

Overview

This document captures the key steps, payload iterations, and diagnostics exchanged while troubleshooting and successfully authorising Worldpay payments within the Nomad Stop Next.js project. Use it as a reference or to brief another developer on the journey from the initial errors to the final working payload.

1. Context & Starting Point

- Project: `Nomad-Stop-NextJS` (Next.js app with Prisma, Supabase, Worldpay).
 - Initial blockers:
 - Checkout returning "Requested endpoint was not found".
 - Menu content missing when database disabled.
 - Need for sandbox payments to succeed locally while Supabase connection was unreliable.
-

2. Menu Mock Data (DB Disabled Mode)

What was done:

- Added `DISABLE_DB="true"` in `.env` to allow frontend testing without Supabase.
- Created reusable mock product catalogue (`src/lib/mockMenu.ts`).
- Updated `PopularProducts` and `/menu` page to load mock data when DB disabled.

Result:

- Complete menu (Afghan Specials, Karahi, Doner & Grill, Pizza, Sides, Desserts, Drinks, Deals) is available while offline or during DB outages.
-

3. Worldpay API Troubleshooting Timeline

3.1 Initial Attempt – Legacy Orders API

- Endpoint: `/orders` – returned **404 Not Found**.
- Diagnosis: Sandbox credentials are for **Access Worldpay** (Payments API), not the legacy Orders product.

3.2 Switch to `/payments/requests`

- Endpoint: `/payments/requests` with `application/vnd.worldpay.payments-v6+json` .
- Error: **404 Not Found** – endpoint not provisioned for our tenant.
- Insight: Resource discovery file shows enabled endpoints under `/payments/authorizations` .

3.3 Move to `/payments/authorizations`

- Initial payload mirrored card fields but caused **405 Method Not Allowed** and **bodyDoesNotMatchSchema** responses.
- Actions:
 1. Adjusted media types and request path.
 2. Reworked payload to match Access schema.
 3. Iteratively addressed validation errors reported in `validationErrors` .

3.4 Payload Evolution

Attempt	Error	Fix
404 – endpoint not found	Using /orders	Switch to Access API endpoints
405 – method not allowed	POST to /payments	Use /payments/authorizations
bodyDoesNotMatchSchema (missing instruction.value)	Added instruction.value as integer	Still wrong structure
fieldMustBeNumber, fieldIsMissing	Expiry year, narrative, value structure incorrect	Converted to numbers, added instruction.value.amount, narrative objects
fieldHasInvalidValue (narrative length)	Narrative > 24 chars	Truncated to short reference (orderId.slice(0, 24))
✅ Final	No validation errors	Payment authorised

3.5 Final Authorisation Payload

```
{
  "transactionReference": "mock-order-1762881663310",
  "merchant": {
    "entity": "P04085650052"
  },
  "instruction": {
    "value": {
      "currency": "GBP",
      "amount": 2499
    },
    "narrative": {
      "line1": "Order 1762881663310"
    },
    "amount": {
      "currency": "GBP",
      "value": 2499
    },
    "paymentInstrument": {
      "type": "card/plain",
      "cardHolderName": "Test User",
      "cardNumber": "444433332221111",
      "cardExpiryDate": {
        "month": 11,
        "year": 2028
      },
      "cardSecurityCode": "123"
    }
  }
}
```

Headers (unchanged throughout debugging):

```
Content-Type: application/vnd.worldpay.payments-v6+json
Accept:      application/vnd.worldpay.payments-v6+json
Authorization: Basic base64(username:password)
```

4. Logging Enhancements

- Added verbose logging in `src/lib/worldpay.ts` :
 - Payload preview (with masked card).
 - Raw Worldpay responses (successful or error).
 - Captured validation error arrays (`validationErrors`) to guide schema fixes.
-

5. Successful Authorisations (Final Logs)

```
Worldpay env check { ... apiUrl:
'https://try.access.worldpay.com/payments/authorizations' }
Worldpay authorization request { orderId: 'mock-order-1762881663310', payload: { ... }
}
POST /api/payments/worldpay/create-intent 200 in 1131ms
GET /order/pending?oid=mock-order-1762881663310 200 in 63ms
```

- The checkout UI confirms status “Payment Authorized”.
 - Order pending page displays order number, status, and total.
-

6. Key Files Touched

- `src/lib/worldpay.ts` – request payload schema, logging, endpoint selection.
 - `src/app/api/payments/worldpay/create-intent/route.ts` – error handling and Prisma bypass when DB disabled.
 - `src/lib/mockMenu.ts` , `src/app/menu/page.tsx` , `src/components/PopularProducts.tsx` – menu mock data.
 - `.env` – `DISABLE_DB=true` , Worldpay credentials, Supabase DSN placeholder during local testing.
-

7. Remaining Recommendations

- Remove `DISABLE_DB` once Supabase credentials are solid; re-enable Prisma queries.
 - Mask or rotate Worldpay credentials if shared publicly.
 - Consider storing Worldpay response metadata (e.g., `payment.id`) in the payment record when DB is enabled.
 - Implement capture/void endpoints as needed (stubs already in `worldpay.ts`).
-

8. Quick Reference Commands

```
# Start dev server
npm run dev
```

```
# Seed database (when Supabase connection ready)
npx prisma migrate deploy
npx prisma db seed

# Environment variable template
DATABASE_URL="postgresql://..."
WORLDPAY_USERNAME="..."
WORLDPAY_PASSWORD="..."
WORLDPAY_CHECKOUT_ID="..."
WORLDPAY_ENTITY_ID="P04085650052"
WORLDPAY_ENVIRONMENT="sandbox"
WORLDPAY_WEBHOOK_SECRET="..."
DISABLE_DB="true"
```

9. Useful Links

- Worldpay Access Payments API: <https://developer.worldpay.com/products/payments>
- Sandbox API reference (/payments/authorizations):
<https://developer.worldpay.com/products/payments/openapi>

10. Contact

Prepared by Cursor (AI assistant) for handoff to collaborating developers. Feel free to share this document as-is or export to PDF for distribution.