PCB Auto × Dough Gateway — Sandbox Integration

🚀 What This Does

Complete payment processing integration for **PCB Auto** (auto repair shop module within PCBISV.com) connected to the **Dough Gateway** sandbox environment.

Includes:

- Full TypeScript service for all Dough Gateway API endpoints
- PCB Auto business-logic layer (repair orders, deposits, vault, fees)
- React `<PCBAutoPaymentForm>` with secure Tokenizer iframe (PCI-compliant)
- Webhook handler for real-time transaction notifications
- Sandbox test suite with all test card numbers
- Complete type definitions for the entire API

⚠️ WHAT YOU NEED (Before This Will Work)

You're missing 3 things that only come from your Dough Gateway sandbox account:

Item	Format	Where to Get It
1. Sandbox API Key	`api_***`	Sandbox portal → Settings → API Keys
2. Sandbox Public Key	`pub_***`	Same page → Create Public Key
3. Processor ID	alphanumeric string	Settings → Processors

How to get them:

1. **Log into the sandbox portal:**
   `https://sandbox.doughgateway.com`
   (If you don't have login credentials yet, contact your Dough/PC Bancard rep to provision a sandbox merchant account)
2. **Create API Keys:**
   - Navigate to **Settings → API Keys**
   - Create a new **API Key** (starts with `api_`) — this is your server-side key

- Create a new **Public Key** (starts with `pub_`) — this is for the client-side Tokenizer

3. **Get your Processor ID:**
   - Navigate to **Settings → Processors**
   - Copy the ID of your active sandbox processor
   - If a default processor is configured, this step is optional
4. **Set environment variables:**

```bash
export DOUGH_SANDBOX_API_KEY="api_your_actual_key"
export DOUGH_SANDBOX_PUBLIC_KEY="pub_your_actual_key"
export DOUGH_SANDBOX_PROCESSOR_ID="your_processor_id"   # if needed
```

📁 Project Structure

```
pcb-auto-dough-gateway/
├── src/
│   ├── index.ts                    # Barrel exports
│   ├── types/
│   │   └── dough-gateway.types.ts      # All TypeScript types
│   ├── services/
│   │   ├── dough-gateway.service.ts    # Core gateway API service
│   │   ├── pcb-auto-payment.service.ts  # Auto repair business logic
│   │   └── webhook-handler.ts          # Express webhook middleware
│   ├── components/
│   │   └── PCBAutoPaymentForm.tsx      # React payment form w/ Tokenizer
│   ├── utils/
│   │   └── config.ts               # Config + test data constants
│   └── tests/
│       └── sandbox-test.ts           # Sandbox connectivity test suite
├── .env.example
└── README.md
```

🧪 Running the Sandbox Tests

```bash
```

```
# 1. Set your credentials
export DOUGH_SANDBOX_API_KEY="api_your_key"

# 2. Run tests
npx ts-node src/tests/sandbox-test.ts
```

The test suite runs 10 tests:

1. Basic card sale ($25.99)
2. Expected decline
3. Full repair order payment
4. Transaction retrieval
5. Transaction void
6. Customer vault creation (card on file)
7. Charge stored customer card
8. Transaction search
9. ACH payment
10. Fee/amount calculation

💳 Test Card Numbers

Approval Cards

Card	Brand	Type	Surchargeable
4111111111111111	Visa	Debit	No
4005519200000004	Visa	Credit	Yes
5555555555554444	Mastercard	Debit	No
2223000048400011	Mastercard	Credit	Yes
378282246310005	Amex	Credit	Yes
6011111111111117	Discover	Credit	Yes

Trigger Cards (Test Specific Responses)

Card	Response
4000000000000002	**Decline**
4000000000009995	Insufficient Funds
4000000000009987	Lost Card
4000000000009979	Stolen Card
4000000000000069	Expired Card
4000000000000051	Partial Approval (50%)

CVV Triggers

CVV	Response
Any value	M (Match)
200	N (No Match)
201	U (Not Verified)
301	S (Not Participating)

AVS Triggers

Postal Code	Response
Any value	M (Match)
20000	N (No Match)
20001	U (Not Verified)

🔧 Integration into PCBISV.com

Server-Side (API Route)

typescript

```typescript
import { PCBAutoPaymentService, SANDBOX_CONFIG } from './pcb-auto-dough-gateway';

const payments = new PCBAutoPaymentService(SANDBOX_CONFIG);

// In your API route handler:
app.post('/api/pcb-auto/pay', async (req, res) => {
  const { repairOrder, token } = req.body;

  const result = await payments.processTokenPayment(repairOrder, token);

  if (result.success) {
    // Update repair order status in your DB
    // Store transactionId for refunds/voids
    res.json({ success: true, receipt: result.receiptData });
  } else {
    res.status(400).json({ success: false, error: result.errorMessage });
  }
});
```

Client-Side (React)

tsx

```tsx
import { PCBAutoPaymentForm } from './pcb-auto-dough-gateway';

function CheckoutPage({ repairOrder }) {
  const handleSuccess = async (token: string) => {
    const res = await fetch('/api/pcb-auto/pay', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ repairOrder, token }),
    });
    const data = await res.json();
    if (data.success) {
      // Show receipt, redirect, etc.
    }
  };

  return (
    <PCBAutoPaymentForm
      publicKey="pub_YOUR_KEY"
      gatewayUrl="https://sandbox.doughgateway.com"
      repairOrder={{
        repairOrderId: repairOrder.id,
        customerName: repairOrder.customerName,
        subtotal: repairOrder.subtotal,
        tax: repairOrder.tax,
        total: repairOrder.total,
      }}
      onPaymentSuccess={handleSuccess}
      onPaymentError={(err) => console.error(err)}
    />
  );
}
```

Webhooks

typescript

```
import { createWebhookHandler } from './pcb-auto-dough-gateway';

app.post('/webhooks/dough',
  express.raw({ type: 'application/json' }),
  createWebhookHandler({
    clientSecret: process.env.DOUGH_WEBHOOK_SECRET!,
    onTransactionSettled: async (data) => {
      // Update repair order as settled
      console.log(`Transaction ${data.transaction_id} settled`);
    },
    onSettlementBatch: async (data) => {
      // Record batch settlement for reconciliation
    },
  })
);
```

🔑 API Endpoint Reference

Method	Endpoint	Description
POST	/api/transaction	Process sale or auth
GET	/api/transaction/{id}	Get transaction
POST	/api/transaction/{id}/capture	Capture auth
POST	/api/transaction/{id}/void	Void pending
POST	/api/transaction/{id}/refund	Refund settled
POST	/api/transaction/search	Search transactions
POST	/api/calculate/amounts	Calculate fees/surcharge
POST	/api/lookup/fees	Look up fees
POST	/api/customer	Create vault customer
GET	/api/customer/{id}	Get vault customer
PUT	/api/customer/{id}	Update vault customer
DELETE	/api/customer/{id}	Delete vault customer

Method	Endpoint	Description
POST	/api/recurring/plan	Create recurring plan
POST	/api/recurring/subscription	Create subscription
POST	/api/invoice	Create invoice
POST	/api/settlement/search	Search batches

Base URLs:

- Sandbox: https://sandbox.doughgateway.com
- Production: https://app.doughgateway.com *(confirm with Dough rep)*

Headers (all requests):

```
Authorization: api_YOUR_KEY
Content-Type: application/json
```

Important: Every response includes x-correlation-id header — save this for support troubleshooting.

📌 Key Architecture Notes

1. **PCI Compliance:** The Tokenizer JS iframe captures card data client-side. Raw card numbers never touch your server when using the token flow. For the sandbox test script, we use direct card data (acceptable in sandbox only).
2. **Amounts in Cents:** All amounts are in cents. $12.99 = 1299. The gateway will reject decimal amounts.
3. **Surcharge/Dual Pricing:** Use calculateAmounts() before submitting to get accurate totals including any surcharge or service fee. The base_amount field auto-calculates surcharge.
4. **Idempotency:** Pass idempotency_key (UUID) to safely retry failed requests. Default TTL is 5 minutes.
5. **Processor ID:** If your sandbox account has a default processor, you can omit processor_id. Otherwise, it's required on every transaction.