NEUTARGET / IMPLEMENTATION

Website installation guide

A practical handoff for your website, engineering and growth teams.

Installation pack 1.0.3 | 2026-11-21

1. Choose one platform path	2. Connect the outcome	3. Verify received evidence

IN THIS GUIDE

1. Prepare your website
2. Place one browser tracker
3. Connect privacy and optional identity
4. Connect your selected business outcome
5. Connect payments from your server
6. Shopify
7. WooCommerce
8. Verify and hand over

Use the companion code files for implementation. PDF code may wrap for reading. Keep Get started open for the current verification state.

Connect the website first, then the business outcome. The steps below follow Website setup, opened through Get started inside a selected website. Keep it open to see received verification status.

1. Prepare your website

Start at Dashboard > Websites. Choose or add a website, then an editor or admin selects its platform and primary outcome in Get started. Copy the website UUID from Website setup or Website settings > General. An admin provides the private website API secret through an approved secret-sharing process when a server or store integration needs it. The selected website's configuration is in Website settings.

The website ID is public configuration. The API secret authenticates financial receipt submissions and belongs only on a server or in protected platform settings. Do not prefix it with NEXT_PUBLIC_, embed it in HTML, or put it in a frontend environment file. Replace any credential previously exposed in a browser before relying on its secrecy.

2. Place one browser tracker

For a custom HTML site, WordPress header tool, Magento storefront, Angular, Vue or React SPA, use `browser/site-tag.html` in the shared HTML layout. Replace the placeholder website ID. Publish once for the entire site; do not install a tag per page. For a React SPA, use the HTML document, not a script tag recreated in every component render.

browser/site-tag.html

```
<!-- Add once in your shared HTML layout. Replace YOUR_WEBSITE_ID. -->
<!-- Skip this tag if your platform integration already installs the tracker. -->
<script defer data-website-id="YOUR_WEBSITE_ID"
  src="https://cdn.neutarget.com/script/v1/nt.min.js"></script>
```

For Next.js, use `browser/nextjs-layout.tsx.example` inside the body of the root layout. It uses next/script with afterInteractive. Merge the example with your existing layout; do not replace the rest of your application.

browser/nextjs-layout.tsx.example

```
import Script from "next/script";

// Merge inside <body> in your existing root layout. Replace YOUR_WEBSITE_ID.
// Skip when an existing platform integration already installs the tracker.
<Script
  src="https://cdn.neutarget.com/script/v1/nt.min.js"
  data-website-id="YOUR_WEBSITE_ID"
  strategy="afterInteractive"
/>
```

Shopify app embeds and the WooCommerce plugin can install the tracker for you. If you use them, do not add the manual tag as well. For a restrictive Content Security Policy, permit the SDK's CDN and its HTTPS collection origin according to your policy; inspect blocked requests rather than disabling CSP.

Publish and open a public page. Confirm Get started shows a received browser visit. On a single-page application, verify navigation to another route too. NeuTarget's browser SDK sets no analytics cookies; optional context and consent still follow your configured collection policy.

3. Connect privacy and optional identity

An admin reviews the selected website's Privacy and consent settings before collecting data. If your policy requires consent, connect your consent manager to the SDK. `browser/consent-bridge.js` exports a small adapter you can call after the tag loads. On initial load pass the existing decision; on every later change pass the updated decision. Before a decision, keep pending. Never turn a missing choice into granted.

browser/consent-bridge.js

```
// Browser module. Call from your consent manager AFTER the tracker loads.
// Pass its initial stored choice and every later change. Never assume consent.
export function applyAnalyticsConsent(decision) {
  if (!["granted", "denied", "pending"].includes(decision)) {
    throw new Error("Use the actual granted, denied or pending consent state");
  }
  return window.neutarget?.setConsent?.(decision)
    ?? { ok: false, code: "tracker_not_loaded" };
}
```

If your consent manager allows scripts only after permission, load the tag at that point and then apply the existing decision. The adapter intentionally does not install a consent banner or wait indefinitely for a missing tracker. An unready or denied tracker must not prevent normal website activity.

Declared customer identity is optional. In an identified collection mode, your server may derive a site-scoped HMAC from an internal customer ID using a private key and provide the resulting digest to the SDK's setUserHash method after your collection conditions are met. Clear it with clearUserHash on sign-out. Do not send raw email, names, phone numbers or identifiers in page paths, event names, UTM fields or error details. A customer digest remains pseudonymous data.

4. Connect your selected business outcome

For a lead, booking, product trial or another nonfinancial action, select a completed-action definition as the website default in Website setup before testing its exact event name. Call `recordOutcome` from `browser/record-outcome.js` only after your application confirms success. For example, wait for the server to accept the lead or create the booking. A button click is not confirmation.

browser/record-outcome.js

```
// Browser module. Call AFTER your application confirms the action succeeded.
// Save the same event name as your primary goal before testing.
export async function recordOutcome(eventName) {
  try {
    const context = await window.neutarget?.whenReady?.({ timeoutMs: 1500 });
    if (!context?.ready) return false;
    window.neutarget.track(eventName);
    return true; // Emitted, not a guarantee of storage; verify in Get started.
  } catch {
    return false; // Analytics must not fail the customer's successful action.
  }
}
```

Example: after a successful trial response, call recordOutcome("trial_started"). Use the same name as the saved goal. The helper is browser code; include it in your frontend bundle. It sends no financial value. When recording is unavailable it returns false without failing the customer's successful action.

Complete the real action and refresh verification. The matching browser event must arrive after the current goal's effective time. Changing the website default requires new evidence for that version. The reusable library in Analysis > Conversion Goals is separate from that default: saving a definition does not select it. Managed campaigns retain their pinned outcome definition until an explicit campaign revision changes it. Website analytics first needs no extra outcome event; you can add one later.

5. Connect payments from your server

Custom commerce, SaaS invoices and paid bookings use the same financial-event API. Shopify and WooCommerce users should follow their platform path below instead of adding a duplicate receipt adapter.

Optional frontend handoff: use `browser/checkout-context.js` in your existing checkout flow. Send available sessionHash and consentStatus to your own authenticated create-order endpoint, alongside your normal order input. Validate consent against your consent process and persist allowed context with the order. The server still validates the cart, prices, amount and currency itself. Missing context must not block checkout.

browser/checkout-context.js

```
// Browser module. Merge the result into your EXISTING checkout request.
// Your server validates the cart and retains allowed context with the order.
export async function checkoutContext() {
  try {
    const context = await window.neutarget?.whenReady?.({ timeoutMs: 1500 });
    const privacy = window.neutarget?.getPrivacyState?.();
    return {
      sessionHash: context?.ready ? context.sessionHash : null,
      consentStatus: context?.ready && privacy?.ready ? privacy.consent : null
    };
  } catch {
    return { sessionHash: null, consentStatus: null };
  }
}
```

Backend setup: copy `server/client.mjs`, `server/currency-exponents.json` and `server/report-payment.mjs` together into a private backend directory. Configure the two variables in `server/.env.example` through your deployment's secret storage. The helper uses Node.js 20+ and built-in fetch; it does not automatically load an .env file.

For Java, PHP, Python, .NET or another backend, use the same authenticated HTTPS/JSON contract in `server/http-contract.md`. The Node helper is optional; provider verification and durable delivery are required regardless of language.

server/.env.example

```
# Backend only. Supply through your application's environment/secret storage.
# Never use NEXT_PUBLIC_ or publish this file in your website's assets.
NEUTARGET_WEBSITE_ID=YOUR_WEBSITE_ID
NEUTARGET_API_SECRET=REPLACE_WITH_PRIVATE_WEBSITE_SECRET
```

In your existing webhook receiver, verify the provider signature and canonical payment state, then persist a durable job before acknowledging the provider. Map the verified capture, refund or chargeback to `reportVerifiedPayment`. It is an adapter for an already verified provider record, not a webhook signature verifier. Keep this module out of frontend imports and public assets.

server/report-payment.mjs

```
import { reportFinancialEvent } from "./client.mjs";

// Server only. Your webhook handler must verify the provider FIRST and
// persist an immutable retry job BEFORE acknowledging the provider.
// Map a canonical provider record to this contract; do not pass browser input.
export async function reportVerifiedPayment(verified, storedOrder) {
  if (typeof verified.provider !== "string" || !/^[a-z0-9_-]{1,40}$/i.test(verified.provider)
      || typeof verified.providerEventId !== "string" || !verified.providerEventId
      || verified.providerEventId.length > 140) {
    throw new Error("Keep the provider and original event ID with the retry job");
  }
  return reportFinancialEvent({
    idempotencyKey: `${verified.provider}:${verified.kind}:${verified.providerEventId}`,
    orderId: verified.orderId,
    kind: verified.kind, // capture | refund | chargeback
    amountMinor: verified.amountMinor, // Exact positive decimal string
    currency: verified.currency, // Explicit ISO code from the provider
    occurredAt: verified.occurredAt, // Original event time; preserve on retries
    sessionHash: storedOrder.sessionHash ?? null,
    consentStatus: storedOrder.consentStatus ?? null,
    customerStatus: "unknown"
  }, {
    websiteId: process.env.NEUTARGET_WEBSITE_ID,
    apiSecret: process.env.NEUTARGET_API_SECRET
  });
}
```

Read amountMinor as a positive exact decimal string in the provider's currency scale. Use decimalToMinor from client.mjs if your provider supplies a decimal major-unit string. Do not multiply JavaScript floating-point amounts. Preserve the original provider occurrence time, order ID and financial event ID across retries. Use the same order ID for its capture and adjustments, and a distinct stable key for each event. Deliver the capture before its refund or chargeback.

A 200 recorded or duplicate response acknowledges a durable receipt; analytics delivery may still be pending. Persist and retry transient failures with the same payload and key. A 401/403 requires credential/access review; a 409 requires a conflict review, not a new key. A 204 collection_closed response means collection policy closed the event; stop retrying it as though it were a network error and review the website state. The client's short retries are not a durable job queue.

6. Shopify

Choose one path for the store. When your store has access to the NeuTarget Shopify app, install it through Shopify's supported install flow, bind the corresponding website using its private website credentials in the authenticated app admin, enable the NeuTarget theme app embed and save the published theme. Inspect the app's delivery state as well as Get started. Public listing availability is separate from app installation and can change.

For an existing manual connection, open Website settings > Integrations and use the Shopify configuration. Save the canonical myshopify.com domain, correct currency and signing secret. Register the signed orders/paid, refunds/create and order_transactions/create topics at the webhook URL displayed by NeuTarget. Install the public tracker once in the published theme.

The optional `platforms/shopify-storefront.js` supports standard theme/cart context handoff. Load it after the tracker and await neutargetPrepareShopifyCheckout in your actual checkout handler, always allowing

checkout to continue on failure. Loading the file alone does not intercept checkout. Review https://www.neutarget.com/integrations/shopify for the handler example. Cart drawers and accelerated checkout may need a different supported handoff. An eligible paid order still appears without that attribution context.

Test a paid order and a successful refund transaction. Refund creation alone does not prove money movement. Compare financial source, amount, currency, refund and any unattributed balance in NeuTarget with Shopify.

7. WooCommerce

Upload only `platforms/neutarget-woocommerce-1.1.2.zip` at WordPress > Plugins > Add New > Upload Plugin, then activate it. This requires WooCommerce 8.2+ and PHP 8+. Open WooCommerce > Settings > Integration > NeuTarget Analytics. Enter the matching website UUID and private website secret, enable storefront tracking and save. The plugin encrypts the stored secret using WordPress salts; server-side wp-config.php constants are an alternative.

Do not add the custom browser tag or Node adapter when this plugin handles those functions. The plugin supports Classic checkout and Checkout Blocks context handoff. WordPress/WooCommerce Action Scheduler and a functioning cron runner are needed for receipt delivery. Inspect WooCommerce > Status > Scheduled Actions when a job waits or fails.

Online payment_complete events and successful gateway API refunds feed the receipt workflow. Cash on delivery, bank transfer and cheque need an authorized operator's explicit confirmation. Manual refunds are opt-in merchant assertions. These generic merchant receipts are server_reported; they do not independently prove gateway settlement.

Test each intended gateway and checkout type in a staging store: paid order, duplicate callback, refund, missing context and temporary delivery failure. Compare the plugin's accepted receipt, Scheduled Actions, Get started and revenue report. Read https://www.neutarget.com/integrations/woocommerce for the current compatibility and update notes.

8. Verify and hand over

Use CHECKLIST.md. Get started distinguishes the browser visit, selected event or payment receipt, checkout context and refund receipt. Browser analytics may be explored before outcome setup is finished. A fully refunded order still proves receipt delivery, although its net revenue is zero. Currencies remain separate unless an explicit dated FX process is configured. A session included on a receipt does not by itself establish campaign attribution; inspect the report.

Nothing arriving? Check the published website ID, duplicate tags, consent choice, tracker load, browser Network/CSP errors, selected event name, webhook signature, queued jobs and website subscription/lifecycle state. If verification is temporarily unavailable, retry before deciding the event was lost. Do not create fake financial events to turn a check green.

Give the workspace owner the accepted receipt ID, event name, observed timestamps, deployed build/theme version and remaining limitations. Keep private credentials out of that handoff. Continue with the feature activation guide for the other capabilities your team wants to use.