Payment Service API Documentation

API Endpoints

The Payment Service provides the following RESTful API endpoints:

Customer Management

Endpoint	Method	Description
`/payment/customer/:email`	GET	Get a customer by email
`/payment/customer`	POST	Create a new customer

Payment Method Management

Endpoint	Method	Description
`/payment/payment-method`	POST	Add a new payment method to a customer
`/payment/payment-method-details`	POST	Add a payment method with detailed card information
`/payment/payment-method`	DELETE	Delete a payment method
`/payment/payment-methods/:customerId`	GET	List all payment methods for a customer

Payment Processing

Endpoint	Method	Description
`/payment/payment-intent`	POST	Create a payment intent and process payment
`/payment/refund`	POST	Process a refund for a payment

Request & Response Formats

All API endpoints accept and return JSON data. The standard response format includes:

```
{
  "success": true/false,
  "data": { ... },  // For successful responses
  "message": "..."  // For error responses
}
```

Authentication Methods

The Payment Service accepts authentication via Bearer token in the Authorization header:

```
Authorization: Bearer <token>
```

This token is passed to the Order Service when updating order status after successful payment.

API Examples

1. Create a Customer

Request:

```
POST /payment/customer
Content-Type: application/json

{
  "email": "customer@example.com",
  "name": "John Doe"
}
```

Response:

```
{
  "success": true,
  "data": {
    "customerId": "cus_1234567890"
  }
}
```

2. Get a Customer

Request:

```
GET /payment/customer/customer@example.com
```

Response:

```
{
  "success": true,
  "data": {
    "customerId": "cus_1234567890"
  }
}
```

3. Add a Payment Method

Request:

```
POST /payment/payment-method
Content-Type: application/json

{
  "customerId": "cus_1234567890",
  "paymentMethodId": "pm_1234567890"
}
```

Response:

```
{
  "success": true,
  "data": {
    "payment_method_id": "pm_1234567890",
    "card_last4": "4242",
    "card_brand": "visa"
  }
}
```

4. Add a Payment Method with Details

Request:

```
POST /payment/payment-method-details
Content-Type: application/json

{
  "customerId": "cus_1234567890",
  "paymentMethodId": "pm_1234567890",
  "cardholderName": "John Doe",
  "last4": "4242",
  "cardType": "visa",
  "expiryDate": "12/25",
  "isDefault": true
}
```

Response:

```
{
  "success": true,
  "data": {
    "payment_method_id": "pm_1234567890",
    "card_last4": "4242",
    "card_brand": "visa",
    "cardholder_name": "John Doe",
    "expiry_date": "12/25",
    "is_default": true
  }
}
```

5. List Payment Methods

Request:

```
GET /payment/payment-methods/cus_1234567890
```

Response:

```
{
  "success": true,
  "data": [
    {
      "payment_method_id": "pm_1234567890",
      "card_last4": "4242",
      "card_brand": "visa",
      "cardholder_name": "John Doe",
      "expiry_date": "12/25",
      "is_default": true
    }
  ]
}
```

6. Delete Payment Method

Request:

```
DELETE /payment/payment-method
Content-Type: application/json

{
  "customerId": "cus_1234567890",
  "paymentMethodId": "pm_1234567890"
}
```

Response:

```
{
  "success": true,
  "data": {
    "success": true,
    "message": "Payment method deleted successfully"
  }
}
```

7. Create Payment Intent

Request:

```
POST /payment/payment-intent
Content-Type: application/json
Authorization: Bearer <token>

{
  "amount": 2500,
  "customerId": "cus_1234567890",
  "paymentMethodId": "pm_1234567890",
  "orderId": "order_1234567890",
  "currency": "usd",
  "returnUrl": "https://example.com/checkout/complete"
}
```

Response:

```
{
  "success": true,
  "data": {
    "payment_intent_id": "pi_1234567890",
    "status": "succeeded",
    "amount": 2500,
    "currency": "usd"
  }
}
```

8. Process Refund

Request:

```
POST /payment/refund
Content-Type: application/json

{
  "paymentIntentId": "pi_1234567890",
  "amount": 1000,
  "reason": "customer_requested"
}
```

Response:

```
{
  "success": true,
  "data": {
    "refund_id": "re_1234567890",
    "status": "succeeded",
    "amount": 1000,
    "currency": "usd",
    "payment_intent": "pi_1234567890"
  }
}
```

Error Responses

When an error occurs, the API returns an appropriate HTTP status code and a JSON response with error details:

```
{
  "success": false,
  "message": "Error message describing what went wrong"
}
```

Common error status codes:

- `400` - Bad Request (missing or invalid parameters)
- `404` - Not Found (resource doesn't exist)
- `500` - Internal Server Error (server-side issue)

Integration with Other Services

The Payment Service integrates with other CommuneDrop microservices in the following ways:

Order Service Integration

- Notifies Order Service when payments are successful or failed
- Receives order details and pricing information from Order Service

Location Service Integration

- Uses distance data to calculate delivery fees
- Verifies delivery completion before finalizing certain payments

Authentication Service Integration

- Validates user tokens for secure payment processing
- Ensures only authorized users can access payment information

Security Considerations

The Payment Service implements the following security measures:

1. **PCI DSS Compliance**
   - All credit card data is handled according to Payment Card Industry Data Security Standard
   - Card details are never stored directly in our databases
2. **Data Encryption**
   - All payment data is encrypted in transit using TLS 1.2+
   - Sensitive data is encrypted at rest
3. **Tokenization**
   - Card information is tokenized through Stripe
   - Only payment tokens are stored in our system
4. **Authentication**
   - All endpoints require valid JWT authentication
   - Role-based access control limits access to payment operations