Headless WooCommerce

Developer Integration Guide

Integrating VERIFIED Crypto Checkout with a decoupled WooCommerce storefront

Version 1.0 | March 2026 | Audience: Frontend & Backend Developers

What this guide covers: This is the Headless WooCommerce Developer Integration Guide for VERIFIED Crypto Checkout. It is for development teams building a headless or decoupled WooCommerce storefront who want to integrate VERIFIED Crypto Checkout as a payment method. You will use the WooCommerce Store API to manage the cart and initiate checkout — the VERIFIED plugin handles the rest.

Plugin installation: This guide assumes the VERIFIED Crypto Checkout plugin was installed from one of the official distribution sources: VerifiedCryptoCheckout.com or the WordPress Plugin Repository.

How VERIFIED Handles Payments

Important — read this before writing any integration code: Your frontend never interacts directly with payment providers. You do not need to build payment session logic, initialize crypto wallets, or configure provider SDKs. The VERIFIED Crypto Checkout plugin handles all of this automatically.

Here is what happens the moment your checkout POST reaches WooCommerce:

1. **Plugin intercepts the checkout**
   The VERIFIED plugin registers as a WooCommerce payment gateway. When your checkout request specifies verified_crypto_checkout as the payment method, the plugin takes over — your code is done at this point.
2. **Session parameters are assembled**
   The plugin constructs a signed checkout session using the merchant configuration. No frontend involvement is required for this step.
3. **Session metadata is embedded automatically**
   The plugin embeds all required session parameters automatically. Developers do not need to initialize payment sessions, manage provider SDKs, or pass special routing parameters from the frontend.
4. **A redirect URL is returned**
   The plugin generates a signed checkout session and returns a redirect URL pointing directly to the VERIFIED hosted checkout environment (pay.verifiedcryptocheckout.com). Your only job is to send the customer's browser to this URL immediately.
5. **Hosted checkout and payment**
   The customer arrives at the VERIFIED hosted checkout environment (pay.verifiedcryptocheckout.com), selects a payment provider, completes their card payment, and USDC settles on-chain to the merchant wallet.
6. **Order status updates automatically**
   A server-side callback from VERIFIED updates the WooCommerce order status. No polling or frontend action required.

The takeaway for developers: Your integration has exactly two responsibilities — (1) call the WooCommerce Store API in sequence to build the cart and submit checkout, and (2) redirect the customer to the URL the plugin returns. Everything else — payment routing, provider selection, session handling, and settlement — is handled by VERIFIED infrastructure.

Developer philosophy: Developers integrating the VERIFIED Crypto Checkout plugin do not need to interact with payment providers, routing infrastructure, or blockchain settlement logic. The WooCommerce plugin manages the entire payment session and checkout flow automatically.

1. Architecture Overview

In a headless setup, your frontend (React, Next.js, Vue, or similar) communicates with WooCommerce exclusively through REST API calls. The VERIFIED Crypto Checkout plugin operates as a standard WooCommerce payment gateway on the backend — your frontend simply

selects it as the payment method at checkout, and the plugin returns a redirect URL that sends the customer to the hosted payment flow.

This means your frontend never needs to handle crypto logic, wallet addresses, or payment provider SDKs. VERIFIED takes care of all of that behind the scenes.

How the checkout works: The VERIFIED Crypto Checkout plugin operates as a standard WooCommerce payment gateway. When a checkout request is submitted, the plugin generates a signed checkout session and returns a redirect URL that sends the customer directly to the VERIFIED hosted checkout environment at pay.verifiedcryptocheckout.com — where they select a payment provider and complete their card payment.

1	**Headless Frontend** Your React, Next.js, Vue, or custom frontend. Manages the shopping experience, calls the WooCommerce Store API, and handles the redirect to the payment page.
2	**WooCommerce Store API** The standard WooCommerce headless API layer. Handles cart management, customer data, and order creation. Your frontend speaks only to this layer.
3	**VERIFIED Crypto Checkout Plugin** Installed on the WooCommerce backend. Registers as a standard payment gateway, intercepts the checkout request, generates a signed checkout session, and returns a redirect URL pointing to the VERIFIED hosted checkout environment.
4	**Hosted Checkout (pay.verifiedcryptocheckout.com)** The secure VERIFIED hosted checkout environment where the customer selects a payment provider and completes their card payment. No frontend SDK or integration required.
5	**Payment Provider** A licensed third-party provider (e.g. MoonPay, Stripe, Ramp) handles card processing and converts the payment to USDC, which is sent on-chain to the merchant wallet.
6	**USDC Settlement** Payment settles in USDC on the Polygon blockchain, sent directly to the merchant wallet address configured in the plugin settings.
7	**Webhook / Order Callback** Once on-chain settlement is confirmed, a callback updates the WooCommerce order status to Processing or Completed automatically. No frontend polling required.

2. Prerequisites

Before starting your integration, make sure the following are in place on both the WooCommerce backend and your frontend environment. Missing any of these is the most common cause of checkout failures.

Requirement	Details & Notes
WooCommerce installed	WooCommerce must be installed, activated, and configured with at least one product and a valid store setup.
VERIFIED Crypto Checkout plugin	Install the VERIFIED Crypto Checkout plugin directly from: • VerifiedCryptoCheckout.com • WordPress Plugin Repository (if available) **After installation:** • Activate the plugin • Configure the merchant USDC wallet address in WooCommerce → Settings → Payments • Perform a test transaction using standard WooCommerce checkout before attempting a headless integration
WooCommerce Store API enabled	The Store API (wc/store/v1) is included with WooCommerce 6.9+. No additional plugin is required. Confirm it is accessible at /wp-json/wc/store/v1/cart before proceeding.
Store API publicly reachable	The Store API must be reachable without interactive WordPress login. If your WooCommerce site is behind Basic Auth, a WAF, or a reverse proxy, ensure that /wp-json/wc/store/v1/* is whitelisted for unauthenticated access. Many staging environments restrict this by default. Session cookies must also be permitted cross-origin if your frontend is on a different domain — see Section 5 for CORS and SameSite configuration details.
HTTPS required	Both your WooCommerce backend and your headless frontend must be served over HTTPS. The VERIFIED hosted checkout requires a secure origin — HTTP will not work.
Cookie persistence on frontend	Your frontend must preserve and resend WooCommerce session cookies (wp_woocommerce_session_*) on every request. Without this, the cart resets between calls and checkout will fail. See Section 5 for details.
pay.verifiedcryptocheckout.com accessible	The browser must be able to redirect to the VERIFIED hosted checkout domain. Ensure pay.verifiedcryptocheckout.com is not blocked by CSP headers, firewall rules, or proxy configuration.
Inbound callbacks not blocked	WooCommerce must be able to receive inbound HTTP callbacks from the VERIFIED infrastructure to update order statuses. Ensure your hosting does not block inbound POST requests from external IPs.

3. Payment Method Identifier

When submitting the checkout request via the Store API, you must reference the VERIFIED Crypto Checkout gateway using its exact payment method identifier. Using a different string will result in WooCommerce returning a payment method not available error.

```
verified_crypto_checkout
```

Where to use this: This identifier is passed as the payment_method field in your POST /wc/store/v1/checkout request body. It must match exactly — including underscores and lowercase. See Step 4 of the checkout flow in Section 4.

4. Step-by-Step Checkout Flow

The following steps represent a complete headless checkout integration from cart creation through payment redirect. Each step corresponds to one Store API call. Implement them in order — skipping or reordering steps will cause session or cart errors.

1

Create or Verify the Cart Session

Before adding any items, ensure a WooCommerce cart session exists. Most Store API implementations create the session implicitly on the first cart request, but it is good practice to explicitly initialize it.

`POST` `/wc/store/v1/cart`

Request — no body required for cart initialization

```
POST /wc/store/v1/cart
Host: yourstore.com
Content-Type: application/json
```

Session cookie: The response from this call will set a wp_woocommerce_session_* cookie. You must capture and resend this cookie on every subsequent Store API request in the same checkout flow. If you are using fetch() or axios in the browser, set credentials: "include". If you are making server-side calls, capture the Set-Cookie header and forward it manually.

2 Add Items to Cart

Add the customer's selected product(s) to the cart. Repeat this call for each distinct product. The quantity field accepts integers only.

`POST` `/wc/store/v1/cart/add-item`

Request body

```
{
  "id": 123,
  "quantity": 1
}
```

Product ID: The id field must be the WooCommerce product ID (integer), not a slug or SKU. For variable products, use the variation ID. You can retrieve product IDs from the WooCommerce Products REST API at /wp-json/wc/v3/products.

3 Set Customer Billing Data

WooCommerce requires a billing address to process a checkout, even for digital products. At minimum, provide first_name, last_name, email, and country. The VERIFIED plugin uses the email for the payment provider's KYC flow — providing it here avoids the customer having to re-enter it at the provider's checkout page.

`POST` `/wc/store/v1/cart/update-customer`

Request body — minimum required fields

```
{
  "billing_address": {
    "first_name": "Jane",
    "last_name": "Smith",
    "email": "jane@example.com",
    "country": "US"
  }
}
```

Additional fields: You can also pass address_1, city, state, and postcode if your store requires them for tax calculation or shipping. For digital-only products, the four fields shown above are typically sufficient. Passing a complete address improves KYC match rates with some payment providers.

If Your Store Ships Physical Items

Digital products only require the billing fields above. If your store sells physical goods, WooCommerce will also need a shipping address and a selected shipping rate before the checkout request will succeed. Without these, the checkout POST will return a validation error.

Extended request body — physical goods

```
{
  "billing_address": {
    "first_name": "Jane",
    "last_name": "Smith",
    "email": "jane@example.com",
    "address_1": "123 Main St",
    "city": "Austin",
    "state": "TX",
    "postcode": "78701",
    "country": "US"
  },
  "shipping_address": {
    "first_name": "Jane",
    "last_name": "Smith",
    "address_1": "123 Main St",
    "city": "Austin",
    "state": "TX",
    "postcode": "78701",
    "country": "US"
  }
}
```

Shipping rate selection: After setting the shipping address, retrieve available shipping rates with GET /wc/store/v1/cart (the response includes a shipping_rates array). Your frontend must then select a rate with POST /wc/store/v1/cart/select-shipping-rate before the checkout POST will succeed for shippable orders. If you skip this step, WooCommerce will return a 'no shipping rate selected' error.

Tax calculation: WooCommerce calculates taxes automatically based on the shipping or billing address and your store's tax settings. Taxes are applied to the cart totals returned in the GET /wc/store/v1/cart response after the address is set. The amount the customer pays at the VERIFIED hosted checkout includes any WooCommerce-calculated taxes — no additional tax handling is required in your frontend.

4 Submit the Checkout Request

This is the key step. Submitting the checkout request with payment_method set to verified_crypto_checkout triggers the VERIFIED plugin to create a payment session and return a redirect URL. Your frontend must then immediately redirect the customer to this URL.

`POST` `/wc/store/v1/checkout`

Request body

```
{
  "payment_method": "verified_crypto_checkout"
}
```

Successful response

```
{
  "order_id": 1234,
  "status": "pending",
  "payment_result": {
    "redirect_url": "https://pay.verifiedcryptocheckout.com/session/sess_..."
  }
}
```

What the redirect_url does: The redirect_url is generated directly by the VERIFIED Crypto Checkout plugin. It points to the VERIFIED hosted checkout environment where the customer completes payment. Flow: Storefront → WooCommerce checkout endpoint → VERIFIED plugin generates session → pay.verifiedcryptocheckout.com (hosted checkout page) → payment provider → USDC settlement → WooCommerce callback.

⚠️ **The redirect_url is time-sensitive:** Do not cache or store the redirect_url. Redirect the customer immediately after receiving it. The URL contains signed session parameters that expire — presenting a stale URL will result in the customer seeing a session expired or invalid payment error at the checkout page.

5 Redirect the Customer

Once you have the redirect_url, send the customer's browser to it immediately. Do not attempt to load this URL in an iframe — redirect the full browser window.

JavaScript — browser redirect

```
// Use the redirect_url exactly as returned — do not modify it
```

```
window.location.href = response.payment_result.redirect_url;
```

Next.js / server-side redirect (if handling checkout server-side)

```
// In a Next.js API route or server action:
res.redirect(302, response.payment_result.redirect_url);

// Or in App Router:
redirect(response.payment_result.redirect_url);
```

⚠️ **Never iframe the checkout:** The VERIFIED hosted checkout and the payment provider pages must open in the full browser window. Iframing will break the payment provider's security requirements, may prevent cookies from being set correctly, and will result in checkout failures. Always use a full-page redirect.

Use the URL exactly as returned: The redirect_url must be used exactly as returned by WooCommerce. Do not modify the URL or its query parameters. The URL contains signed session parameters used by the VERIFIED routing infrastructure — any modification will invalidate the session.

5. Session Management & Cookie Persistence

This is the most common source of integration issues in headless WooCommerce setups. WooCommerce uses server-side sessions to track cart state. The session is identified by a cookie named wp_woocommerce_session_* (the suffix is a hash of the site URL). Your frontend must treat this cookie like a user authentication token — capture it on the first request and send it back on every subsequent request in the checkout flow.

Browser-Based Frontends (SPA / CSR)

If your frontend runs in the browser and calls the Store API directly, use the credentials option in your HTTP client:

fetch() — include credentials

```
const response = await
fetch('https://yourstore.com/wp-json/wc/store/v1/cart/add-item', {
  method: 'POST',
  credentials: 'include',   // Required: sends and receives cookies
  headers: { 'Content-Type': 'application/json' },
```

```
  body: JSON.stringify({ id: 123, quantity: 1 })
});
```

Cross-origin note: If your frontend domain differs from your WooCommerce domain (e.g. shop.mysite.com vs mysite.com/wp), you must configure WooCommerce to allow cross-origin requests and set the appropriate CORS headers. Additionally, the SameSite cookie attribute on the WooCommerce session cookie may need to be set to None with Secure to allow cross-origin cookie transmission. Check with your hosting provider if you encounter cookie issues across subdomains.

Server-Side / SSR Frontends (Next.js, Nuxt, etc.)

If your frontend makes Store API calls from the server (e.g. in Next.js API routes, server actions, or middleware), you must manually forward the WooCommerce session cookie between the browser and your server, and from your server to WooCommerce.

Next.js API route — manual cookie forwarding

```
// Receive the cookie from the browser request
const cookie = req.headers.cookie;

// Forward it to WooCommerce
const wcResponse = await
fetch('https://yourstore.com/wp-json/wc/store/v1/checkout', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Cookie': cookie,          // Forward the session cookie
  },
  body: JSON.stringify({ payment_method: 'verified_crypto_checkout' })
});

// Forward the Set-Cookie header back to the browser if it changes
const setCookie = wcResponse.headers.get('set-cookie');
if (setCookie) res.setHeader('Set-Cookie', setCookie);
```

What happens if cookies are not persisted: WooCommerce will treat each request as a new anonymous session. The cart will appear empty, the customer data will not be attached, and the checkout request will fail with a cart is empty error or return a 400 response. If you are seeing these errors, cookie forwarding is almost always the cause.

6. Order Lifecycle

Once the customer is redirected to the VERIFIED checkout and completes their card payment, the order status in WooCommerce updates automatically via a server-side callback. Your frontend does not need to poll for status — but understanding the lifecycle helps you build the right post-checkout experience.

Status	What It Means
Pending	The checkout request was submitted and the order was created in WooCommerce. The customer has been redirected to the payment page but has not yet completed payment. This is the expected status immediately after your checkout POST returns.
On Hold	Payment has been initiated by the customer at the provider's checkout but on-chain settlement has not yet confirmed. Some provider flows will set this intermediate status.
Processing	On-chain USDC settlement is confirmed. The payment is complete. This is the primary success status — trigger any fulfillment actions here.
Completed	Order has been fulfilled. Typically set manually by the merchant or automatically by a fulfillment plugin after Processing.
Failed	The payment was not completed. This can happen if the customer abandoned the checkout, the provider declined the card, or the session expired. A new order and checkout session should be created if the customer wants to retry.
Cancelled	Merchant or system cancelled the order. Not typically triggered by the payment flow.

Post-payment redirect behaviour: Most payment provider flows do not redirect customers back to your storefront automatically after payment is complete. Treat this as the default: design your post-checkout UX to show a 'processing payment' or 'order received' state immediately after the redirect, without expecting the customer to return to the site. WooCommerce order status updates server-side via callback regardless of whether the customer ever returns. You can also build a lightweight status page that polls GET /wc/store/v1/order/{order_id} to detect when the status moves to Processing and show a live confirmation.

7. Standard Plugin Installation

The VERIFIED Crypto Checkout plugin is the same plugin for all integration types — classic WooCommerce checkout, WooCommerce Blocks, and headless storefronts. No special build is required for a headless integration.

> **Install from an official source:** VerifiedCryptoCheckout.com or WordPress Plugin Repository

- The same plugin works for classic WooCommerce checkout, WooCommerce Blocks checkout, and headless WooCommerce storefronts.
- No special plugin build is required for headless integrations.
- After installation, activate the plugin and configure your merchant USDC wallet address in WooCommerce → Settings → Payments.
- Perform a test transaction using the standard WooCommerce checkout before attempting the headless integration.

8. Troubleshooting

The issues below cover the most common integration problems encountered in headless WooCommerce setups. Work through these before contacting support — most can be resolved without any changes to the VERIFIED plugin.

Symptom	Likely Cause	Fix
Cart returns empty after add-item	Session cookie not being persisted between requests	Add credentials: 'include' to all fetch calls (browser) or manually forward the wp_woocommerce_session_* cookie on server-side requests. Verify the cookie is present in the request headers using your browser's DevTools Network tab.
400 error on checkout POST	Cart is empty or customer data not set	Confirm the session cookie is present. Confirm the add-item call succeeded (check the response — it should return the updated cart). Confirm update-customer was called before checkout.
payment_result.redirect_url missing in response	VERIFIED plugin not activated or gateway not enabled	Go to WooCommerce → Settings → Payments and confirm the VERIFIED Crypto Checkout gateway is active. Check that the wallet address is configured. Verify the plugin is activated in WooCommerce → Plugins.

404 or session expired at checkout page	Stale redirect_url or URL was modified	Redirect immediately after receiving the URL — do not store or reuse it. Do not modify any query parameters in the redirect_url. The URL is signed and any changes will invalidate it.
Order stuck in Pending forever	Inbound callback from VERIFIED is being blocked by hosting, WAF, or security plugin	Run through these checks in order: (1) In Wordfence → Firewall → All Firewall Options, disable 'Block requests from IP addresses that access WordPress in ways that indicate they are malicious' or add the VERIFIED callback IP to the allowlist. (2) In your WAF (Cloudflare, Sucuri, etc.), ensure POST requests to /wc-api/ and /wp-json/wc/ are not blocked for external IPs. (3) SSH into the server and check error logs (e.g. /var/log/nginx/error.log or Apache error.log) for 403 or 503 responses at the time a test payment completes. (4) Temporarily disable all security plugins and retry a test payment — if the order updates, re-enable plugins one at a time to identify the blocker.
CORS error on Store API call	Cross-origin requests not configured on WooCommerce	Add your frontend domain to the allowed origins in WooCommerce or via a CORS plugin. If using a proxy (e.g. Next.js API route), route Store API calls through your server to avoid cross-origin issues entirely.
SameSite cookie warning in browser console	WooCommerce session cookie not configured for cross-origin	This typically requires server-level configuration to add SameSite=None; Secure to the session cookie. Contact your hosting provider or configure it via a cookie-management plugin.

9. Integration Checklist

Run through this checklist before going live. Every item here represents a real failure mode discovered in headless integrations.

☐	VERIFIED Crypto Checkout plugin installed, activated, and wallet address configured
☐	WooCommerce Store API accessible at /wp-json/wc/store/v1/cart
☐	Store API reachable without WordPress login (Basic Auth, WAF, or reverse proxy not

	blocking /wp-json/wc/store/v1/*)
☐	HTTPS enabled on both WooCommerce backend and headless frontend
☐	Session cookie (wp_woocommerce_session_*) persisted and forwarded on all Store API calls
☐	Cart add-item, update-customer, and checkout calls tested end-to-end in sequence
☐	payment_method: 'verified_crypto_checkout' submitted exactly in checkout POST
☐	Redirect to redirect_url performed immediately after receiving it (no caching or modification)
☐	Full-page redirect used — not an iframe or fetch of the checkout URL
☐	VERIFIED Crypto Checkout plugin installed from VerifiedCryptoCheckout.com or the WordPress Plugin Repository
☐	Inbound callbacks to WooCommerce not blocked by WAF or security plugin
☐	pay.verifiedcryptocheckout.com accessible from the browser — not blocked by firewall, CSP headers, or proxy
☐	Order status checked after a test payment — confirm it moves from Pending to Processing
☐	Post-checkout UX implemented (confirmation page, no redirect-back expectation)
☐	Order confirmation email tested and wording reviewed for accuracy

10. Quick Reference

Store API Endpoints Used in This Integration

Endpoint	Purpose
`POST /wc/store/v1/cart`	Initialize or retrieve the cart session.
`POST /wc/store/v1/cart/add-item`	Add a product to the cart by WooCommerce product ID.
`POST /wc/store/v1/cart/update-customer`	Set billing address and customer email on the cart.
`POST`	Submit checkout. Returns order_id and redirect_url.

`/wc/store/v1/checkout`	

Key Values

Item	Value
Payment method identifier	verified_crypto_checkout
Hosted checkout domain	pay.verifiedcryptocheckout.com
Plugin source	VerifiedCryptoCheckout.com or WordPress Plugin Repository
Works with	Classic WooCommerce checkout, WooCommerce Blocks, and headless storefronts
Session cookie name	wp_woocommerce_session_* (suffix varies by site)

Plugin Installation Source

The VERIFIED Crypto Checkout plugin should be installed directly from an official VERIFIED source. The same plugin works for all WooCommerce integration types — no special build is required for headless setups.

Install from: VerifiedCryptoCheckout.com or WordPress Plugin Repository

✅ **Compatible with all WooCommerce setups:** The standard VERIFIED Crypto Checkout plugin works with classic WooCommerce checkout, WooCommerce Blocks, and headless storefronts using the Store API. No different version or build is required for this integration.

Need help? Email us at: CryptoCheckout@VerifiedCreditCardProcessing.com | Visit: verifiedcryptocheckout.com