Converge Breakdown

Got you. Here’s a clean, end-to-end breakdown—no app code—covering Converge setup (whitelisting + API user), Cloudflare Tunnel steps, and what each part of your React/Express app does.

**Architecture at a glance**

* **Frontend (React/Vite)** runs on your PC (localhost:5173) and is exposed to the internet by **Cloudflare Tunnel** (a public https://\*.trycloudflare.com URL).
* **Backend (Express)** runs on your PC (you chose port 3173) and is exposed by a second Cloudflare Tunnel URL.
* **Flow**: Frontend → Backend /api/converge/session-token → Converge returns a *session token* → Frontend opens the Hosted Payments modal → customer pays → Converge returns a result (contains the **vault token ssl\_token**) → Frontend calls Backend /api/store-vault-token so you can save it.

**1) Cloudflare Tunnel (quick tunnels) — front & back**

You’ll run **two** tunnels (one per local service). Open two PowerShell windows:

**A. Frontend tunnel**

1. Start your React dev server locally (Vite) on port 5173.
2. In PowerShell window #1, start a quick tunnel pointing to http://localhost:5173.
3. Copy the printed public HTTPS URL (call this **FRONTEND\_URL**). You will whitelist this in Converge and allow it in Vite.

**B. Backend tunnel**

1. Start your backend locally on port **3173**.
2. In PowerShell window #2, start a quick tunnel pointing to http://localhost:3173.
3. Copy the printed public HTTPS URL (call this **BACKEND\_URL**). You will point the frontend to this.

**Notes**

* Keep both PowerShell windows open; closing them kills the tunnels.
* Quick tunnel URLs change every time. If you want a stable hostname, set up a **Named Tunnel** later and map it to a subdomain you control (one-time whitelisting in Converge).

**2) Converge — whitelist your Cloudflare frontend URL**

Converge only allows the payment modal to load from **known origins**.

1. Sign in to Converge.
2. Go to **Settings → Hosted Payments** (sometimes under “Advanced API Settings”).
3. Find **Allowed Referrers / Allowed Origins** (naming varies).
4. Add your **FRONTEND\_URL** exactly as printed by cloudflared (e.g., https://tribune-…trycloudflare.com).
5. Save.

If your account uses IP/domain controls for server-to-server APIs, also open **Settings → Online API Security Allowed List Manager** and add your **public egress IP** (this is for the backend → Converge call; the Cloudflare URL itself isn’t used here).

**3) Converge — enable Hosted Payments, tokenization, and fix modal fields**

For the session token + vault token flow to work:

1. **Hosted Payments enabled** for the terminal.
2. **Tokenization enabled** for the terminal (so ssl\_token is returned).
   * You’ll typically see this under **Settings → Terminal Information** (look for “Tokenization: Enabled”).
3. **Hosted Payments modal fields sane**:
   * Open **Settings → Hosted Payments → Modal Setting**.
   * While testing, keep it minimal: **Show Amount** on; both **Address Verification Fields** off.
   * Language = English. Save.
4. **CVV/CSC field** must be either **properly enabled** (with a label) or **fully disabled**. A half-configured CVV causes the Angular template errors.
   * Go to **Settings → Payment Fields**.
   * In the **Order Section**, either add “Security Code / CVV / CSC” (display ON, label set), or leave it out entirely. Save.
   * If you don’t see a CVV field to add, flip it at **Settings → Terminal Information** (or a Security/Terminal page): set **Card Security Code (CVV/CVC/CSC)** to **Enabled / Prompt** or **Disabled**, then Save.

Hard-refresh your page after saving so the modal template reloads.

**4) Converge — make sure the API user “BOSS” can call Hosted Payments**

You already have the **BOSS** user and verified credentials work from PowerShell. Double-check these so your app keeps working:

1. **User exists and is active**
   * Go to **Employees / Users** (menu label varies).
   * Edit the **BOSS** user (or create if needed) and ensure it’s linked to the correct terminal/merchant.
2. **PIN set**
   * API calls use the **PIN**, not the web password.
   * If you’ve exposed the PIN anywhere public, rotate it immediately and update your app’s environment variables.
3. **Permissions**
   * The BOSS user must have permission to use **Hosted Payments / Checkout.js / Payment Modal** and specifically **request a transaction token**.
   * If your account has separate toggles, also ensure **Transaction API / Online API** is enabled for BOSS.
   * If your org uses **Online API Security Allowed List Manager**, ensure your server’s public IP is allowed.
4. **Tokenization on the terminal**
   * Needed for Converge to return ssl\_token after approval.
5. **Whitelisting done**
   * The **FRONTEND\_URL** must be in Hosted Payments Allowed Referrers/Origins (Section 2 above).

**5) What each part of your web app does (no code)**

**Frontend (React/Vite)**

* **Main page (“Live Payment + Vault Token”)**
  + Renders a **Pay** button.
  + When clicked, it calls the backend to **request a session token** for the specific amount/order/customer.
  + Loads **PayWithConverge.js** and opens the **Hosted Payment modal** using that session token.
  + Handles all outcomes:
    - **Approved**: shows a friendly status and key fields (approval code, result text, transaction id, last4, AVS/CVV responses, amount) and the **vault token** if present.
    - **Declined**: shows the decline message and details.
    - **Canceled/Error**: shows what Converge returned (e.g., errorText) and any hints.
  + Always clears the “Processing…” state even if the modal misbehaves.
  + Sends the **vault token (ssl\_token)** and transaction id to the backend to store.
* **Vite host allow-list**
  + Vite restricts which hostnames can load the dev server. You allowed your **FRONTEND\_URL** (or all \*.trycloudflare.com) so the page loads via the tunnel without “Blocked request” errors.

**Backend (Express)**

* **/api/converge/session-token**
  + Receives amount/order/customer.
  + Uses your **Merchant ID + API User ID + PIN** to call Converge’s **transaction\_token** endpoint (production host).
  + Accepts either text or JSON response from Converge; extracts the **session token** and returns it to the frontend.
  + If Converge rejects the request, the route forwards the **HTTP status** and **upstream message** and adds a short **hint** (e.g., 401/403/origin/IP, missing fields), so the frontend can display a helpful error panel.
* **/api/store-vault-token**
  + Receives the **vault token (ssl\_token)** and **transaction id** from the frontend after approval.
  + Today it just logs them; in your real app you persist to your DB keyed by your customer ID.
  + This route does **not** call Converge; it’s purely for saving.
* **CORS allow-list**
  + The backend only accepts browser requests from your **FRONTEND\_URL**. If it doesn’t match exactly, the backend returns a clear JSON “CORS origin blocked” error (rather than a silent 403).
* **Environment values** (what they mean)
  + Merchant ID, API User ID, PIN: the credentials for the **BOSS** API user on your terminal.
  + Converge Base URL: the **production** Converge host.
  + CORS Allowed Origins: the **FRONTEND\_URL** from Cloudflare Tunnel.
  + Port: your backend’s local port (**3173** in your case).

**6) End-to-end checklist (what to do each time)**

1. Start **backend** locally on the correct port (3173).
2. Start **frontend** locally (Vite on 5173).
3. Start **two Cloudflare tunnels** (one pointing to 5173, one to 3173).
4. In the frontend, set its **backend base URL** to the **BACKEND\_URL**.
5. In the backend, set **CORS Allowed Origins** to the **FRONTEND\_URL**, then restart the backend.
6. In **Converge**, make sure **FRONTEND\_URL** is in **Allowed Referrers/Origins**.
7. Ensure **Hosted Payments** and **Tokenization** are enabled, and **BOSS** has permission to request transaction tokens.
8. Open the **FRONTEND\_URL** in your browser. Click **Pay**.
9. Complete a **real card**. On approval you’ll see:
   * Friendly status (**APPROVED/DECLINED/ERROR/CANCELED**),
   * Key fields (approval code, result message, transaction id, AVS/CVV),
   * The **vault token** (ssl\_token) in the details and stored via the backend route.

**7) Troubleshooting quick map**

* **Modal never opens / “host not allowed”** → Your Vite allowed host is missing or **FRONTEND\_URL** not whitelisted in Converge.
* **401 from /session-token** → wrong Merchant/User/PIN or user lacks Hosted Payments permission.
* **403 from /session-token** → Hosted Payments disabled for the terminal, API user lacks permission, or your Online API Security allow-list is blocking your server IP.
* **Angular errors / {{paymentModalCtrl…}} in a field** → CVV/CSC field mis-configured. Either properly enable it (with a label) in **Payment Fields** or fully disable CVV at the terminal.
* **Stuck on “Processing…”** → modal didn’t return a callback (often due to the CVV issue). After fixing CVV, the UI will recover normally; the frontend also has a safety timeout.
* **You see a charge in Converge, but no token on your page** → Tokenization must be **enabled** on the terminal; some configurations only return a token when requested (ssl\_get\_token/ssl\_add\_token) and permitted.

**8) Security notes (just to be safe)**

* Rotate the **PIN** if it was ever posted publicly.
* Do not log PANs or sensitive values in production; store **tokens** only.
* Treat **FRONTEND\_URL** as a testing origin; move to a **Named Tunnel** or a real HTTPS domain for production and whitelist that once in Converge.

If you want, I can also outline the **Named Tunnel** steps so you get stable URLs and don’t have to update Converge every time.

Sweet—let’s move you from “quick tunnels” to a **Named Tunnel** with **stable subdomains**. We’ll put **both** your frontend and backend behind **one** tunnel and two hostnames.

**Prereqs**

* You already installed cloudflared on Windows.
* Your domain is on Cloudflare (nameservers pointing to Cloudflare).
* You can log in to the Cloudflare dashboard with permission to create DNS records.

Pick your permanent hostnames (examples):

* **Frontend:** pay.yourdomain.com
* **Backend API:** pay-api.yourdomain.com

Replace yourdomain.com with your real domain throughout.

**Part 1 — Authenticate cloudflared**

1. Open **PowerShell**.
2. Run:
3. cloudflared tunnel login

A browser window opens → pick your Cloudflare account & zone (your domain).  
When it finishes, it drops a cert file at:

C:\Users\<YOU>\.cloudflared\cert.pem

**Part 2 — Create one Named Tunnel**

1. Create a tunnel (name it anything, e.g. payments):
2. cloudflared tunnel create payments

Output will include a **Tunnel UUID** and create a credentials file:

C:\Users\<YOU>\.cloudflared\<UUID>.json

(Keep the UUID handy; we’ll reference it in the config.)

**Part 3 — Map DNS (stable hostnames)**

Create DNS routes (Cloudflare will add CNAMEs for you):

# Frontend hostname

cloudflared tunnel route dns payments pay.yourdomain.com

# Backend hostname

cloudflared tunnel route dns payments pay-api.yourdomain.com

These commands create DNS records pointing to the tunnel. No need to add A/AAAA records yourself.

**Part 4 — Create the tunnel config (single file, 2 hostnames)**

Create (or edit) the config file:

C:\Users\<YOU>\.cloudflared\config.yml

Put this in it (adjust paths/hostnames/ports):

tunnel: <YOUR-TUNNEL-UUID>

credentials-file: C:\Users\<YOU>\.cloudflared\<YOUR-TUNNEL-UUID>.json

ingress:

# Frontend: Vite dev server on localhost:5173

- hostname: pay.yourdomain.com

service: http://localhost:5173

# Backend: Express on localhost:3173

- hostname: pay-api.yourdomain.com

service: http://localhost:3173

# Default rule (catch-all)

- service: http\_status:404

Later, when you build for production, you can point service to your production web server instead of Vite.

**Part 5 — Test the tunnel**

Run it in the foreground first:

cloudflared tunnel run payments

* Visit https://pay.yourdomain.com → you should see your frontend.
* Visit https://pay-api.yourdomain.com/api/health (or any route) → your backend should respond.

If both work, you’re ready to run it as a Windows service.

**Part 6 — Install as a Windows Service (auto-start)**

1. Stop the foreground run (Ctrl+C).
2. Install the service:
3. cloudflared service install

(It reads config.yml from C:\Users\<YOU>\.cloudflared\.)

1. Start/stop/check the service:
2. net start cloudflared
3. sc query cloudflared
4. net stop cloudflared

After editing config.yml, just restart the service.

If service install complains, ensure config.yml exists and the credentials-file path is correct.

**Part 7 — Update your app & Converge to use the stable domains**

**Frontend (React/Vite)**

* In your client code where you set the backend base URL, change to:
* https://pay-api.yourdomain.com
* Vite dev server host checks: allow your new frontend host. In your Vite config, set server.allowedHosts to include pay.yourdomain.com (or a wildcard like .yourdomain.com). Restart Vite.

**Backend (Express)**

* CORS allow-list should now be:
* CORS\_ALLOWED\_ORIGINS=https://pay.yourdomain.com

Restart the backend.

**Converge (Hosted Payments)**

* **Allowed Referrers / Allowed Origins**: add
* https://pay.yourdomain.com

(You no longer need to re-whitelist every time—this hostname is stable.)

* Ensure **Hosted Payments** and **Tokenization** are enabled on your terminal.
* Confirm your API user (e.g., **BOSS**) has permission to request **transaction\_token**.
* If your org uses **Online API Security Allowed List Manager** (IP allowlisting), make sure your **outbound public IP** is allowed (requests go from your PC/server to Converge, not from Cloudflare).

**Part 8 — One-tunnel vs two-tunnel notes**

* **Recommended**: one named tunnel with multiple hostnames (as we configured). Simple and resilient.
* If you prefer, you can make **two** named tunnels (one per service). Not needed unless you want to separate blast radius or policies.

**Part 9 — Daily operations & tips**

* **Editing config**: update config.yml → net stop cloudflared → net start cloudflared.
* **Logs**:
* Get-EventLog -LogName Application -Newest 50 | where {$\_.Source -like "\*cloudflared\*"}

Or run foreground temporarily: cloudflared tunnel run payments.

* **Origin HTTPS**: If your local origin is HTTPS with a self-signed cert, set:
* originRequest:
* noTLSVerify: true

(Put it at the top level of config.yml.)

* **Zero Trust Access** (optional): You can protect pay-api.yourdomain.com behind Cloudflare Access policies later; not required for Converge.

**Part 10 — Final checklist (stable)**

1. cloudflared tunnel login (one time)
2. cloudflared tunnel create payments (one time)
3. cloudflared tunnel route dns payments pay.yourdomain.com  
   cloudflared tunnel route dns payments pay-api.yourdomain.com
4. config.yml with two ingress hostnames → local ports
5. cloudflared tunnel run payments (test) → then cloudflared service install (persist)
6. Frontend points to https://pay-api.yourdomain.com
7. Backend CORS allows https://pay.yourdomain.com
8. Converge **Allowed Referrers** = https://pay.yourdomain.com
9. Hosted Payments + Tokenization enabled; API user **BOSS** permitted
10. Run a live card → UI shows approved/declined + **ssl\_token**; backend stores token

That’s it—you’ll have stable HTTPS URLs that don’t change, no more re-whitelisting, and your dev/prod can share the same pattern.