# 🛠️ BlockReceipt.ai – Vendor Verification & Promotion Attachment Guide

## 1. Goal

Enable BlockReceipt to automatically verify receipts from any merchant and attach vendor‑defined, time‑limited promotions—starting with heuristic matching (Option C) and scaling to Point‑of‑Sale webhooks (Option A) for Toast, Square, Clover, and eventually big‑box chains like Walmart, CVS, and Target.

## 2. Phased Roll‑Out

• \*\*Phase 0 – Heuristic Matching (launch)\*\*  
 ‑ Merchant directory regex + receipt fingerprint.  
 ‑ Default promos auto‑attached.  
• \*\*Phase 1 – Cloud POS Webhooks (Toast, Square, Clover)\*\*  
 ‑ /pos/webhook endpoint with HMAC verification.  
 ‑ Merchants self‑service in dashboard.  
• \*\*Phase 2 – Big‑Box Retail Integrations\*\*  
 ‑ E‑receipt API or printed watermark HMAC.  
 ‑ Pilot with CVS / Walmart digital teams.

## 3. Merchant Directory

File: `data/merchantDirectory.json`

```json  
[  
 {  
 "merchantId": "WALMART",  
 "regex": "WAL[- ]?MART",  
 "cityCode": "US\_WMT",  
 "defaultPromoTemplate": "ROLLBACK10"  
 },  
 {  
 "merchantId": "CVS",  
 "regex": "CVS",  
 "cityCode": "US\_CVS",  
 "defaultPromoTemplate": "CVS3OFF"  
 }  
]  
```

## 4. Receipt Fingerprint

```ts  
export interface ReceiptFP {  
 merchantName: string;  
 storeNumber?: string;  
 subtotal: number;  
 timestamp: number;  
}  
```

## 5. POS Webhook (Toast/Square)

File: `server/routes/posWebhook.ts`

```ts  
import express from 'express';  
import db from '../db';  
import { verifyToastSignature } from '../services/signatureService';  
  
const router = express.Router();  
  
router.post('/pos/webhook', verifyToastSignature, async (req, res) => {  
 const { merchantId, orderId, total, lineItems, timestamp } = req.body;  
 await db.posOrders.updateOne(  
 { merchantId, orderId },  
 { $set: { total, lineItems, timestamp } },  
 { upsert: true }  
 );  
 res.status(200).end();  
});  
export default router;  
```

## 6. uploadAndMint Enhancements

```ts  
// after OCR  
const fp: ReceiptFP = { merchantName, storeNumber, subtotal, timestamp };  
const merchant = await matchMerchant(fp.merchantName);  
  
let isVerified = false;  
if (merchant) {  
 const posMatch = await db.posOrders.findOne({  
 merchantId: merchant.merchantId,  
 total: fp.subtotal,  
 timestamp: { $gt: fp.timestamp - 5\*60\*1000, $lt: fp.timestamp + 5\*60\*1000 }  
 });  
 isVerified = !!posMatch;  
}  
  
const promoTemplate = merchant  
 ? await findApplicablePromo(merchant.merchantId, lineItems, subtotal)  
 : null;  
  
const promoData = promoTemplate ? encryptPromo(promoTemplate) : null;  
```

## 7. Promo Template

```json  
{  
 "merchantId": "CVS",  
 "title": "$3 Off Shampoo",  
 "code": "CVS3OFF",  
 "rules": { "category": "HairCare", "minSpend": 5 },  
 "expiresDays": 14,  
 "isActive": true  
}  
```

## 8. Merchant Dashboard MVP

• React admin route `/admin` (protected by Google OAuth).  
• Pages: Webhook Setup (URL & secret), Promo Templates CRUD.  
• Shows live receipts verified count.

## 9. Replit Integration Steps

1. Add `data/merchantDirectory.json` and helper `matchMerchant()`.  
2. Implement `/pos/webhook` with signature checks for Toast (later Square, Clover).  
3. Extend `uploadAndMint.ts` to verify receipts and attach promo via template.  
4. Create `PromoTemplate` model and CRUD API for dashboard.  
5. Front‑end gallery: display verification badge and promo button.