Product Specification — Payouts API v2 (Full Technical Design)

Doc type: Product Spec • **Team:** Payments • **Owner:** L. Reyes (PM) • **Authors:** A. Patel (BE), D. Chen (SRE), I. Moon (Docs)

Status: In Review → Final • **Last updated:** 2024-10-03

0. Executive Summary

Payouts API v2 provides a reliable, auditable mechanism for creating one-off and batch payouts to external bank accounts and cards across multiple providers. The v1 surface suffered from inconsistent callback contracts, limited idempotency guarantees, and ambiguous error taxonomy. The v2 release introduces formal idempotency, deterministic callbacks with HMAC signatures, and an extensible provider abstraction.

Goals

- Reduce payout creation errors by 30% (rolling 28d)

- Bring callback delivery success to ≥ 99.9% (w/ retries)

- Achieve p50 creation latency ≤ 300ms; p99 ≤ 1200ms

- Complete parity with v1 and add batch endpoint up to 1,000 items

Non-Goals

- Reconcile/settlement reporting (out of scope; separate Ledger project)

- In-app disbursement UX (separate workstream)

1. Problem & Background

- Callback inconsistency (v1): three formats, two signature variants, and missing replay protection. Merchants wrote brittle handlers.

- Idempotency gaps: duplicate payouts on client retries during provider timeouts; v1 idempotency window was 5 minutes and not enforced across regions.

- Provider coupling: business logic leaked into provider shims; difficult to route around outages or to burst to a secondary.

- Observability: lack of end-to-end trace; partial message logging; ad-hoc dashboards.

2. Proposed Solution Overview

2.1 API Surface

- POST /api/v2/payouts — single payout creation

- POST /api/v2/payouts/batch — batch creation (≤ 1000 items)

- GET /api/v2/payouts/{payout_id} — read state

- GET /api/v2/payouts/batch/{batch_id} — batch state

- Webhook callbacks to merchant-provided callback URL

2.2 Guarantees

- Idempotency: header Idempotency-Key (opaque ≤ 128 chars), enforced for 24h across regions; returns same 201 body if retried.

- Callback authenticity: header X-Signature: sha256=<hex>; body canonicalization per Appendix A.

- Ordering: callbacks per payout_id are ordered; batch callback includes per-item status.

2.3 Data Model (simplified)

```
payouts(
  payout_id PK, merchant_id, amount, currency, destination, provider_ref,
  status ENUM(queued, processing, succeeded, failed), failure_code, created_at, updated_at,
  idempotency_key UNIQUE(merchant_id, idempotency_key)
)
```

Batch metadata table stores batch_id, counts, and a pointer to the request payload blob.

3. Detailed API

3.1 Create Single Payout — POST /api/v2/payouts

Request

```
{
  "amount": 100.50,
  "currency": "USD",
  "destination": { "type": "bank_account", "iban": "US123..." },
  "callback": "https://merchant.example/payouts/cb"
}
```

Headers

- Authorization: Bearer <token>

- Idempotency-Key: 4b6e9f73-...

- X-Idem-TTL: 86400 (optional override up to 24h)

Response (201)

```
{ "payout_id":"po_9yY...", "status":"queued" }
```

Failure Codes

- INVALID_DESTINATION — invalid IBAN or account type

- UNSUPPORTED_CURRENCY

- INSUFFICIENT_FUNDS

- RATE_LIMITED — back off with Retry-After

- PROVIDER_UNAVAILABLE

3.2 Create Batch — POST /api/v2/payouts/batch

Request

```
{
  "items": [
    { "amount": 100, "currency":"USD", "destination": {"type":"bank_account","iban":"US1..."} },
    { "amount": 90,  "currency":"EUR", "destination": {"type":"bank_account","iban":"EU2..."} }
  ],
  "callback": "https://merchant.example/payouts/cb"
}
```

Response (201)

```
{ "batch_id":"bat_1hA...", "accepted": 2 }
```

3.3 Read State — GET /api/v2/payouts/{payout_id}

Returns current status; states are monotonic.

4. Callbacks

Contract

```
{
  "payout_id": "po_9yY",
  "status": "succeeded",
  "amount": 100.50,
  "currency": "USD",
  "signature": "sha256=<hex>",
  "replay_id": "rpl_7f2a9"
}
```

Signature generation

```
sha256( canonical_json(body) + "." + secret )
```

Retry Policy

- Exponential backoff: 5s, 30s, 2m, 10m, 1h (max 24h)

- Drops after max attempts; merchant can query state endpoint

Replay Protection

- replay_id stored for 24h; duplicates ignored

5. Architecture

- Gateway (authZ, quotas, idem precheck) → Payout Orchestrator → Provider Router
- Provider Adapters (AlphaPay, BetaWire) implement uniform interface
- Callback Worker delivers webhooks; stores retry schedule
- Observability: trace id in headers; spans across gateway/orchestrator/provider

Sequence Diagram (single payout)

1. Client → Gateway POST /payouts (Idem key)
2. Gateway → Orchestrator: create message
3. Orchestrator → Router: choose provider (routing policy)
4. Router → Provider Adapter: submit
5. Provider → Router: ack + provider_ref
6. Orchestrator: write row; enqueue callback job
7. Callback Worker: send signed webhook; schedule retries on non-2xx

6. Routing Policy

- Weighted round-robin with health checks
- Merchant tiering (Premium → more AlphaPay)
- Currency capability matrix
- Overload shedding: route 100% to least loaded healthy provider

7. Idempotency Semantics

- (merchant_id, idempotency_key) unique; conflicts return the original response
- Mutations are atomic; provider double-submissions guarded with provider-side idem keys when available

8. Error Taxonomy

Code | Description | HTTP

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

INVALID_DESTINATION | IBAN/account invalid | 400

UNSUPPORTED_CURRENCY | not enabled | 400

INSUFFICIENT_FUNDS | wallet balance too low | 402

RATE_LIMITED | quota exceeded | 429

PROVIDER_UNAVAILABLE | upstream down | 503

9. Observability

- payout.created, payout.submitted, payout.provider_ack, payout.callback_sent, payout.callback_failed
- Dashboards: latency (p50/p95/p99), success rate, callback retry backlog
- SLO: 99.9% callback delivery within 24h

10. Security & Compliance

- HMAC secrets rotated quarterly; per-merchant secrets
- PII minimization: destination hashed at rest with per-tenant key
- Audit log writes on all state transitions

11. Rollout & Backward Compatibility

- Dual-write for 1 week; v1 remains readable
- Migration guide: mapping of v1 → v2 error codes and callbacks
- Kill switch payouts_v2_disable (global)

12. Open Questions

- Should we support per-item callback endpoints in batch?
- Is 24h idempotency window sufficient for long-running workflows?

Appendix A — Canonical JSON Rules

- UTF-8 no BOM; keys sorted; no trailing zeros; numbers as strings allowed when flagged; stable object order within arrays.