digicel", "country": "DO/HT", "min_amount": 5, "max_amount": 5000},
      {"code": "natcom", "name": "Natcom", "country": "HT", "min_amount": 5, "max_amount": 5000},
      {"code": "viva", "name": "Viva", "country": "DO", "min_amount": 5, "max_amount": 5000}
    ]
  }
}
```

5.4 GET /catalog/data-packages?carrier=claro — paquetes de datos

Devuelve los paquetes de internet disponibles con sus `package_code` para usar en `POST /data-package`.

5.5 POST /recharge — recarga móvil

Body:

```
{
  "client_reference_id": "REF-000001",
  "carrier": "claro",
  "phone": "8091234567",
  "amount": 100
}
```

Response 200:

```
{
  "ok": true,
  "data": {
    "id": "TXN-API-XXXXXXXXXX",
    "client_reference_id": "REF-000001",
    "type": "recarga_movil",
    "provider": "claro",
    "phone": "8091234567",
    "amount": 100,
    "amount_charged": 94,
    "commission": 6,
    "currency": "DOP",
    "status": "completado",
    "external_reference": "RM-XXXX",
    "provider_response_message": "OK",
    "created_at": "2026-10-03T19:30:54Z",
    "updated_at": "2026-10-03T19:30:55Z",
    "mode": "live"
  }
}
```

Response 402:

```
{"detail": {"code": "insufficient_balance", "message": "Balance insuficiente. Requerido: RD$94.00"}}
```

5.6 POST /data-package — paquete de internet

Body:

```
{
  "client_reference_id": "REF-000002",
  "carrier": "claro",
  "phone": "8091234567",
  "package_code": "CLARO_DATA_1GB_DAY"
}
```

Shape de response idéntico a `/recharge` pero con `type: "paquete_internet"`.

5.7 POST /invoice/query — consultar facturas pendientes

Body:

```
{"provider": "edenorte", "account_number": "4599616"}
```

Response 200:

```
{
  "ok": true,
  "data": {
    "provider": "edenorte",
    "account_number": "4599616",
    "client_name": "JUAN PEREZ",
    "invoices": [
      {"invoice_id": "INV123", "period": "2026-01", "amount": 850.50, "due_date": "2026-02-15"}
    ],
    "total_to_pay": 850.50
  }
}
```

Providers soportados: edenorte, edesur, edeeste, caasd, coraasan, claro_hogar, altice_hogar.

5.8 POST /invoice/pay — pagar factura(s)

Paga una o varias facturas del mismo `provider` + `account_number`. Si envías N facturas, se hace una sola llamada al operador por el total consolidado. En base de datos se crean N transacciones locales con un `group_id` común.

Body:

```
{
  "client_reference_id": "REF-000003",
  "provider": "edenorte",
  "account_number": "4599616",
  "invoices": [
    {"invoice_id": "INV123", "amount": 850.50},
    {"invoice_id": "INV124", "amount": 340.20}
  ]
}
```

- `invoices[].amount` debe coincidir con el monto retornado en `/invoice/query`.
- Máximo 20 facturas por request. No se permiten `invoice_id` duplicados.

Response 200 (consolidado):

```
{
  "ok": true,
  "data": {
    "id": "TXN-API-XXXXXXXXXX",
    "client_reference_id": "REF-000003",
    "type": "factura",
    "provider": "edenorte",
    "phone": "4599616",
    "amount": 850.50,
    "status": "completado",
    "group_id": "GRP-API-XXXXXXX",
    "total_amount": 1190.70,
    "total_charged": 1190.70,
    "invoices": [
      {"invoice_id": "INV123", "amount": 850.50, "status": "completado", "transaction_id": "TXN-API-..."},
      {"invoice_id": "INV124", "amount": 340.20, "status": "completado", "transaction_id": "TXN-API-..."}
    ]
  }
}
```

Estados posibles finales:

- `completado` — operador confirmó
- `fallido` — rechazo seguro + refund automático
- `pendiente_revision` — timeout ambiguo. Balance NO refundado automáticamente, contactar soporte

5.9 `GET /transaction/{client_reference_id}` — buscar TX por ID del cliente

5.10 `GET /transactions?limit=50&skip=0&status_filter=completado` — listar transacciones

6. Endpoints Fase 2 — Luz Prepago y Remesas

6.1 GET /light/prepaid/providers — proveedores de luz prepago

```
{
  "ok": true,
  "data": {
    "providers": [
      {"code": "cepm_prepago", "name": "CEPM Prepago", "min_amount": 50, "max_amount": 10000},
      {"code": "edesur_prepago", "name": "Edesur Prepago", "min_amount": 50, "max_amount": 10000},
      {"code": "ceb_prepago", "name": "CEB Prepago", "min_amount": 50, "max_amount": 10000}
    ]
  }
}
```

6.2 GET /light/prepaid/account?provider=X&contract=Y — consultar titular

⚠ La query string NO se firma. Firma el path `/api/v1/public/light/prepaid/account`.

Response 200:

```
{
  "ok": true,
  "data": {
    "provider": "cepm_prepago",
    "contract": "D035010514",
    "account_holder": "JUAN PEREZ",
    "address": "CALLE PRINCIPAL #45, PUNTA CANA",
    "min_amount": 50,
    "max_amount": 10000,
    "is_simulation": false
  }
}
```

Degradación grácil: Si el operador no responde el titular, devolvemos los datos disponibles. El cliente API puede continuar con `/light/prepaid/charge` validando con el usuario final.

6.3 POST /light/prepaid/charge — recarga de luz prepago

Body:

```
{
  "client_reference_id": "REF-LUZ-000001",
  "provider": "cepm_prepago",
  "contract": "D035010514",
  "amount": 500
}
```

Response 200:

```
{
  "ok": true,
  "data": {
    "id": "TXN-API-XXXXXXXXXXXX",
    "client_reference_id": "REF-LUZ-000001",
    "type": "luz_prepago",
    "provider": "cepm_prepago",
    "contract": "D035010514",
    "amount": 500,
    "amount_charged": 492.50,
    "commission": 7.50,
    "currency": "DOP",
    "status": "completado",
    "pin": "0000-0000-0000-0000-0000",
    "token": "ABC123XYZ456",
    "kwh": 45.5,
    "external_reference": "...",
    "created_at": "2026-10-03T19:30:00Z",
    "mode": "live"
  }
}
```

Campos `pin`, `token`, `kwh` vienen cuando el operador los emite — mostrarlos al cliente final en el recibo.

6.4 GET /remittance/rate?wallet_type=moncash — tasa y comisiones vigentes

Devuelve tasa de cambio, comisión del servicio y límites. Útil para calcular en frontend antes de enviar.

`wallet_type` : `moncash` | `natcash`

Response 200:

```
{
  "ok": true,
  "data": {
    "wallet_type": "moncash",
    "exchange_rate": 2.05,
    "service_fee_pct": 10.0,
    "service_fee_min": 50.0,
    "service_fee_cap": 500.0,
    "rapido_fee_pct": 1.41,
    "min_amount_dop": 115,
    "max_amount_dop": 50000,
    "commission_modes": ["aparte", "descontada"],
    "speed_options": ["normal", "fast"],
    "currency_sent": "DOP",
    "currency_received": "HTG"
  }
}
```

6.5 POST /remittance/send — enviar remesa a Haití (MonCash/NatCash)

Body:

```
{
  "client_reference_id": "REF-REM-000001",
  "wallet_type": "moncash",
  "amount_dop": 1000,
  "recipient_name": "Jean Baptiste",
  "recipient_phone": "50948041356",
  "commission_mode": "aparte",
  "speed_option": "normal"
}
```

Campos obligatorios:

Campo	Valores	Notas
<code>client_reference_id</code>	string (máx 64)	Único por envío. Idempotente.
<code>wallet_type</code>	moncash natcash —	
<code>amount_dop</code>	$115 \leq x \leq 50,000$	Se trunca a enteros.
<code>recipient_name</code>	string	Si no se conoce, enviar literal "enviar sin nombre".
<code>recipient_phone</code>	6-8 dígitos	Solo dígitos, sin prefijos.
<code>commission_mode</code>	aparte descontada	SIN default — debe elegirse por envío.

Campos opcionales:

Campo	Valores	Default
-------	---------	---------

|---|---|---|

| `speed_option` | `normal` | `fast` | `normal` |

Diferencia entre `commission_mode` :

Modo	Qué paga el cliente final (al API consumer)	Qué recibe el destinatario
<code>aparte</code>	<code>amount_dop + service_fee</code>	<code>amount_dop × exchange_rate</code> HTG (completo)
<code>descontada</code>	<code>amount_dop</code> (exactos)	<code>(amount_dop - service_fee) × exchange_rate</code> HTG

Response 200:

```
{
  "ok": true,
  "data": {
    "id": "TXN-API-XXXXXXXXXXXX",
    "client_reference_id": "REF-REM-000001",
    "type": "remesa",
    "wallet_type": "moncash",
    "recipient_name": "Jean Baptiste",
    "recipient_phone": "50948041356",
    "amount": 1000,
    "service_fee_dop": 100,
    "amount_received_htg": 2050.00,
    "currency_received": "HTG",
    "exchange_rate": 2.05,
    "commission": 50,
    "amount_charged": 1050,
    "commission_mode": "aparte",
    "speed_option": "normal",
    "status": "pendiente",
    "created_at": "2026-10-03T19:30:00Z",
    "mode": "live"
  }
}
```

Estados posibles:

- `pendiente` — Operación recibida y en proceso. El estado final llegará vía webhook cuando el envío se complete.
- `completado` — Confirmada en destino.
- `fallido` — Rechazo + refund automático al balance.

Protección contra envíos duplicados — 3 capas

Capa 1 — Idempotencia por `client_reference_id`:

Si repites el mismo `client_reference_id`, devolvemos la respuesta de la 1ra transacción con `idempotent_replay: true`.

Capa 2 — Guardia server-side (60s):

Si en los últimos 60 segundos recibimos otra remesa con el mismo `(wallet_type + recipient_phone + amount_dop)` del mismo cliente API, rechazamos con:

```
HTTP 429
{"detail": {"code": "duplicate_transaction", "message": "Remesa idéntica detectada en los últimos 60s (ref: TXN-API-XXX)"}}
```

Capa 3 — Reintentos seguros:

Si una llamada tarda >30s por timeout de red, reintentar con el mismo `client_reference_id` devuelve el estado real sin duplicar.

7. Webhooks

Cuando cambia el estado de una TX, enviamos `POST` al `webhook_url` configurado.

Headers:

```
Content-Type: application/json
X-Recargameya-Signature: sha256=<hex_hmac>
X-Recargameya-Timestamp: <epoch>
X-Recargameya-Event: transaction.updated
X-Recargameya-Webhook-Id: <uuid>
```

Verificar firma (Python):

```
import hashlib, hmac
WEBHOOK_SECRET = "whsec_xxxxx" # entregado al configurar webhook

def verify(body_bytes, signature_header):
    mac = hmac.new(WEBHOOK_SECRET.encode(), body_bytes, hashlib.sha256)
    expected = f"sha256={mac.hexdigest()}"
    return hmac.compare_digest(expected, signature_header)
```

Body:

```
{
  "event": "transaction.updated",
  "created_at": "2026-10-03T19:30:55Z",
  "transaction": { /* mismo shape que POST /recharge, /remittance/send, etc. */ }
}
```

Retry policy: si no recibimos `2xx` en 15s, reintentamos con backoff: 1min → 2min → 4min → 8min → 16min → 32min → 64min → 128min. Tras 8 intentos → `failed` (notificable).

Respuesta esperada: HTTP 200-299 (cualquier body). Devolver `2xx` rápido y procesar async idealmente.

8. Códigos de error

HTTP	code	Descripción
400	<code>invalid_carrier</code> , <code>invalid_phone</code> , <code>invalid_wallet</code> , <code>invalid_commission_mode</code> , <code>invalid_contract</code> , <code>amount_too_low</code> , <code>amount_too_high</code> , <code>recipient_name_required</code> , <code>invalid_recipient_phone</code> , <code>duplicate_invoice_id</code>	Validación del request
401	(varios)	Auth: key inválida, timestamp fuera de rango, firma incorrecta
402	<code>insufficient_balance</code>	No hay fondos suficientes
403	(varios)	Key deshabilitada, IP no autorizada
404	<code>account_not_found</code>	Contrato/cuenta no encontrada en el operador
429	<code>rate_limit_exceeded</code> , <code>duplicate_transaction</code>	Rate limit o duplicado detectado
502	<code>provider_error</code>	Error temporal del operador. Reintentar con backoff.
503	<code>service_unavailable</code>	Servicio temporalmente no disponible

Todos los errores devuelven:

```
{"detail": {"code": "xxx", "message": "..."}}
```

9. Modelo de precios y comisiones

Cada cliente API tiene un conjunto de comisiones por servicio configurado por RecargameYa (visible en el panel del cliente). Para cada transacción:

- `amount` = monto de la operación (lo que llega al destino, o el total de la factura, o la remesa enviada)
- `commission` / `user_commission` = margen que gana el cliente API (su ganancia por revender)
- `amount_charged` = lo que se descuenta del balance = `amount - commission` (o `amount + service_fee - commission` para remesas en modo `aparte`)

Para remesas con comisión aparte:

- Si el cliente API cobra el fee a su cliente final, su ganancia es `user_commission` .
- RecargameYa se queda con `service_fee - user_commission` .

10. Sandbox → Live checklist

Antes de pasar a live, verificar:

- [] Firma HMAC calculada correctamente (probado con curl del ejemplo).
- [] Manejo de idempotencia con `client_reference_id` único por operación.
- [] Endpoint de webhook devuelve 2xx en <15s.
- [] Verificación de firma de webhook implementada.
- [] Manejo de `HTTP 429` con backoff exponencial.
- [] Manejo de `HTTP 402` (balance insuficiente) con alerta operacional.
- [] Manejo de `pendiente_revision` en facturas y `pendiente` en remesas (polling + webhook).
- [] Balance live inicial recargado con RecargameYa.

11. Operación en producción

- Horario de procesamiento de remesas: 8:00 AM a 10:00 PM (hora DO, UTC-4). Fuera de este horario las remesas quedan en estado `pendiente` y se procesan al abrir el siguiente turno. El webhook se dispara cuando se completa.
- Reconciliación: Los pagos de factura tienen reconciliación asíncrona cada 60s. Si una tx queda en `procesando` , se cierra automáticamente al confirmarse en el operador.
- Mantenimiento: Avisamos con 24h de anticipación por email. Rate limits se mantienen; endpoints devuelven `503` solo si hay corte completo.

12. Soporte

- Email técnico: soporte@recargameya.com — respuesta <24h hábiles.
- Incidentes urgentes: WhatsApp del contacto comercial asignado.
- Status page: pedir al contacto comercial al momento de la integración.