# Payment Reconciliation Exception Report

Sample deliverable — this is the document delivered with every order

> **This is a sample.** The figures below are illustrative and do not come from a customer account. The exception classes, the matching logic and the report format are real and are what gets delivered. Your version runs against your processor and your books.

## 1. PERIOD AND SCOPE

One calendar month, one payment processor against one accounting source. Every processor transaction in the window is matched to a record in the books; every book record is matched back. Anything that fails to match cleanly in either direction becomes an exception.

Matching is deterministic: the same inputs always produce the same output. Nothing here depends on a judgment call the workflow makes differently on a second run. That matters because you may have to defend a number to an accountant, a lender, or yourself six months from now.

## 2. EXCEPTIONS FOUND

| Exception | Why it is an exception | Amount |
|---|---|---|
| Payout spans a period boundary | One deposit batched three days of sales across the month close. Recorded entirely in the later period, so both months are wrong. | $2,140.18 |
| Refund matched to the wrong original charge | Refund posted against a different sale of the same amount. Both rows look correct in isolation; only the pairing is wrong. | $318.00 |
| Processor fees never booked | Gross amounts recorded, fee side missing for the period. Revenue looks right. Margin does not. | $1,065.42 |
| Chargeback reversed, not reflected | Processor returned the funds after the dispute was won. The books still carry the loss. | $89.99 |
| Deposit with no matching payout | Money arrived in the bank that the processor does not account for. Usually a transfer from elsewhere, but it has to be explained, not ignored. | $450.00 |
| **Total value of unexplained differences** | | **$4,063.59** |

**Note what this total is and is not.** It is the amount of money whose treatment is wrong or unexplained. It is not money owed to you. Two of these five are timing and classification problems that cost nothing once corrected. Any report that presents a number like this as recoverable cash is selling you something.

## 3. HOW EACH ROW IS WRITTEN

Every exception carries the processor reference, the book reference where one exists, the amount, the rule that flagged it, and a plain-sentence reason. For example:

```
PAYOUT po_8841 ($2,140.18) covers transactions dated 29 Mar–02 Apr. Book entry JE-2214 records the full amount in April. Rule: period_boundary_split.
```

That sentence is written to be handed to a bookkeeper without further explanation. A flag you have to investigate from scratch is barely better than no flag.

## 4. THREE DESIGN DECISIONS WORTH EXPLAINING

**Unmatched in both directions.** Most reconciliation checks one way: every processor row must exist in the books. That misses entries in the books with nothing behind them, which is the more dangerous error. This runs both directions.

**A tolerance on amounts, not on counts.** Sub-cent rounding differences are absorbed; a missing transaction never is. Tolerance exists so the report stays readable, not so it stays short.

**Silence is the success signal.** On a scheduled run, a clean month sends nothing. An alert means something needs a human. A report that arrives every month regardless stops being read by the third month.

## 5. HONEST LIMITS

- **I find and report. I do not post.** Journal entries into your accounting system are yours or your bookkeeper's. I am not an accountant.
- Accuracy depends on both exports being complete for the same window. A partial export produces false exceptions, so the workflow checks date coverage before it compares anything.
- Where a processor exposes no API, this runs from exports. If those exports change shape month to month, that is the problem to fix first, and I will say so.
- It reconciles what both sides report. Cash that never touched either system is invisible to it.

---

Sample exception report prepared as a work sample. Built in n8n with a JavaScript matching node; runs against processor exports or APIs and an accounting source.