### \*\*Chapter 5: Verifying Payments and Handling Refunds\*\*

In this chapter, we will explore how to verify payment status and handle refunds using Cashfree’s API. These functionalities are essential for ensuring payment success and managing customer refunds effectively.

---

#### \*\*Topic 1: Verifying Payment Status\*\*

After initiating a payment or receiving a webhook notification, it’s crucial to verify the payment status to ensure it was successfully processed. Cashfree provides an API to check the payment status for a given order.

---

##### \*\*Step 1: Cashfree Payment Verification API\*\*

Cashfree’s \*\*Order Status API\*\* allows you to check the status of a payment order using the \*\*order ID\*\*.

- \*\*API Endpoint:\*\*

```

/api/v2/order/info/status

```

- \*\*Method:\*\* `POST`

- \*\*Required Parameters:\*\*

- `orderId` (String) – The unique order ID for the transaction you want to verify.

---

##### \*\*Step 2: Implementing Payment Verification in Spring Boot\*\*

Here’s how to implement the order status check in your Spring Boot application:

1. \*\*Service Method to Verify Payment Status:\*\*

Add a method in your `PaymentService` class to call the Cashfree API for order verification.

```java

public String verifyPaymentStatus(String orderId) throws Exception {

// Prepare request payload

Map<String, String> requestBody = new HashMap<>();

requestBody.put("orderId", orderId);

// Prepare headers

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

headers.set("x-client-id", cashfreeConfig.getAppId());

headers.set("x-client-secret", cashfreeConfig.getSecretKey());

HttpEntity<Map<String, String>> request = new HttpEntity<>(requestBody, headers);

// Send POST request to Cashfree API

String url = cashfreeConfig.getBaseUrl() + "/order/info/status";

ResponseEntity<Map> response = restTemplate.postForEntity(url, request, Map.class);

if (response.getStatusCode() == HttpStatus.OK) {

// Extract and return payment status from the response

Map<String, String> responseBody = response.getBody();

return responseBody.get("txStatus");

} else {

throw new Exception("Failed to verify payment status");

}

}

```

2. \*\*Controller to Expose Verification API:\*\*

Create a REST endpoint in your `PaymentController` to allow your frontend to verify the payment status.

```java

@GetMapping("/verify/{orderId}")

public ResponseEntity<String> verifyPayment(@PathVariable String orderId) {

try {

String status = paymentService.verifyPaymentStatus(orderId);

return ResponseEntity.ok("Payment status: " + status);

} catch (Exception e) {

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body("Error verifying payment");

}

}

```

---

##### \*\*Step 3: Using Payment Verification in Frontend\*\*

Once the verification API is in place, your frontend can call this endpoint to check the payment status of a specific order.

```javascript

fetch('http://localhost:8080/api/payment/verify/ORDER12345')

.then(response => response.text())

.then(status => {

console.log('Payment Status:', status);

})

.catch(error => {

console.error('Error verifying payment:', error);

});

```

---

#### \*\*Topic 2: Handling Refunds\*\*

Cashfree also provides an API to handle refunds. This is useful when you need to refund a customer for a specific transaction.

---

##### \*\*Step 1: Cashfree Refund API\*\*

Cashfree’s \*\*Refund API\*\* allows you to initiate a refund for a successful payment.

- \*\*API Endpoint:\*\*

```

/api/v2/refund/create

```

- \*\*Method:\*\* `POST`

- \*\*Required Parameters:\*\*

- `orderId` (String) – The order ID of the transaction to be refunded.

- `refundAmount` (String) – The amount to be refunded.

- `refundId` (String) – A unique ID for the refund request (you must generate this).

- `refundNote` (String) – A note describing the reason for the refund.

---

##### \*\*Step 2: Implementing Refunds in Spring Boot\*\*

Let’s implement the refund functionality in your Spring Boot application.

1. \*\*Service Method to Handle Refunds:\*\*

Add a method in your `PaymentService` class to handle refund requests.

```java

public String initiateRefund(String orderId, String refundId, String refundAmount, String refundNote) throws Exception {

// Prepare request payload

Map<String, String> requestBody = new HashMap<>();

requestBody.put("orderId", orderId);

requestBody.put("refundAmount", refundAmount);

requestBody.put("refundId", refundId);

requestBody.put("refundNote", refundNote);

// Prepare headers

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

headers.set("x-client-id", cashfreeConfig.getAppId());

headers.set("x-client-secret", cashfreeConfig.getSecretKey());

HttpEntity<Map<String, String>> request = new HttpEntity<>(requestBody, headers);

// Send POST request to Cashfree API

String url = cashfreeConfig.getBaseUrl() + "/refund/create";

ResponseEntity<Map> response = restTemplate.postForEntity(url, request, Map.class);

if (response.getStatusCode() == HttpStatus.OK) {

// Extract refund status from response

Map<String, String> responseBody = response.getBody();

return responseBody.get("refundStatus");

} else {

throw new Exception("Failed to initiate refund");

}

}

```

2. \*\*Controller for Refund API:\*\*

Create a REST endpoint in your `PaymentController` to handle refund requests.

```java

@PostMapping("/refund")

public ResponseEntity<String> initiateRefund(@RequestBody RefundRequest refundRequest) {

try {

String refundStatus = paymentService.initiateRefund(

refundRequest.getOrderId(),

refundRequest.getRefundId(),

refundRequest.getRefundAmount(),

refundRequest.getRefundNote()

);

return ResponseEntity.ok("Refund status: " + refundStatus);

} catch (Exception e) {

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body("Error initiating refund");

}

}

```

3. \*\*Model for Refund Request:\*\*

Define a model to represent the refund request.

```java

public class RefundRequest {

private String orderId;

private String refundId;

private String refundAmount;

private String refundNote;

// Getters and setters

}

```

---

##### \*\*Step 3: Example Refund Request from Frontend\*\*

Here’s an example of how your frontend could request a refund:

```javascript

fetch('http://localhost:8080/api/payment/refund', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

orderId: 'ORDER12345',

refundId: 'REFUND67890',

refundAmount: '500',

refundNote: 'Customer request'

})

})

.then(response => response.text())

.then(data => {

console.log('Refund Status:', data);

})

.catch(error => {

console.error('Error initiating refund:', error);

});

```

---

#### \*\*What We’ve Covered So Far:\*\*

1. \*\*Verifying Payment Status:\*\* We implemented the API to check the payment status using Cashfree’s `/order/info/status` endpoint.

2. \*\*Handling Refunds:\*\* We implemented the refund process using Cashfree’s `/refund/create` endpoint.

---