EasySale External Integrations — Provider Setup Checklist

Version 1.0 · Generated 2026-01-30

Purpose

This checklist is the **operator / developer setup** you (EasySale) do once, plus the **merchant/tenant setup** each store does (usually just clicking “Connect” and approving OAuth). It is written to match a white-label, multi-tenant POS where each tenant can bring their own provider account.

0) One-time prerequisites (applies to all providers)

You need these in place before OAuth/webhooks work reliably:

- **Public HTTPS base URL** for your backend (not localhost). Example: *https://pos.easysale.app*
- **Stable callback routes** implemented in backend: */api/integrations/{provider}/callback*
- **Secrets management**: all provider secrets live in environment variables (never in git)
- **Webhook endpoint** base path: */api/webhooks/{provider}* (even if Phase 1 doesn’t use all webhooks)
- For local dev: use a tunnel (ngrok/cloudflared) and set callback URLs to the tunnel domain

1) Stripe — Connect (recommended for multi-tenant)

What you (EasySale) set up once

- Create/verify a Stripe account for the EasySale **platform** (Test mode first).
- Enable **Stripe Connect** in the Stripe dashboard: Settings → Connect → Get started.
- In Connect settings, add your **OAuth redirect URI**: *https://YOUR_DOMAIN/api/integrations/stripe/callback*
- Create a **webhook endpoint** for your platform: Developers → Webhooks → Add endpoint → *https://YOUR_DOMAIN/api/webhooks/stripe*
- Select minimum events (for Phase 2 payments): *checkout.session.completed*, *payment_intent.succeeded*, *account.updated*.
- Copy platform keys into env vars (server only): **STRIPE_SECRET_KEY** (and optionally STRIPE_PUBLISHABLE_KEY), **STRIPE_WEBHOOK_SECRET**.

What each merchant/tenant does

- In EasySale: Settings → Integrations → Stripe → click **Connect Stripe**.
- Merchant completes Stripe’s OAuth onboarding; you receive a connected account id (*acct_...*).
- Back in EasySale: status shows **Connected**; **Test Connection** calls Stripe to verify access.

Notes

- If you are using **Stripe Checkout** (Phase 2), payments are created on the platform but paid out to the connected account using the connect account id.
- Do not support raw API keys per tenant unless you are intentionally doing a single-merchant deployment.

2) Square — OAuth app (multi-tenant)

What you (EasySale) set up once

- Create a Square Developer account and create an **Application**.
- Set OAuth redirect URL: *https://YOUR_DOMAIN/api/integrations/square/callback*
- Choose scopes (minimum for connection test; expand later): *MERCHANT_PROFILE_READ*, *LOCATIONS_READ* (and for payments in Phase 2: *PAYMENTS_WRITE*, *PAYMENTS_READ*).
- Store in env vars: **SQUARE_APPLICATION_ID**, **SQUARE_APPLICATION_SECRET**.
- Optional for Phase 2: set webhook endpoint: *https://YOUR_DOMAIN/api/webhooks/square*.

What each merchant/tenant does

- In EasySale: Settings → Integrations → Square → click **Connect Square**.
- Merchant authorizes; EasySale stores the OAuth token securely; status shows Connected.

3) Clover — OAuth app (multi-tenant)

What you (EasySale) set up once

- Create a Clover Developer account and create an **App**.
- Set redirect URL: *https://YOUR_DOMAIN/api/integrations/clover/callback*
- Record App ID/Secret and store in env vars: **CLOVER_APP_ID**, **CLOVER_APP_SECRET**, **CLOVER_REDIRECT_URI**.
- Ensure backend implements CSRF state tracking for OAuth.

What each merchant/tenant does

- In EasySale: Settings → Integrations → Clover → click **Connect Clover**.
- Merchant authorizes; EasySale stores merchant_id + tokens encrypted; status shows Connected.

4) WooCommerce — REST API keys (per merchant store)

What each merchant/tenant does

- In WooCommerce admin: WooCommerce → Settings → Advanced → REST API → **Add key**.

- Permissions: **Read/Write** (or Read-only if you only pull).
- Copy **Consumer Key** and **Consumer Secret** into EasySale's WooCommerce integration form.
- Ensure permalinks are enabled and site is accessible over HTTPS.

What you (EasySale) do

- Provide clear UI for base URL, consumer key/secret, test connection, and sync policy (already exists per your audit).

5) Supabase Hub — project + service key (single hub, tenant scoped)

If you use Supabase as a centralized hub, you typically create one project and store its credentials in EasySale. Tenant isolation must be enforced by schema + application logic.

- Create Supabase project (hub) and copy: **SUPABASE_URL** and **SUPABASE_SERVICE_ROLE_KEY** into server env vars.
- Confirm tables include **tenant_id** on every row and all queries filter by tenant_id.
- If using Row Level Security (RLS), define policies consistent with tenant_id (optional for Phase 1, recommended for Phase 2).

6) Demo readiness: minimal checks

Before tomorrow's run-through, validate these end-to-end:

- **Stripe**: Connect (OAuth) → Status shows Connected → Test Connection passes → Disconnect works → Logs show events.
- **Square**: Connect → Status → Test → Disconnect → Logs.
- **Clover**: Connect OAuth → Status → Test → Disconnect → Logs.
- **WooCommerce**: Test connection passes and one safe operation (e.g., list products).
- **Supabase**: Hub status loads (last sync time, pending queue count) and tenant scope enforced.

Environment variable summary

Set these on the backend host (never commit to git):

```
STRIPE_SECRET_KEY=sk_live_... (platform)
STRIPE_WEBHOOK_SECRET=whsec_...
SQUARE_APPLICATION_ID=sq0idp-...
SQUARE_APPLICATION_SECRET=sq0csp-...
CLOVER_APP_ID=...
CLOVER_APP_SECRET=...
```

```
CLOVER_REDIRECT_URI=https://YOUR_DOMAIN/api/integrations/clover/callback
SUPABASE_URL=https://xxxx.supabase.co
SUPABASE_SERVICE_ROLE_KEY=...
PUBLIC_BASE_URL=https://YOUR_DOMAIN (used to build redirect URLs safely)
```