ClearPath RD

clearpathrd.com

Stripe Integration Specification

Version 1.0 — February 2026 — Confidential

What this document covers

- Stripe account setup — products, prices, tax configuration, and dashboard settings
- Checkout experience decision — Hosted Checkout vs Stripe Elements compared side by side
- Complete checkout flow for both options — API calls, redirects, and state transitions
- Stripe Tax configuration for full Canadian coverage (GST, HST, PST, QST)
- Referral code discount application at checkout
- Webhook event map — every event ClearPath handles or ignores, with exact actions
- Webhook security — signature verification implementation
- Refund handling — full and partial refund flows and session state consequences
- Test mode checklist — cards, amounts, and scenarios to verify before going live
- Environment variables and Stripe dashboard configuration checklist

1. Stripe Account Setup

1.1 Products and Prices

Three products must be created in the Stripe dashboard before any orders can be processed. Each product has one price. Prices are fixed (not metered). Currency: CAD.

Product name	Env var	Price (CAD)	Description
ClearPath RD — Long-Term Kit	`STRIPE_PRICE_ID_STANDARD_LONG`	**$54.99**	Standard 91-day long-term radon test kit. Maps to product_sku = standard_long.
ClearPath RD — Real Estate Kit	`STRIPE_PRICE_ID_REAL_ESTATE_SHORT`	**$89.99**	48–96 hour real estate short-term kit. Maps to product_sku = real_estate_short.
ClearPath RD — Twin Pack	`STRIPE_PRICE_ID_TWIN_PACK`	**$99.99**	Two long-term kits. Maps to product_sku = twin_pack. Note: this is one Stripe line item with quantity 1, not two line items.

Product setup notes:

- Set product type to "One-time" (not recurring). ClearPath does not use subscriptions.
- Enable "Tax code" on each product. Use tax code physical_goods for all three — radon test kits are physical goods shipped to a Canadian address.
- Do not create separate test-mode and live-mode products manually — Stripe maintains separate environments. Create in test mode first, then recreate in live mode when ready to launch.
- Price IDs (price_...) are different between test mode and live mode. Maintain both sets of env vars.

1.2 Stripe Tax Configuration

> *Stripe Tax handles Canadian GST, HST, and provincial taxes automatically based on the customer's shipping postal code. This eliminates the need for a manual tax table in the API. ClearPath must register a tax nexus in each province where it is required to collect tax.*

Setup steps in the Stripe dashboard:

- Go to Stripe Dashboard → Tax → Settings → Enable automatic tax calculation
- Set your business address to the ClearPath RD registered address (Canada)
- Add a tax registration for each province where ClearPath has nexus (initially: province of incorporation + provinces where significant revenue is earned)
- Set the tax behaviour on each price to "exclusive" (tax added on top of the listed price, not included)

Canadian tax rates by province — Stripe Tax applies these automatically:

Province / Territory	GST	PST/QST	HST	Notes
Alberta (AB)	5%	—	—	GST only. No provincial sales tax.
British Columbia (BC)	5%	7% PST	—	GST + PST. Combined: 12%.
Manitoba (MB)	5%	7% RST	—	GST + Retail Sales Tax. Combined: 12%.
Saskatchewan (SK)	5%	6% PST	—	GST + PST. Combined: 11%.
Quebec (QC)	5%	9.975% QST	—	GST + QST. Stripe Tax handles QST registration separately.

Ontario (ON)	—	—	13% HST	HST replaces GST + PST.
New Brunswick (NB)	—	—	15% HST	HST.
Newfoundland (NL)	—	—	15% HST	HST.
Nova Scotia (NS)	—	—	15% HST	HST.
PEI (PE)	—	—	15% HST	HST.
NWT / Nunavut / Yukon	5%	—	—	GST only. No territorial sales tax.

GST registration note: ClearPath RD must register for GST/HST with the CRA once annual revenue exceeds $30,000 CAD. Below this threshold, GST collection is optional but Stripe Tax can still be configured. Consult an accountant before enabling tax collection. Until registered, set tax behaviour to "exclusive" but do not add tax registrations in Stripe.

1.3 Stripe Dashboard Settings

Property	Specification
Customer emails	Enable "Send receipts to customers" in Stripe Dashboard → Settings → Emails. Stripe sends a payment receipt automatically. This is separate from the ClearPath order_confirm email — users receive both.
Customer portal	Not required for MVP. ClearPath manages refunds manually via the admin dashboard.
Branding	Upload ClearPath RD logo, set brand colour to #1A7A6E (teal), set icon. Applies to Stripe Checkout and receipt emails.
Statement descriptor	Set to "CLEARPATH RD" (max 22 chars). This is what appears on the customer's credit card statement.
Dispute handling	Enable email notifications for disputes. Respond to disputes within 7 days with: order confirmation, shipping proof, and product description.

2. Checkout Experience Options

Two checkout options are available. The decision has not been made yet — both are fully specified here. Choose before beginning frontend development.

	Option A — Stripe Checkout	Option B — Stripe Elements
Payment UI	Hosted by Stripe. User redirected to stripe.com subdomain.	Embedded in ClearPath UI. User never leaves the site.
PCI compliance	Stripe handles entirely. No PCI scope for ClearPath.	SAQ A (minimal). Card data goes directly to Stripe, not ClearPath servers.
Customisation	Limited. Logo, colours, background. Cannot match ClearPath brand fully.	Full control. Match brand exactly.
Apple/Google Pay	Included automatically.	Must be enabled and configured separately.
Dev complexity	Low. One API call to create session, redirect, handle webhook.	Medium. Load Stripe.js, mount Elements, handle confirm flow.
Referral code UX	Applied before redirect via metadata. Cannot show discount on Stripe page.	Can show discount applied inline before payment.
Mobile experience	Stripe-optimised. Consistent across devices.	Depends on ClearPath implementation quality.
Recommendation	Good for MVP — fastest to implement, lowest risk.	Better long-term UX. Consider for v2 or if brand consistency is critical at launch.

Recommendation: start with Stripe Checkout (Option A) for MVP. It is faster to implement, requires no frontend payment form work, and handles all edge cases (3D Secure, Apple/Google Pay, address validation) automatically. Migrate to Stripe Elements in v2 if brand consistency at the payment step becomes a priority.

3. Checkout Flow — Option A (Stripe Checkout)

3.1 Flow Overview

- 1. User selects kit and clicks "Buy now" on the ClearPath frontend
- 2. Frontend calls POST /api/v1/orders on the Fastify API
- 3. Fastify creates a KitOrder record (payment_status = pending), generates kit serial(s)
- 4. Fastify calls stripe.checkout.sessions.create() with line items, tax, and metadata
- 5. Fastify returns { order, checkoutUrl } to the frontend
- 6. Frontend redirects the browser to checkoutUrl (Stripe-hosted page)
- 7. User completes payment on Stripe's page
- 8. Stripe redirects user to success_url or cancel_url
- 9. Stripe sends payment_intent.succeeded webhook to /api/v1/webhooks/stripe
- 10. Fastify webhook handler: sets payment_status = paid, creates TestSession(s), submits to lab, sends order_confirm email

Do not rely on the success_url redirect to confirm payment. The redirect can be intercepted, duplicated, or faked. The webhook is the only authoritative confirmation of payment. The success page should poll GET /api/v1/orders/:id until payment_status = paid before showing a confirmation.

3.2 Stripe Checkout Session Creation

Fastify API call to Stripe when creating a checkout session:

```
const session = await stripe.checkout.sessions.create({
  mode: 'payment',
  line_items: [{
    price: process.env.STRIPE_PRICE_ID_STANDARD_LONG,
    quantity: 1,
  }],
  customer_email: user.email,
  automatic_tax: { enabled: true },
  shipping_address_collection: { allowed_countries: ['CA'] },
  shipping_options: [{ shipping_rate: 'shr_...' }], // free shipping rate
  success_url: `${process.env.APP_BASE_URL}/checkout/success?order_id={order.id}`,
  cancel_url: `${process.env.APP_BASE_URL}/checkout/cancelled`,
  metadata: {
    clearpath_order_id: order.id,
    clearpath_user_id: user.id,
    referral_code_id: order.referralCodeId ?? '',
  },
  payment_intent_data: {
    metadata: {
      clearpath_order_id: order.id,
    },
  },
});
```

Important notes on the session creation call:

- metadata is included on both the session and the payment_intent_data. The webhook receives the PaymentIntent object, not the Session object, so metadata must be on the payment_intent to be accessible in the webhook handler.
- shipping_address_collection restricts to CA (Canada only). This prevents international orders which ClearPath does not support at MVP.
- automatic_tax: { enabled: true } activates Stripe Tax. Stripe calculates the correct tax based on the shipping address the user enters.
- The free shipping rate (shr_...) must be created in the Stripe dashboard under Products → Shipping rates before it can be referenced here.

3.3 Referral Code Discounts

Referral code discounts are applied as Stripe Coupons. The coupon is created dynamically at checkout time if a valid referral code is present.

- Before creating the checkout session, validate the referral code via the referral_codes table
- If valid and has a discount_pct, create a Stripe coupon: stripe.coupons.create({ percent_off: discountPct, duration: 'once' })
- Apply the coupon to the checkout session via the discounts parameter: [{ coupon: coupon.id }]
- Store the coupon ID in the KitOrder metadata for audit purposes
- If the referral code has no discount (source tracking only, no price reduction), do not create a coupon — just store the referral_code_id on the order

3.4 Post-Payment Success Page

The success_url receives the user after payment. The page at /checkout/success must:

- Extract order_id from the URL query parameter
- Poll GET /api/v1/orders/:id every 2 seconds until payment_status = paid (max 30 seconds)
- Show a loading state while polling — "Confirming your order..."
- Once payment_status = paid, show the order confirmation screen with order details and next steps
- If polling times out (30 seconds), show: "Your payment was received. Your order confirmation email is on its way. If you have questions, contact support@clearpathrd.com."

4. Checkout Flow — Option B (Stripe Elements)

4.1 Flow Overview

With Stripe Elements, the payment form is embedded in the ClearPath UI. The flow differs from Checkout at steps 4–8:

- 1–3: Same as Option A
- 4. Fastify calls stripe.paymentIntents.create() instead of checkout.sessions.create()
- 5. Fastify returns { order, clientSecret } to the frontend
- 6. Frontend uses the clientSecret to mount a Stripe Payment Element on the page
- 7. User fills in card details directly on the ClearPath page
- 8. Frontend calls stripe.confirmPayment() — Stripe handles 3D Secure if required
- 9–10: Same webhook flow as Option A

4.2 PaymentIntent Creation

```
const paymentIntent = await stripe.paymentIntents.create({
  amount: order.totalCad * 100, // Stripe uses cents
  currency: 'cad',
  customer_email: user.email,
  metadata: { clearpath_order_id: order.id, clearpath_user_id: user.id },
  automatic_payment_methods: { enabled: true },
});
```

Tax handling with Elements: Stripe Tax does not integrate automatically with PaymentIntents the way it does with Checkout Sessions. Two options:

- Option B1: Calculate tax server-side using stripe.tax.calculations.create() before creating the PaymentIntent. Include the tax amount in the PaymentIntent amount.
- Option B2: Use Stripe's tax calculation API to get the tax amount, display it in the UI, then include it in the final PaymentIntent amount.
- Either way, the tax amount must be calculated and stored in kit_orders.tax_cad before the PaymentIntent is confirmed.

5. Webhook Event Map

The Stripe webhook endpoint is POST /api/v1/webhooks/stripe. All events are received here. The handler verifies the Stripe-Signature header before processing any event.

Idempotency rule: the webhook handler must be idempotent. Stripe may deliver the same event more than once. Before processing any event, check if the PaymentIntent ID has already been processed (check kit_orders.stripe_payment_intent_id). If found with the correct status, return HTTP 200 without re-processing.

5.1 Handled Events

Stripe event	Handling	ClearPath action
`payment_intent.succeeded`	**Process**	Set KitOrder.payment_status = paid, set paid_at = NOW(). Create TestSession record(s) with status = ordered. Call labService.submitOrder(). Send order_confirm email. Log to audit.
`payment_intent.payment_failed`	**Process**	Set KitOrder.payment_status = failed. No TestSession created. Send no email. Log failure reason from event.last_payment_error.message for admin visibility.
`charge.refunded`	**Process**	Set KitOrder.payment_status = refunded. Set TestSession(s) status = cancelled. Cancel any queued email_log rows for the session(s). Send no automated email — admin handles customer communication.
`charge.dispute.created`	**Process**	Log dispute. Send admin email alert. Flag KitOrder with dispute metadata for manual review. Do not cancel session automatically — await dispute resolution.
`charge.dispute.closed`	**Process**	If dispute lost: same handling as charge.refunded. If dispute won: no action, restore KitOrder to normal status.
`checkout.session.completed`	Ignore	Not used. ClearPath uses payment_intent.succeeded as the authoritative payment confirmation.
`checkout.session.expired`	**Process**	If the Stripe Checkout session expires before payment, set KitOrder.payment_status = failed. Log for diagnostics.
`payment_intent.canceled`	**Process**	Set KitOrder.payment_status = failed. Same as payment_failed.
`customer.created`	Ignore	ClearPath does not create Stripe Customer objects. customer_email is passed per-session.
`invoice.*`	Ignore	No subscriptions. All invoice events are irrelevant.
`All other event types`	Ignore	Acknowledge with HTTP 200, no processing. Log event type for diagnostics.

5.2 Webhook Signature Verification

Every inbound webhook request must have its Stripe-Signature header verified before any processing. This prevents spoofed webhook events.

```
import Stripe from 'stripe';
const stripe = new Stripe(process.env.STRIPE_SECRET_KEY);
```

```
// In the Fastify webhook route handler:
const sig = request.headers['stripe-signature'];
const rawBody = request.rawBody; // Fastify must provide raw body, not parsed JSON
```

```
let event: Stripe.Event;
try {
  event = stripe.webhooks.constructEvent(rawBody, sig,
process.env.STRIPE_WEBHOOK_SECRET);
} catch (err) {
  reply.status(400).send({ error: 'Invalid signature' });
  return;
}
```

Raw body requirement: Stripe signature verification requires the raw, unparsed request body as a Buffer or string. Fastify must be configured to preserve the raw body for the /webhooks/stripe route specifically. Use the addContentTypeParser plugin or the rawBody option to achieve this. Parsing the body as JSON before verification will break signature checking.

6. Refund Handling

6.1 Refund Policy

ClearPath's refund policy must be defined before launch. The technical implementation supports the following policy framework — the exact policy is a business decision:

- Full refund: available if the kit has not yet been shipped (KitOrder.fulfilled_at is null).
- Partial refund: kit cost minus a restocking/processing fee if the kit has been shipped but not activated.
- No refund: once the kit is activated (TestSession.status = active), the test is in progress and cannot be reversed.

6.2 Refund Flow (Admin-Initiated)

Refunds are initiated manually by an admin via the Stripe dashboard or the admin panel in the ClearPath app. There is no user-facing self-serve refund for MVP.

- 1. Admin navigates to the order in the admin dashboard
- 2. Admin clicks "Issue refund" — specifies full or partial amount
- 3. ClearPath API calls stripe.refunds.create({ payment_intent: order.stripePaymentIntentId, amount: refundAmountCents })
- 4. Stripe processes the refund and sends a charge.refunded webhook
- 5. Webhook handler sets KitOrder.payment_status = refunded
- 6. If TestSession exists and is not complete: set status = cancelled
- 7. Cancel queued email_log rows for the cancelled session(s)
- 8. Admin communicates with the customer directly — no automated email for refunds in MVP

6.3 Partial Refunds

Stripe supports partial refunds by specifying an amount in cents less than the original charge. The charge.refunded event fires for partial refunds as well. ClearPath distinguishes full vs partial refunds by comparing the refund amount to the original total:

- If refunded amount = original total: set payment_status = refunded
- If refunded amount < original total: set payment_status = partially_refunded (add this status to the PaymentStatus enum if needed, or handle via a separate refunded_amount_cad field on KitOrder)

7. Environment Variables and Key Management

Property	Specification
STRIPE_SECRET_KEY	Server-side only. Never in frontend. Starts with sk_test_ in test mode, sk_live_ in production. Used for all Stripe API calls from the Fastify API.
STRIPE_PUBLISHABLE_KEY	Frontend only (NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY). Starts with pk_test_ / pk_live_. Used to initialise Stripe.js in the browser (Option B only).
STRIPE_WEBHOOK_SECRET	Server-side only. Starts with whsec_. Used to verify webhook signatures. One secret per registered webhook endpoint. Different values in test mode and live mode.
STRIPE_PRICE_ID_STANDARD_LONG	Server-side. The price_... ID for the $54.99 long-term kit. Different in test vs live.
STRIPE_PRICE_ID_REAL_ESTATE_SHORT	Server-side. The price_... ID for the $89.99 real estate kit.
STRIPE_PRICE_ID_TWIN_PACK	Server-side. The price_... ID for the $99.99 twin pack.
STRIPE_SHIPPING_RATE_FREE	Server-side. The shr_... ID for the free shipping rate. Option A only.

Key rotation: if STRIPE_SECRET_KEY or STRIPE_WEBHOOK_SECRET is ever exposed (committed to Git, logged, etc.), rotate it immediately in the Stripe dashboard and update the environment variables in Render. Stripe allows creating new API keys without invalidating existing ones, giving a safe rotation window.

8. Test Mode Checklist

Complete all of these in Stripe test mode before switching to live mode. Use Stripe's test card numbers and the Stripe CLI to simulate webhook events.

8.1 Test Cards

Test card number	Expected behaviour
4242 4242 4242 4242	Successful payment. Any future expiry, any 3-digit CVV, any postal code.
4000 0027 6000 3184	Requires 3D Secure authentication. Tests the 3DS flow.
4000 0000 0000 9995	Card declined (insufficient funds). Tests payment_failed flow.
4000 0000 0000 0002	Card declined (generic). Tests payment_failed flow.
4000 0025 0000 3155	Requires 3DS but the bank does not support it — payment fails.

8.2 Scenarios to Test

- Successful payment — confirm order_confirm email sent, TestSession created, KitOrder.payment_status = paid
- Payment failure — confirm KitOrder.payment_status = failed, no TestSession created, no email sent
- 3D Secure success — confirm same outcome as successful payment
- Webhook delivered twice — confirm idempotency, no duplicate TestSession or duplicate email
- Successful payment with referral code — confirm discount applied, coupon visible on Stripe receipt
- Referral code with no discount (tracking only) — confirm no coupon created, referral_code_id stored on order
- Canadian tax calculation — test an Alberta shipping address (GST only) and an Ontario address (HST 13%)
- Stripe Checkout session expires before payment — confirm KitOrder.payment_status = failed
- Refund initiated — confirm charge.refunded webhook fires, payment_status = refunded, TestSession = cancelled
- Twin pack order — confirm two TestSession records created, two kit serials generated, one order_confirm email

8.3 Stripe CLI Webhook Testing

Use the Stripe CLI to forward test webhooks to a local Fastify instance during development:

```
stripe listen --forward-to localhost:3001/api/v1/webhooks/stripe
```

Trigger specific events manually:

```
stripe trigger payment_intent.succeeded
stripe trigger payment_intent.payment_failed
stripe trigger charge.refunded
```

8.4 Going Live Checklist

- Create live-mode products and prices in Stripe dashboard (separate from test-mode)
- Copy live-mode price IDs to Render environment variables (STRIPE_PRICE_ID_*)

- Register live-mode webhook endpoint: https://api.clearpathrd.com/api/v1/webhooks/stripe
- Copy live-mode webhook signing secret (whsec_...) to Render env vars
- Set STRIPE_SECRET_KEY to live-mode key (sk_live_...) in Render
- If using Stripe Elements: set NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY to live pk_live_ key in Vercel
- Enable Stripe Tax registrations for applicable provinces
- Set statement descriptor to "CLEARPATH RD"
- Upload logo and set brand colours in Stripe dashboard
- Confirm live-mode test transaction processes correctly before announcing launch

— *End of Stripe Integration Specification* —

ClearPath RD | clearpathrd.com | Version 1.0 | February 2026