ChatGPT  
  
Short answer: use Converge’s **Hosted Payments “Payment Modal” (aka Checkout.js / PayWithConverge.js)**. It keeps card data out of your servers (much lighter PCI scope), supports **vault token** creation in the same flow, and gives you full-page control with an iframe modal. Converge requires your page to be **HTTPS** and your domain **whitelisted** before their script will run—use a quick tunnel (e.g., ngrok) if you don’t have a cert yet. ([Amazon Web Services, Inc.](https://s3.us-east-2.amazonaws.com/assets.dps.mn.gov/s3fs-public/dvs-converge-fees-guide-2019-1.pdf?utm_source=chatgpt.com), [WooCommerce](https://woocommerce.com/document/elavon-vm-payment-gateway/?utm_source=chatgpt.com))

Below is a complete, production-oriented starter that will:

1. create a **session token** on your server, 2) open the Converge modal in React, 3) process a **real** card, and 4) capture the returned **vault token** for future monthly billing.

**Why this method (and why not the others)**

* **Hosted Payments – Payment Modal / Checkout.js (recommended)**
  + Card data is posted **directly** from the browser to Converge; your app only handles a short-lived **session token** and the **vault token** that comes back. Much smaller PCI scope than handling card data yourself. Requires HTTPS + Allowed Origin and an API user permitted to request session tokens. ([Amazon Web Services, Inc.](https://s3.us-east-2.amazonaws.com/assets.dps.mn.gov/s3fs-public/dvs-converge-fees-guide-2019-1.pdf?utm_source=chatgpt.com), [WooCommerce](https://woocommerce.com/document/elavon-vm-payment-gateway/?utm_source=chatgpt.com))
* **XML / Name-Value (processxml.do)**
  + Works, but your server must **touch raw PAN data** if you post the card to your backend—this pushes you into heavy PCI scope. Use this only for **subsequent** charges using the **vault token** (no card data). ([Elavon Developer](https://developer.elavon.com/products/xml-api/v1/non-elavon-certified-device?utm_source=chatgpt.com" \o "Non-Elavon certified device - XML API))
* **Full Hosted Payment Page (redirect)**
  + Also SAQ-light, but you lose the polished in-page UX the modal provides.
* **Lightbox vs Checkout.js naming**
  + Converge’s docs use both terms; the script you include is **PayWithConverge.js** and the flow hinges on a server-created **transaction/session token**. ([Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407206952087-Developer-Portal-and-Useful-Links?utm_source=chatgpt.com" \o "Developer Portal and Useful Links))

**Live cards without HTTPS?** Not with Hosted Payments—the script refuses to load on HTTP pages. Use a temporary **HTTPS** public URL (e.g., ngrok http 3000) and whitelist that origin in Converge to run live transactions now. ([Amazon Web Services, Inc.](https://s3.us-east-2.amazonaws.com/assets.dps.mn.gov/s3fs-public/dvs-converge-fees-guide-2019-1.pdf?utm_source=chatgpt.com))

**One-time Converge setup (account/portal)**

1. **Enable Hosted Payments (Payment Modal)** and **Tokenization/Card Manager** on your terminal. If you want tokens usable across multiple terminals, ask for a **shared token vault**. ([CivicPlus](https://civicplus.us.document360.io/docs/set-up-converge-for-payments?utm_source=chatgpt.com" \o "Set Up Converge for Payments), [CivicPlus](https://www.civicplus.help/docs/converge-saved-payment-methods?utm_source=chatgpt.com))
2. Create an **API user** allowed to post **session token requests**; note its **User ID** and **PIN**. ([WooCommerce](https://woocommerce.com/document/elavon-vm-payment-gateway/?utm_source=chatgpt.com" \o "WooCommerce Elavon Converge Documentation))
3. Add your site (or ngrok) to **Allowed Origins** / IP allow lists per Converge docs. ([WooCommerce](https://woocommerce.com/document/elavon-vm-payment-gateway/?utm_source=chatgpt.com" \o "WooCommerce Elavon Converge Documentation), [CivicPlus](https://civicplus.us.document360.io/docs/set-up-converge-for-payments?utm_source=chatgpt.com))
4. Confirm Converge’s script requirements: **HTTPS**, **valid certificate**, **TLS 1.2+**. ([Amazon Web Services, Inc.](https://s3.us-east-2.amazonaws.com/assets.dps.mn.gov/s3fs-public/dvs-converge-fees-guide-2019-1.pdf?utm_source=chatgpt.com))

**Backend (Node/Express/Axios)**

**What this does:** your server exchanges your **merchant\_id / user\_id / pin** for a **session token** (aka ssl\_txn\_auth\_token) by calling Converge’s /hosted-payments/transaction\_token endpoint. Include ssl\_get\_token=Y and ssl\_add\_token=Y so Converge returns a **vault token** after the modal completes. ([Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407206952087-Developer-Portal-and-Useful-Links?utm_source=chatgpt.com" \o "Developer Portal and Useful Links), [Elavon Developer](https://developer.elavon.com/products/checkout-js/v1/transaction-types?utm_source=chatgpt.com))

**1) Install deps**

npm i express axios cors dotenv qs xml2js

**2) .env (temporary hard-coded creds; later pull from DB)**

CONVERGE\_MERCHANT\_ID=YOUR\_ACCOUNT\_ID

CONVERGE\_USER\_ID=YOUR\_API\_USER

CONVERGE\_PIN=YOUR\_API\_PIN

# Use the real (production) host for live transactions

CONVERGE\_BASE=https://api.convergepay.com

**3) app.js (server)**

// app.js

require('dotenv').config();

const express = require('express');

const axios = require('axios');

const cors = require('cors');

const qs = require('qs');

const xml2js = require('xml2js');

const app = express();

app.use(express.json());

app.use(cors());

const CONVERGE\_BASE = process.env.CONVERGE\_BASE || 'https://api.convergepay.com';

// 1) Create a Checkout.js / Lightbox session token

app.post('/api/converge/session-token', async (req, res) => {

try {

const { amount, orderId, customerId, addToken = true } = req.body;

const form = {

ssl\_merchant\_id: process.env.CONVERGE\_MERCHANT\_ID,

ssl\_user\_id: process.env.CONVERGE\_USER\_ID,

ssl\_pin: process.env.CONVERGE\_PIN,

// This is the transaction you're about to perform in the modal

ssl\_transaction\_type: 'ccsale',

ssl\_amount: amount,

ssl\_currency\_code: 'USD',

ssl\_invoice\_number: orderId,

// Ask Converge to return/store a vault token

ssl\_get\_token: addToken ? 'Y' : 'N',

ssl\_add\_token: addToken ? 'Y' : 'N',

// Optional association fields

ssl\_customer\_id: customerId,

};

const url = `${CONVERGE\_BASE}/hosted-payments/transaction\_token`; // returns ssl\_txn\_auth\_token

const { data } = await axios.post(url, qs.stringify(form), {

headers: { 'Content-Type': 'application/x-www-form-urlencoded' },

timeout: 15000,

});

// Typical response includes { ssl\_txn\_auth\_token: '...' }

res.json(data);

} catch (err) {

console.error(err?.response?.data || err.message);

res.status(500).json({ error: 'Failed to get session token' });

}

});

// 2) Charge later with a saved vault token (server-to-server, no PAN)

app.post('/api/converge/charge-token', async (req, res) => {

try {

const { token, amount, orderId } = req.body;

const xml = `

<txn>

<ssl\_merchant\_id>${process.env.CONVERGE\_MERCHANT\_ID}</ssl\_merchant\_id>

<ssl\_user\_id>${process.env.CONVERGE\_USER\_ID}</ssl\_user\_id>

<ssl\_pin>${process.env.CONVERGE\_PIN}</ssl\_pin>

<ssl\_transaction\_type>ccsale</ssl\_transaction\_type>

<ssl\_amount>${amount}</ssl\_amount>

<ssl\_token>${token}</ssl\_token>

<ssl\_invoice\_number>${orderId}</ssl\_invoice\_number>

</txn>`.trim();

const url = `${CONVERGE\_BASE}/VirtualMerchant/processxml.do`;

const { data } = await axios.post(url, xml, {

headers: { 'Content-Type': 'text/xml' },

timeout: 15000,

});

const parsed = await xml2js.parseStringPromise(data, { explicitArray: false });

res.json(parsed);

} catch (err) {

console.error(err?.response?.data || err.message);

res.status(500).json({ error: 'Charge failed' });

}

});

app.listen(3000, () => console.log('Server up on http://localhost:3000'));

* **Why these endpoints?**
  + /hosted-payments/transaction\_token is the documented way to request a **session token** for Checkout.js/Lightbox. Your credentials live only on the server. ([Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407206952087-Developer-Portal-and-Useful-Links?utm_source=chatgpt.com" \o "Developer Portal and Useful Links))
  + For later monthly charges, post to processxml.do with ssl\_token—that’s how you run a sale against the stored vault token. ([Elavon Developer](https://developer.elavon.com/products/xml-api/v1/non-elavon-certified-device?utm_source=chatgpt.com" \o "Non-Elavon certified device - XML API))

**Frontend (React) – open the modal and capture the vault token**

**What this does:** loads Converge’s **PayWithConverge.js** script, asks your backend for the **session token**, opens the modal, and reads the result. On success, Converge returns fields including **ssl\_token** for future billing when you request tokenization. ([Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407206952087-Developer-Portal-and-Useful-Links?utm_source=chatgpt.com" \o "Developer Portal and Useful Links), [Elavon Developer](https://developer.elavon.com/products/checkout-js/v1/transaction-types?utm_source=chatgpt.com))

// ConvergePayButton.jsx

import { useEffect, useState } from 'react';

import axios from 'axios';

export default function ConvergePayButton({ amount, orderId, customerId }) {

const [ready, setReady] = useState(false);

useEffect(() => {

// Load Converge modal script (PRODUCTION)

const s = document.createElement('script');

s.src = 'https://api.convergepay.com/hosted-payments/PayWithConverge.js';

s.onload = () => setReady(true);

s.onerror = () => console.error('Failed to load Converge JS');

document.head.appendChild(s);

return () => document.head.removeChild(s);

}, []);

const handlePay = async () => {

// 1) Get session token from your server

const { data } = await axios.post('/api/converge/session-token', {

amount,

orderId,

customerId,

addToken: true, // request vault token

});

const sessionToken = data.ssl\_txn\_auth\_token || data?.data?.ssl\_txn\_auth\_token;

if (!sessionToken) throw new Error('No Converge session token in response');

// 2) Open the modal

const fields = { ssl\_txn\_auth\_token: sessionToken };

// PayWithConverge is injected by the script

window.PayWithConverge.open(fields, async (result) => {

// NOTE: log result to see your exact payload with your terminal config

console.log('Converge result:', result);

if (result?.error || result?.errorText) {

alert(result?.errorText || 'Payment failed.');

return;

}

// 3) Persist the returned vault token (if present)

const vaultToken = result?.ssl\_token || result?.data?.ssl\_token;

const txnId = result?.ssl\_transaction\_id || result?.data?.ssl\_transaction\_id;

if (vaultToken) {

await axios.post('/api/store-vault-token', { customerId, vaultToken }); // implement on your backend

}

alert(`Approved. Transaction ID: ${txnId}`);

});

};

return (

<button disabled={!ready} onClick={handlePay}>

Pay ${amount}

</button>

);

}

* Script URL for the modal: **https://api.convergepay.com/hosted-payments/PayWithConverge.js** (use the demo host if you switch to demo). ([Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407206952087-Developer-Portal-and-Useful-Links?utm_source=chatgpt.com" \o "Developer Portal and Useful Links))
* The backend call returns the **session token** (often ssl\_txn\_auth\_token) used to open the modal. ([Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407206952087-Developer-Portal-and-Useful-Links?utm_source=chatgpt.com" \o "Developer Portal and Useful Links))
* Include ssl\_get\_token=Y and ssl\_add\_token=Y when you request the session token so Converge returns **ssl\_token** after a successful charge. ([Elavon Developer](https://developer.elavon.com/products/checkout-js/v1/transaction-types?utm_source=chatgpt.com" \o "CheckoutJs | Payment types))

**Running repeat monthly billing**

When it’s time to bill again, call your **server** with the saved vaultToken and amount. The server uses the **XML API** with ssl\_token (no PAN) to run a ccsale:

// Example: POST /api/converge/charge-token

// body: { token: 'the-ssl-token', amount: '49.00', orderId: 'INV-1002' }

That call maps to Converge’s processxml endpoint and charges the stored card. ([Elavon Developer](https://developer.elavon.com/products/xml-api/v1/non-elavon-certified-device?utm_source=chatgpt.com" \o "Non-Elavon certified device - XML API))

**Important gotchas (production)**

* **HTTPS is mandatory** for any page that loads the Converge script; your domain must be allowed in Converge. (Use ngrok/Cloudflare Tunnel if you don’t have a cert yet.) ([Amazon Web Services, Inc.](https://s3.us-east-2.amazonaws.com/assets.dps.mn.gov/s3fs-public/dvs-converge-fees-guide-2019-1.pdf?utm_source=chatgpt.com))
* **Don’t put credentials client-side.** Converge explicitly instructs keeping merchant credentials on the server. ([Elavon Developer](https://developer-uat.elavon.com/products/checkout-js/v1/api-reference?utm_source=chatgpt.com" \o "Checkout JS | API reference))
* **Enable tokenization** on your Converge terminal (you’ll usually see a **Customers** tab when it’s on). If you have multiple terminals and want one shared set of tokens, ask support to enable a **shared token vault**. ([Benji Pays Support](https://support.benjipays.com/support/solutions/articles/150000019656-connect-to-elavon-converge?utm_source=chatgpt.com" \o "Connect to Elavon Converge), [CivicPlus](https://www.civicplus.help/docs/converge-saved-payment-methods?utm_source=chatgpt.com))
* **User permissions:** your API user must be authorized to post **session token requests**; if not, you’ll get 401/403 errors. ([WooCommerce](https://woocommerce.com/document/elavon-vm-payment-gateway/?utm_source=chatgpt.com" \o "WooCommerce Elavon Converge Documentation))

**Endpoints & parameters quick reference**

* **Session token (server → Converge):**  
  POST https://api.convergepay.com/hosted-payments/transaction\_token  
  Body (x-www-form-urlencoded): ssl\_merchant\_id, ssl\_user\_id, ssl\_pin, ssl\_transaction\_type=ccsale, ssl\_amount, ssl\_get\_token=Y, ssl\_add\_token=Y, … → returns ssl\_txn\_auth\_token. ([Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407206952087-Developer-Portal-and-Useful-Links?utm_source=chatgpt.com" \o "Developer Portal and Useful Links))
* **Open modal (browser):**  
  Include script .../PayWithConverge.js and call PayWithConverge.open({ ssl\_txn\_auth\_token }, callback). ([Elavon Developer](https://developer.elavon.com/products/lightbox-payment-modal/v1/take-a-payment?utm_source=chatgpt.com" \o "Lightbox payment modal | Take a payment))
* **Charge with vault token (server → Converge):**  
  POST https://api.convergepay.com/VirtualMerchant/processxml.do with XML containing ssl\_transaction\_type=ccsale, ssl\_token, ssl\_amount, credentials. ([Elavon Developer](https://developer.elavon.com/products/xml-api/v1/non-elavon-certified-device?utm_source=chatgpt.com" \o "Non-Elavon certified device - XML API))

If you want, I can adapt this to your exact React/Next.js layout (hooks, error banners, etc.), wire the **/api/store-vault-token** endpoint to your database, and add a simple monthly billing job (cron/queue) that hits /api/converge/charge-token on schedule.

Great—since your Delphi service will handle the recurring charge from the saved vault token, your new web app just needs to (1) run a **one-time live sale** and (2) receive/store the **vault token** Converge returns.

**First: quick answers**

* **What’s ngrok? Is it free?**  
  ngrok gives you a **public HTTPS URL** that tunnels traffic to your local dev server (e.g., http://localhost:3000). It’s perfect when a provider (like Converge Hosted Payments) requires HTTPS + a public origin during development. There’s a **free plan** with usage limits that’s fine for testing. ([Sendbird](https://sendbird.com/developer/tutorials/what-is-ngrok?utm_source=chatgpt.com" \o "What is ngrok? Using ngrok tunneling), [Ngrok](https://ngrok.com/pricing?utm_source=chatgpt.com))
* **Can I “fake” an HTTPS POST to Converge?**  
  No. You can’t trick the browser or Converge’s Hosted Payments script—**the page that loads their JS must be HTTPS** and your domain (origin) must be authorized in Converge. For server-to-server XML API calls you can be on plain HTTP locally (your server calls Converge over HTTPS), but the **Hosted Payments modal/script won’t run** without HTTPS + allowed origin. ([Elavon Developer](https://developer-uat.elavon.com/products/checkout-js/v1/api-reference?utm_source=chatgpt.com" \o "Checkout JS | API reference), [Stack Overflow](https://stackoverflow.com/questions/57693021/converge-lightbox-integration?utm_source=chatgpt.com))

**Why you still want Hosted Payments (Payment Modal)**

* Keeps raw card data out of your stack (lighter PCI scope) but still returns a **vault token** on approval.
* Requires: **HTTPS page**, **session token** from your backend, and your **origin whitelisted** in Converge. ([Elavon Developer](https://developer.elavon.com/products/converge/v1/checkoutjs?utm_source=chatgpt.com" \o "Checkout.js - Converge), [Elavon Developer](https://developer-uat.elavon.com/products/checkout-js/v1/api-reference?utm_source=chatgpt.com), [Stack Overflow](https://stackoverflow.com/questions/57693021/converge-lightbox-integration?utm_source=chatgpt.com))

**Option A (recommended for live testing today): Use ngrok for HTTPS**

1. **Install & sign in**
   * Create a free account at ngrok, download the agent, and add your **authtoken**.
   * ngrok config add-authtoken <YOUR\_TOKEN>

([Ngrok](https://ngrok.com/pricing?utm_source=chatgpt.com" \o "ngrok pricing | Flexible plans for production and development), [BrowserStack](https://www.browserstack.com/guide/what-is-ngrok?utm_source=chatgpt.com))

1. **Run your app locally**  
   Example: npm run dev (Next/React) or node app.js (Express) on **port 3000**.
2. **Start the tunnel**
3. ngrok http 3000

You’ll get a public [**https://something.ngrok-free.app**](https://something.ngrok-free.app/) URL. ([BrowserStack](https://www.browserstack.com/guide/what-is-ngrok?utm_source=chatgpt.com" \o "What is Ngrok and How Does It Work?))

1. **Tell Converge about your origin**  
   In the Converge portal, add your ngrok URL to **Authorized Referrers / Allowed Origins** (Converge support can help if the UI is locked). Also ensure **Hosted Payments / Checkout.js** is enabled. ([Stack Overflow](https://stackoverflow.com/questions/57693021/converge-lightbox-integration?utm_source=chatgpt.com), [ConvergePay Support](https://support.convergepay.com/s/?utm_source=chatgpt.com))
2. **Use the production script & session token flow**
   * Script: https://api.convergepay.com/hosted-payments/PayWithConverge.js
   * Backend: POST https://api.convergepay.com/hosted-payments/transaction\_token (with your merchant\_id, user\_id, pin, ssl\_transaction\_type=ccsale, ssl\_amount, **ssl\_get\_token=Y**, **ssl\_add\_token=Y**) to obtain ssl\_txn\_auth\_token.
   * Frontend: call PayWithConverge.open({ ssl\_txn\_auth\_token }, callback) and read **ssl\_token** from the result.  
     These are the documented Checkout.js steps. ([Elavon Developer](https://developer.elavon.com/products/converge/v1/checkoutjs?utm_source=chatgpt.com" \o "Checkout.js - Converge), [Elavon Developer](https://developer-uat.elavon.com/products/checkout-js/v1/api-reference?utm_source=chatgpt.com))

Free plan notes: you get a static domain and reasonable dev limits (HTTP requests/month, bandwidth). Plenty for integration/testing. ([Ngrok](https://ngrok.com/docs/pricing-limits/?utm_source=chatgpt.com" \o "Pricing and Limits | ngrok documentation))

**Option B (also free): Cloudflare Tunnel (no client install for viewers)**

If you already use Cloudflare (or don’t mind signing up), **Cloudflare Tunnel** (cloudflared) creates a public HTTPS URL to your local server:

1. Install and authenticate cloudflared.
2. Run:
3. cloudflared tunnel --url http://localhost:3000
4. Add the resulting HTTPS hostname to Converge **Authorized Referrers**. ([Cloudflare Docs](https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/do-more-with-tunnels/trycloudflare/?utm_source=chatgpt.com" \o "Quick Tunnels - Cloudflare Zero Trust))

**“Fake HTTPS” alternatives (why they usually don’t help)**

* **Self-signed or local certs** (mkcert, etc.) can put https://localhost on TLS, but **Converge’s script expects a publicly reachable, whitelisted origin**, and browsers distrust self-signed certs. So this won’t satisfy Hosted Payments’ requirements for a real transaction. Use ngrok or Cloudflare Tunnel instead. ([Stack Overflow](https://stackoverflow.com/questions/57693021/converge-lightbox-integration?utm_source=chatgpt.com))

**Minimal end-to-end steps you can follow now**

1. **Pick a tunnel**: ngrok (simplest) or Cloudflare Tunnel. Start it and get a public **HTTPS** URL. ([BrowserStack](https://www.browserstack.com/guide/what-is-ngrok?utm_source=chatgpt.com" \o "What is Ngrok and How Does It Work?), [Cloudflare Docs](https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/do-more-with-tunnels/trycloudflare/?utm_source=chatgpt.com))
2. **Whitelist that URL in Converge** under **HTTP Referrers / Allowed Origins**. Confirm Hosted Payments is enabled for your terminal. ([Stack Overflow](https://stackoverflow.com/questions/57693021/converge-lightbox-integration?utm_source=chatgpt.com))
3. **Backend (Express)**: create /api/converge/session-token that posts your **merchant\_id / user\_id / pin** to  
   https://api.convergepay.com/hosted-payments/transaction\_token with ssl\_get\_token=Y and ssl\_add\_token=Y. The response includes ssl\_txn\_auth\_token. (Use production host for live cards.) ([Elavon Developer](https://developer-uat.elavon.com/products/checkout-js/v1/api-reference?utm_source=chatgpt.com" \o "Checkout JS | API reference))
4. **Frontend (React)**:
   * Load https://api.convergepay.com/hosted-payments/PayWithConverge.js.
   * Click handler: POST to your /session-token, then PayWithConverge.open({ ssl\_txn\_auth\_token }, callback).
   * On success, store result.ssl\_token (the **vault token**) in your DB for your Delphi service. ([Elavon Developer](https://developer.elavon.com/products/converge/v1/checkoutjs?utm_source=chatgpt.com" \o "Checkout.js - Converge))
5. **(Optional) Test a token charge from your server only**: For verification—**without card data**—you can run a token sale via processxml.do using the returned ssl\_token. (Your site’s own HTTPS doesn’t matter for this server-to-server call, because your server posts to Converge over HTTPS.) ([Elavon Developer](https://developer.elavon.com/products/hosted-payment-page/v1/transaction-management?utm_source=chatgpt.com" \o "Transaction management - Hosted-Payment-Page))

If you want, I can drop in a ready-to-run Express route + React button using your temporary **ngrok** URL baked into the CORS/allowed-origin notes and highlight exactly where to paste your **merchant\_id / user\_id / pin** for live (non-demo) processing.

awesome — here’s a **copy-paste, runnable** starter that does exactly what you asked:

* spins up an **Express** backend with two endpoints:
  + POST /api/converge/session-token → asks Converge for a **session token** (for the modal)
  + POST /api/store-vault-token → placeholder to save the returned **vault token** (you’ll wire to DB)
* a tiny **React** button that loads Converge’s modal (Hosted Payments “Payment Modal”) and captures **ssl\_token** on approval
* directions for **ngrok** so you can run this **live** today over HTTPS (no demo mode)

per your plan, **Delphi** will handle recurring charges later using the saved token; we don’t do rebilling here.

**1) Folder layout**

your-project/

.env

package.json

app.js

client/

package.json

src/

ConvergePayButton.jsx

main.jsx

index.html

**2) Backend (Express)**

**file: .env**

put real, production creds here for live cards. (later, swap to DB lookup)

# Converge production credentials

CONVERGE\_MERCHANT\_ID=YOUR\_MERCHANT\_ID

CONVERGE\_USER\_ID=YOUR\_API\_USER

CONVERGE\_PIN=YOUR\_API\_PIN

# Converge production base URL

CONVERGE\_BASE=https://api.convergepay.com

# Allow your dev origin(s) – include your ngrok URL once you create it

CORS\_ALLOWED\_ORIGINS=http://localhost:5173,https://YOUR-NGROK-SUBDOMAIN.ngrok-free.app

PORT=3000

**file: package.json**

{

"name": "converge-express-backend",

"private": true,

"type": "module",

"scripts": {

"dev": "node app.js"

},

"dependencies": {

"axios": "^1.7.2",

"cors": "^2.8.5",

"dotenv": "^16.4.5",

"express": "^4.19.2",

"qs": "^6.12.0",

"xml2js": "^0.6.2"

}

}

**file: app.js**

import 'dotenv/config';

import express from 'express';

import cors from 'cors';

import axios from 'axios';

import qs from 'qs';

import { parseStringPromise } from 'xml2js';

const app = express();

app.use(express.json());

// ---- CORS: allow your React dev server + ngrok origin(s) ----

const allowed = (process.env.CORS\_ALLOWED\_ORIGINS || '').split(',').map(s => s.trim()).filter(Boolean);

app.use(cors({

origin: (origin, cb) => {

// allow same-origin tools (curl / Postman) and no-origin

if (!origin) return cb(null, true);

if (allowed.includes(origin)) return cb(null, true);

return cb(new Error(`Origin ${origin} not allowed`));

}

}));

const CONVERGE\_BASE = process.env.CONVERGE\_BASE || 'https://api.convergepay.com';

const REQ\_TIMEOUT = 15000;

// sanity check

if (!process.env.CONVERGE\_MERCHANT\_ID || !process.env.CONVERGE\_USER\_ID || !process.env.CONVERGE\_PIN) {

console.warn('[WARN] Missing Converge creds in .env');

}

// 1) Get a Hosted Payments session token (for the modal)

app.post('/api/converge/session-token', async (req, res) => {

try {

const { amount, orderId, customerId, addToken = true } = req.body;

if (!amount || !orderId) {

return res.status(400).json({ error: 'amount and orderId are required' });

}

const form = {

ssl\_merchant\_id: process.env.CONVERGE\_MERCHANT\_ID,

ssl\_user\_id: process.env.CONVERGE\_USER\_ID,

ssl\_pin: process.env.CONVERGE\_PIN,

// transaction to run in the modal

ssl\_transaction\_type: 'ccsale',

ssl\_amount: amount,

ssl\_currency\_code: 'USD',

ssl\_invoice\_number: orderId,

// ask Converge to return/store a vault token

ssl\_get\_token: addToken ? 'Y' : 'N',

ssl\_add\_token: addToken ? 'Y' : 'N',

// optional association

ssl\_customer\_id: customerId || undefined

};

const url = `${CONVERGE\_BASE}/hosted-payments/transaction\_token`;

const { data } = await axios.post(url, qs.stringify(form), {

headers: { 'Content-Type': 'application/x-www-form-urlencoded' },

timeout: REQ\_TIMEOUT

});

// Expecting { ssl\_txn\_auth\_token: '...' } on success

if (!data?.ssl\_txn\_auth\_token) {

console.error('[Converge token resp]', data);

return res.status(502).json({ error: 'No ssl\_txn\_auth\_token in response', raw: data });

}

res.json({ ssl\_txn\_auth\_token: data.ssl\_txn\_auth\_token });

} catch (err) {

const detail = err?.response?.data || err.message;

console.error('[session-token] error:', detail);

res.status(500).json({ error: 'Failed to get session token', detail });

}

});

// 2) Placeholder: store the returned vault token (you’ll wire to DB)

app.post('/api/store-vault-token', async (req, res) => {

const { customerId, vaultToken, transactionId, amount } = req.body || {};

if (!vaultToken) return res.status(400).json({ error: 'vaultToken is required' });

// TODO: replace with your DB insert/update

console.log('[STORE TOKEN]', { customerId, vaultToken, transactionId, amount, ts: new Date().toISOString() });

// Example response; Delphi will pick it up later from your DB

res.json({ ok: true });

});

// (Optional) quick verify: run a token sale server-side (no PAN) — not needed for your Delphi flow

app.post('/api/converge/charge-token', async (req, res) => {

try {

const { token, amount, orderId } = req.body;

if (!token || !amount) return res.status(400).json({ error: 'token and amount are required' });

const xml = `

<txn>

<ssl\_merchant\_id>${process.env.CONVERGE\_MERCHANT\_ID}</ssl\_merchant\_id>

<ssl\_user\_id>${process.env.CONVERGE\_USER\_ID}</ssl\_user\_id>

<ssl\_pin>${process.env.CONVERGE\_PIN}</ssl\_pin>

<ssl\_transaction\_type>ccsale</ssl\_transaction\_type>

<ssl\_amount>${amount}</ssl\_amount>

<ssl\_token>${token}</ssl\_token>

${orderId ? `<ssl\_invoice\_number>${orderId}</ssl\_invoice\_number>` : ''}

</txn>

`.trim();

const url = `${CONVERGE\_BASE}/VirtualMerchant/processxml.do`;

const { data } = await axios.post(url, xml, {

headers: { 'Content-Type': 'text/xml' },

timeout: REQ\_TIMEOUT

});

const parsed = await parseStringPromise(data, { explicitArray: false });

res.json(parsed);

} catch (err) {

const detail = err?.response?.data || err.message;

console.error('[charge-token] error:', detail);

res.status(500).json({ error: 'Charge failed', detail });

}

});

const port = process.env.PORT || 3000;

app.listen(port, () => console.log(`Backend listening on http://localhost:${port}`));

**3) Frontend (React Vite example)**

If you’re on Next.js, this still applies — just import the button in your page and ensure your Next dev server origin is added to CORS\_ALLOWED\_ORIGINS.

**file: client/package.json**

{

"name": "converge-react-client",

"private": true,

"type": "module",

"scripts": {

"dev": "vite"

},

"dependencies": {

"axios": "^1.7.2",

"react": "^18.3.1",

"react-dom": "^18.3.1"

},

"devDependencies": {

"vite": "^5.4.0"

}

}

**file: client/src/ConvergePayButton.jsx**

import { useEffect, useState } from 'react';

import axios from 'axios';

export default function ConvergePayButton({

amount = '49.00',

orderId = 'INV-1001',

customerId = 'CUST-123',

backendBase = 'http://localhost:3000'

}) {

const [ready, setReady] = useState(false);

const [busy, setBusy] = useState(false);

useEffect(() => {

// Load Converge modal script (production)

const s = document.createElement('script');

s.src = 'https://api.convergepay.com/hosted-payments/PayWithConverge.js';

s.async = true;

s.onload = () => setReady(true);

s.onerror = () => console.error('Failed to load Converge JS (check HTTPS and allowed origin)');

document.head.appendChild(s);

return () => document.head.removeChild(s);

}, []);

const handlePay = async () => {

try {

setBusy(true);

// 1) Ask our backend for a session token

const { data } = await axios.post(`${backendBase}/api/converge/session-token`, {

amount, orderId, customerId, addToken: true

});

const sessionToken = data?.ssl\_txn\_auth\_token;

if (!sessionToken) throw new Error('No session token');

// 2) Open Converge modal

const fields = { ssl\_txn\_auth\_token: sessionToken };

window.PayWithConverge.open(fields, async (result) => {

console.log('Converge result:', result);

if (result?.error || result?.errorText) {

alert(result?.errorText || 'Payment failed');

setBusy(false);

return;

}

const vaultToken = result?.ssl\_token || result?.data?.ssl\_token;

const txnId = result?.ssl\_transaction\_id || result?.data?.ssl\_transaction\_id;

const approved = result?.ssl\_result\_message || result?.data?.ssl\_result\_message;

// 3) Store the vault token in our DB (placeholder endpoint)

if (vaultToken) {

await axios.post(`${backendBase}/api/store-vault-token`, {

customerId, vaultToken, transactionId: txnId, amount

});

}

alert(`Approved: ${approved || 'OK'}\nTxn: ${txnId}\nToken saved: ${!!vaultToken}`);

setBusy(false);

});

} catch (e) {

console.error(e);

alert('Payment init failed. See console.');

setBusy(false);

}

};

return (

<button onClick={handlePay} disabled={!ready || busy}>

{busy ? 'Processing…' : `Pay $${amount}`}

</button>

);

}

**file: client/src/main.jsx**

import React from 'react';

import { createRoot } from 'react-dom/client';

import ConvergePayButton from './ConvergePayButton.jsx';

const backendBase = 'http://localhost:3000'; // replace with your ngrok URL in dev if you want

createRoot(document.getElementById('root')).render(

<React.StrictMode>

<h1>Live Credit Card + Vault Token (Converge)</h1>

<ConvergePayButton

amount="1.23"

orderId={"INV-" + Math.floor(Math.random() \* 1e6)}

customerId="CUST-TEST"

backendBase={backendBase}

/>

</React.StrictMode>

);

**file: client/index.html**

<!doctype html>

<html>

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width,initial-scale=1.0" />

<title>Converge Live Payment</title>

</head>

<body>

<div id="root"></div>

<script type="module" src="/src/main.jsx"></script>

</body>

</html>

**4) Run it locally (HTTP) just to sanity check**

# in project root

npm install

npm run dev

# in client/

cd client

npm install

npm run dev

* backend: http://localhost:3000
* frontend (Vite): likely http://localhost:5173

Try the button; you’ll see the Converge script **fail to load** over HTTP — that’s expected. Next step: **ngrok**.

**5) Use ngrok to get HTTPS (required for Hosted Payments)**

1. install & login:

# one-time install: https://ngrok.com/download

ngrok config add-authtoken <YOUR\_NGROK\_AUTHTOKEN>

1. start your servers:

# in project root

npm run dev

# in client/

npm run dev

1. tunnel your **frontend** (the page that loads the Converge script must be HTTPS):

ngrok http 5173

1. copy the https://YOUR-SUBDOMAIN.ngrok-free.app URL and:
   * add it to **Converge Allowed Origins/HTTP Referrers** (Hosted Payments settings)
   * add it to .env → CORS\_ALLOWED\_ORIGINS=...,https://YOUR-SUBDOMAIN.ngrok-free.app
   * restart backend: npm run dev
2. open your **ngrok HTTPS** URL in a browser, click **Pay**, complete a real card.  
   On approval you’ll get a **vault token** (ssl\_token), which we send to /api/store-vault-token — you’ll see it logged in your backend until you wire DB storage.

**6) Production check-list**

* You must serve your real site over **HTTPS** and whitelist that domain in Converge.
* Keep **merchant\_id / user\_id / pin** **server-side only**.
* The **modal script URL** should be the **production** script:
* https://api.convergepay.com/hosted-payments/PayWithConverge.js
* When you request the session token, include:
  + ssl\_transaction\_type=ccsale
  + ssl\_amount
  + ssl\_get\_token=Y
  + ssl\_add\_token=Y
* Save **ssl\_token** returned after approval; your **Delphi** service will charge it later via XML API.

**7) Quick troubleshooting**

* **Modal won’t open / script won’t load** → your page isn’t **HTTPS** or the **origin is not whitelisted** in Converge.
* **401/403 from token endpoint** → the **Converge API user** lacks permission to request transaction\_token.
* **No ssl\_token returned** → ensure you sent ssl\_get\_token=Y and ssl\_add\_token=Y when creating the session token.
* **CORS error from backend** → add your frontend (local + ngrok) to CORS\_ALLOWED\_ORIGINS in .env.

Want me to adapt this to your **Next.js** repo structure (pages/app router), or wire /api/store-vault-token to your actual DB schema?

Perfect — here’s a clean, step-by-step build that (1) spins up a brand-new React app + Express backend, (2) runs both behind **Cloudflare Tunnel** so your page is HTTPS, and (3) processes a live card via Converge Hosted Payments and returns a **vault token** for you to store.

Below, every code sample is labeled with the **file name** to paste it into. I’ve kept only what you need (no rebilling code).

**0) One-time prerequisites**

* A Converge production terminal with **Hosted Payments / Checkout.js** enabled and your API user (user id + PIN) allowed to request **session tokens**. You’ll whitelist your public HTTPS origin in Converge later. ([Elavon Developer](https://developer.elavon.com/products/converge/v1/checkoutjs?utm_source=chatgpt.com" \o "Checkout.js - Converge), [Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407257235351-Overview-of-Converge-API-Integration-Methods?utm_source=chatgpt.com))
* **cloudflared** installed (Cloudflare Tunnel). ([Cloudflare Docs](https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/do-more-with-tunnels/trycloudflare/?utm_source=chatgpt.com" \o "Quick Tunnels - Cloudflare Zero Trust))

## Node 18+ and npm. **Quick Tunnel Steps Recap (Best for Dev Integration)**

1. Ensure your local server (frontend/backend) is running, e.g., on port 3000.
2. Open terminal, install cloudflared, and run:
3. cloudflared tunnel --url http://localhost:3000
4. Copy the public HTTPS URL it provides.
5. Use that in your .env (backend) or frontend code as the HTTPS origin / backendBase.
6. Whitelist that URL in Converge settings as your allowed origin for Hosted Payments.

## Option 1: Download the .msi installer (easiest)

1. Go to Cloudflare’s official downloads page:  
   👉 https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/downloads/
2. Under **Windows**, download the **cloudflared.msi** installer.
3. Run the installer (double-click it), accept defaults.  
   This will place cloudflared.exe in your PATH.
4. Verify install: open **PowerShell** and run:
5. cloudflared --version

## Option 2: Manual .exe (portable)

1. Download the latest **cloudflared-windows-amd64.exe** from GitHub releases:  
   👉 [https://github.com/cloudflare/cloudflared/releases/latest](https://github.com/cloudflare/cloudflared/releases/latest?utm_source=chatgpt.com)
2. Rename it to cloudflared.exe.
3. Move it to a folder already in your PATH, e.g.:
4. C:\Windows\System32

(or alternatively, create C:\cloudflared\ and add that folder to your PATH in Windows environment variables).

1. Verify:
2. cloudflared --version

## Next Step After Install

Once cloudflared is available, you can run a tunnel:

cloudflared tunnel --url http://localhost:5173 # frontend

cloudflared tunnel --url http://localhost:3173 # backend

It will give you the **https://…trycloudflare.com** URLs to use in your React/Express setup.

**1) Start a new React project + Express backend**

**A) Project layout**

converge-hpp/

.env

package.json

app.js

client/

package.json

index.html

src/

main.jsx

ConvergePayButton.jsx

**B) Backend (Express)**

From the root folder:

npm init -y

npm i express axios cors dotenv qs xml2js

**file: .env**

Put your **live** Converge creds here for real processing.

CONVERGE\_MERCHANT\_ID=YOUR\_MERCHANT\_ID

CONVERGE\_USER\_ID=YOUR\_API\_USER

CONVERGE\_PIN=YOUR\_API\_PIN

# Use production host for live cards

CONVERGE\_BASE=https://api.convergepay.com

# Fill this in later with your Cloudflare FRONTEND URL

CORS\_ALLOWED\_ORIGINS=https://YOUR-frontend-subdomain.trycloudflare.com

PORT=3000

**file: package.json**

{

"name": "converge-express-backend",

"private": true,

"type": "module",

"scripts": {

"dev": "node app.js"

},

"dependencies": {

"axios": "^1.7.2",

"cors": "^2.8.5",

"dotenv": "^16.4.5",

"express": "^4.19.2",

"qs": "^6.12.0",

"xml2js": "^0.6.2"

}

}

**file: app.js**

import 'dotenv/config';

import express from 'express';

import cors from 'cors';

import axios from 'axios';

import qs from 'qs';

const app = express();

app.use(express.json());

// Allow only your HTTPS frontend origin(s)

const allowed = (process.env.CORS\_ALLOWED\_ORIGINS || '')

.split(',')

.map(s => s.trim())

.filter(Boolean);

app.use(cors({

origin: (origin, cb) => {

if (!origin) return cb(null, true); // curl/postman

if (allowed.includes(origin)) return cb(null, true);

return cb(new Error(`Origin ${origin} not allowed`));

}

}));

const CONVERGE\_BASE = process.env.CONVERGE\_BASE || 'https://api.convergepay.com';

const TIMEOUT = 15000;

// 1) Ask Converge for a session token (for the modal)

app.post('/api/converge/session-token', async (req, res) => {

try {

const { amount, orderId, customerId, addToken = true } = req.body;

if (!amount || !orderId) return res.status(400).json({ error: 'amount and orderId are required' });

const form = {

ssl\_merchant\_id: process.env.CONVERGE\_MERCHANT\_ID,

ssl\_user\_id: process.env.CONVERGE\_USER\_ID,

ssl\_pin: process.env.CONVERGE\_PIN,

ssl\_transaction\_type: 'ccsale',

ssl\_amount: amount,

ssl\_currency\_code: 'USD',

ssl\_invoice\_number: orderId,

ssl\_get\_token: addToken ? 'Y' : 'N',

ssl\_add\_token: addToken ? 'Y' : 'N',

ssl\_customer\_id: customerId || undefined

};

const url = `${CONVERGE\_BASE}/hosted-payments/transaction\_token`;

const { data } = await axios.post(url, qs.stringify(form), {

headers: { 'Content-Type': 'application/x-www-form-urlencoded' },

timeout: TIMEOUT

});

if (!data?.ssl\_txn\_auth\_token) {

console.error('[Converge token resp]', data);

return res.status(502).json({ error: 'No ssl\_txn\_auth\_token in response', raw: data });

}

res.json({ ssl\_txn\_auth\_token: data.ssl\_txn\_auth\_token });

} catch (err) {

const detail = err?.response?.data || err.message;

console.error('[session-token] error:', detail);

res.status(500).json({ error: 'Failed to get session token', detail });

}

});

// 2) Placeholder to store the vault token you receive (wire to your DB)

app.post('/api/store-vault-token', async (req, res) => {

const { customerId, vaultToken, transactionId, amount } = req.body || {};

if (!vaultToken) return res.status(400).json({ error: 'vaultToken is required' });

// TODO: replace with your DB insert/update

console.log('[STORE TOKEN]', { customerId, vaultToken, transactionId, amount, ts: new Date().toISOString() });

res.json({ ok: true });

});

const port = process.env.PORT || 3000;

app.listen(port, () => console.log(`Backend listening on http://localhost:${port}`));

The session-token endpoint uses Converge’s **transaction\_token** API to obtain ssl\_txn\_auth\_token for Checkout.js/PayWithConverge, which you’ll open in the browser. ([Elavon Developer](https://developer.elavon.com/products/converge/v1/checkoutjs?utm_source=chatgpt.com" \o "Checkout.js - Converge), [Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407206952087-Developer-Portal-and-Useful-Links?utm_source=chatgpt.com))

**C) Frontend (React with Vite)**

From the **converge-hpp/** root:

mkdir client && cd client

npm create vite@latest . -- --template react

npm i

npm i axios

**file: client/index.html**

<!doctype html>

<html>

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width,initial-scale=1.0" />

<title>Converge Live Payment</title>

</head>

<body>

<div id="root"></div>

<script type="module" src="/src/main.jsx"></script>

</body>

</html>

**file: client/src/ConvergePayButton.jsx**

import { useEffect, useState } from 'react';

import axios from 'axios';

export default function ConvergePayButton({

amount = '1.23',

orderId = 'INV-' + Math.floor(Math.random() \* 1e6),

customerId = 'CUST-TEST',

backendBase // set this to your backend HTTPS tunnel URL

}) {

const [ready, setReady] = useState(false);

const [busy, setBusy] = useState(false);

useEffect(() => {

// Load Converge modal script (production)

const s = document.createElement('script');

s.src = 'https://api.convergepay.com/hosted-payments/PayWithConverge.js';

s.async = true;

s.onload = () => setReady(true);

s.onerror = () => console.error('Failed to load Converge JS (check HTTPS + allowed origin in Converge)');

document.head.appendChild(s);

return () => document.head.removeChild(s);

}, []);

const handlePay = async () => {

try {

setBusy(true);

const { data } = await axios.post(`${backendBase}/api/converge/session-token`, {

amount, orderId, customerId, addToken: true

});

const sessionToken = data?.ssl\_txn\_auth\_token;

if (!sessionToken) throw new Error('No session token');

const fields = { ssl\_txn\_auth\_token: sessionToken };

window.PayWithConverge.open(fields, async (result) => {

console.log('Converge result:', result);

if (result?.error || result?.errorText) {

alert(result?.errorText || 'Payment failed');

setBusy(false);

return;

}

const vaultToken = result?.ssl\_token || result?.data?.ssl\_token;

const txnId = result?.ssl\_transaction\_id || result?.data?.ssl\_transaction\_id;

const approvedMsg = result?.ssl\_result\_message || result?.data?.ssl\_result\_message;

if (vaultToken) {

await axios.post(`${backendBase}/api/store-vault-token`, {

customerId, vaultToken, transactionId: txnId, amount

});

}

alert(`Approved: ${approvedMsg || 'OK'}\nTxn: ${txnId}\nToken saved: ${!!vaultToken}`);

setBusy(false);

});

} catch (e) {

console.error(e);

alert('Payment init failed. See console.');

setBusy(false);

}

};

return (

<button onClick={handlePay} disabled={!ready || busy}>

{busy ? 'Processing…' : `Pay $${amount}`}

</button>

);

}

**file: client/src/main.jsx**

import React from 'react';

import { createRoot } from 'react-dom/client';

import ConvergePayButton from './ConvergePayButton.jsx';

const BACKEND\_HTTPS = 'https://YOUR-backend-subdomain.trycloudflare.com'; // fill in after tunneling

createRoot(document.getElementById('root')).render(

<React.StrictMode>

<h1>Live Payment + Vault Token</h1>

<ConvergePayButton

amount="1.23"

customerId="CUST-TEST"

backendBase={BACKEND\_HTTPS}

/>

</React.StrictMode>

);

The page loads Converge’s **PayWithConverge.js** and opens the modal with the ssl\_txn\_auth\_token you get from your backend. Converge returns ssl\_token (vault token) on success, which you then POST to your backend to store. ([CodePen](https://codepen.io/ElavonNiko/pen/XydpEd?utm_source=chatgpt.com" \o "Lightbox Initiate Transaction Demo))

**2) Run the servers locally**

**Terminal 1 (backend):**

# in converge-hpp/

npm run dev

# -> http://localhost:3000

**Terminal 2 (frontend):**

# in converge-hpp/client

npm run dev

# -> likely http://localhost:5173

**3) Put both services behind Cloudflare Tunnel (HTTPS)**

You’ll create two tunnels (one for the frontend, one for the backend) to avoid mixed-content blocks in the browser. Quick Tunnel is fine for dev:  
cloudflared tunnel --url http://localhost:PORT ([Cloudflare Docs](https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/do-more-with-tunnels/trycloudflare/?utm_source=chatgpt.com" \o "Quick Tunnels - Cloudflare Zero Trust))

**Terminal 3 (frontend tunnel):**

cloudflared tunnel --url http://localhost:5173

# copy the https URL it prints, e.g. https://abc123-5173.trycloudflare.com

**Terminal 4 (backend tunnel):**

cloudflared tunnel --url http://localhost:3000

# copy the https URL, e.g. https://xyz456-3000.trycloudflare.com

Update:

* **client/src/main.jsx** → set BACKEND\_HTTPS to your **backend** tunnel URL.
* **.env** → set CORS\_ALLOWED\_ORIGINS to your **frontend** tunnel URL.
* Restart the backend: npm run dev.

(If you prefer a **persistent named** tunnel with a stable hostname, use the Cloudflare “locally-managed tunnel” workflow; it’s a few extra steps to authenticate and create a config. ([Cloudflare Docs](https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/do-more-with-tunnels/local-management/create-local-tunnel/?utm_source=chatgpt.com" \o "Create a locally-managed tunnel · Cloudflare Zero Trust docs)))

**4) Converge portal settings (one time)**

1. Add your **frontend** Cloudflare HTTPS URL to **Allowed Origins / HTTP Referrers** (Hosted Payments / Checkout.js settings).
2. Make sure Hosted Payments / Checkout.js is **enabled**, and your API user has permission to request **transaction tokens**.
3. Use the **live** script in your page:  
   https://api.convergepay.com/hosted-payments/PayWithConverge.js
4. Your backend requests the session token at:  
   POST https://api.convergepay.com/hosted-payments/transaction\_token (server-to-server). ([Elavon Developer](https://developer.elavon.com/products/converge/v1/checkoutjs?utm_source=chatgpt.com" \o "Checkout.js - Converge), [Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407257235351-Overview-of-Converge-API-Integration-Methods?utm_source=chatgpt.com))

**5) Do a live card + capture the vault token**

1. Open the **frontend Cloudflare URL** in your browser.
2. Click **Pay** → Converge modal opens.
3. Enter a real card. On approval you’ll see an alert and your server logs will show [STORE TOKEN] ... with the **ssl\_token** (vault token). Persist it in your DB in /api/store-vault-token.

**6) Wire /api/store-vault-token to your DB (placeholder)**

Replace the console.log with your insert/update (e.g., Informix/MSSQL). Keep the **token** and your own **customer/member** keys.

**7) Quick troubleshooting**

* **Modal won’t open / script fails to load** → Your page is not HTTPS or your **frontend** origin isn’t whitelisted in Converge. ([Cartis Payments](https://knowledge.cartispayments.com/hc/en-us/articles/4407257235351-Overview-of-Converge-API-Integration-Methods?utm_source=chatgpt.com" \o "Overview of Converge API Integration Methods))
* **CORS errors** → .env CORS\_ALLOWED\_ORIGINS must include your **frontend** Cloudflare URL (exact origin).
* **No ssl\_token in result** → ensure backend sent ssl\_get\_token=Y and ssl\_add\_token=Y when requesting the session token. ([Elavon Developer](https://developer.elavon.com/products/converge/v1/checkoutjs?utm_source=chatgpt.com" \o "Checkout.js - Converge))
* **403 when opening modal** → typically an origin/permission issue on the Converge side. Verify the allowed origin and API user privileges. ([Stack Overflow](https://stackoverflow.com/questions/57693021/converge-lightbox-integration?utm_source=chatgpt.com))

**That’s it**

You now have a minimal React + Express app, fully HTTPS via **Cloudflare Tunnel**, that runs a **live** Converge card payment and returns a **vault token** you can save for your Delphi service. If you want this converted to **Next.js** (App Router) instead of Vite React, say the word and I’ll drop in those file layouts next.