**Integration of Razorpay API for Travel Software System**

**Overview**

This documentation outlines the steps necessary to integrate the Razorpay API into a travel software system. Razorpay is a popular payment gateway that allows businesses to accept online payments in multiple modes, including credit/debit cards, net banking, UPI, and wallets. This guide covers the configuration, setup, and integration processes necessary for implementing Razorpay in your travel software.

Prerequisites

Before beginning the integration process, ensure that you have the following:

* Razorpay Account: A Razorpay account with API keys.
* Travel Software System: The system where the Razorpay payment gateway will be integrated.
* Development Environment: Access to the development environment of your software, such as a local server or a staging server.
* Programming Knowledge: Basic knowledge of web development, including the language and framework used in your travel software system.

**Step 1: Set Up Razorpay Account and Obtain API Keys**

1. Create Razorpay Account:
   * Go to [Razorpay's website](https://razorpay.com) and sign up for an account if you do not have one.
   * Complete the KYC process to activate your account for live transactions.
2. Generate API Keys:
   * Navigate to the Razorpay Dashboard.
   * Go to Settings > API Keys.
   * Generate a new set of API keys. This will include a Key ID and Key Secret.
   * Keep these keys secure; you'll need them for integration.

**Step 2: Install Razorpay SDK**

Razorpay provides SDKs for various programming languages and platforms. Choose the SDK that matches your technology stack. Below are installation instructions for some common platforms:

**2.1 Python (Django/Flask)**

Install the Razorpay Python

**Step 3: Integration into the Travel Software System**

**3.1. Backend Integration**

1. **Initialize Razorpay Client**:

Use the API keys obtained earlier to initialize the Razorpay client in your application.

1. **Create an Order**:

When a user initiates a booking in your travel software, create an order in Razorpay.

1. **Send Order Details to Frontend**:

Send the order ID and other necessary details to the frontend for processing the payment

**3.2. Frontend Integration**

1. **Include Razorpay Checkout Script**:

Add the Razorpay checkout script in your frontend code.

1. **Create Checkout Options**:

Configure the checkout options with the necessary details such as the order ID, amount, and customer details.

1. **Open Razorpay Checkout**:

Call the Razorpay checkout to open the payment gateway.

**Step 4: Payment Verification**

1. **Verify Payment Signature:**

After the payment is completed, Razorpay sends a payment signature. Verify this signature on the server to ensure that the payment is legitimate.

1. **Update Booking Status**:

Once the payment is verified, update the booking status in your system and notify the user of the successful payment.

**Step 5: Handling Webhooks (Optional)**

For additional security and functionality, you can set up Razorpay webhooks to automatically handle events such as payment failures, refunds, and more.

1. **Set Up Webhooks**:
   * Go to the Razorpay Dashboard.
   * Navigate to **Settings** > **Webhooks**.
   * Create a new webhook and provide the necessary URL in your application where Razorpay will send event notifications.
2. **Handle Webhook Events**:
   * Implement a handler in your backend to process the events sent by Razorpay.

**Create an Order in Server**

Given below are the order states and the corresponding payment states:

| **Payment Stages** | **Order State** | **Payment State** | **Description** |
| --- | --- | --- | --- |
| Stage I | created | created | The customer submits the payment information, which is sent to Razorpay . The payment is **not processed** at this stage. |
| Stage II | attempted | authorized/failed | An order moves from **created** to **attempted** state when payment is first attempted. It remains in this state until a payment associated with the order is captured. |
| Stage III | paid | captured | After the payment moves to the **captured** state, the order moves to the **paid** state.   * No more payment requests are allowed after an order moves to the **paid** state. * The order continues to be in this state even if the payment for this order is **refunded**. |

**Handler Function or Callback URL**

| **Handler Function** | **Callback URL** |
| --- | --- |
| **When you use this:**   * **On successful payment, the customer is shown your web page.** * **On failure, the customer is notified of the failure and asked to retry the payment.** | **When you use this:**   * **On successful payment, the customer is redirected to the specified URL, for example, a payment success page.** * **On failure, the customer is asked to retry the payment.** |

**Note: In our case we have used Handler Method**

**Conclusion**

With the steps outlined in this guide, you should be able to successfully integrate Razorpay into your travel software system, enabling smooth and secure payment processing for your users. Be sure to test thoroughly in both sandbox and live environments before going into production.

**References**

* Razorpay Documentation
* Razorpay API Reference