LightSpeedPay API Controllers Documentation

This documentation covers the functionality and operations of key controllers used in the **LightSpeedPay** payment gateway. The following controllers manage different aspects of the transaction lifecycle, including payments, QR code generation, pricing, alerts, and callbacks.

---

1. **Alert Controller**

The `alertController.js` handles sending alerts, such as WhatsApp messages, using external APIs.

Key Functions

- **`sendWAMessage(message, to)`**:
  - Sends a WhatsApp message to a specified mobile number.
  - Constructs the API request URL and sends a `GET` request to the WhatsApp service provider.
  - Logs the response for debugging.
  - **Parameters**:
    - `message`: The message content to send.
    - `to`: The recipient's mobile number.
  - **Response**: Returns the response from the WhatsApp API.
  - **Use case**: Send alerts for transaction updates or notifications.

Code Reference

---

2. **Callback Controller**

The `callbackController.js` is responsible for handling callbacks from payment providers like Razorpay. It processes transaction updates and sends data back to the merchant.

Key Functions

- **`razorPayCallBack(req, res)`**:
  - Handles the callback from Razorpay for various events like `qr_code.credited` and `payment.authorized`.
  - Updates the transaction status in the database.
  - Processes affiliate commissions for successful transactions.
  - Sends a callback to the merchant with the updated transaction status.
  - **Use case**: Manage Razorpay callbacks and update the transaction lifecycle.

Code Reference

---

3. **Pricing Controller**

The `pricingController.js` manages pricing plans for merchants. It allows for creating, updating, and deleting pricing plans.

Key Functions

- **`createPrice(req, res)`**:
    - Creates a new pricing plan for a merchant.
    - Validates if the merchant exists and whether a pricing plan already exists.
    - Stores the pricing details and returns the created data.
    - **Use case**: Manage pricing details for merchants.
- **`getPriceByMerchant(req, res)`**:
    - Retrieves pricing details for a specific merchant.
    - Provides all pricing details if the user is an admin.
- **`updatePricingPlan(req, res)`**:
    - Updates the pricing plan for a merchant.
    - **Parameters**: `priceId`, `isStartup`, `numberOfFreebies`, `price`.
- **`deletePricingPlan(req, res)`**:
    - Deletes a pricing plan for a merchant.
    - **Parameters**: `priceId`.

Code Reference

---

4. **PayU Controller**

The `payUController.js` manages UPI transactions via the PayU payment gateway. It handles QR code creation, transaction status updates, and callback processing.

Key Functions

- **`createPayuQrCode(transactionId, amount)`**:
    - Generates a UPI QR code for a transaction using the PayU gateway.
    - Constructs a hash for the PayU payment request using merchant credentials.
    - Sends a POST request to PayU and returns the UPI intent string for payment.
- **`successCallbackCapture(req, res)`**:
    - Handles success callbacks for PayU payments.
    - Updates the transaction status and triggers the affiliate commission process.
    - Sends a callback to the merchant.
- **`failedCallBack(req, res)`**:
    - Marks a transaction as failed upon receiving a failure callback from PayU.
- **`getMerchantKeyAndSalt(trxnId)`**:
    - Retrieves merchant credentials (key and salt) for a given transaction.

- **Use case**: Retrieve PayU merchant details securely.

Code Reference

---

5. **Razorpay Controller**

The `razorpayController.js` manages UPI payments and QR code generation using Razorpay. It processes payments, decodes QR codes, and updates transaction statuses.

Key Functions

- **`createQrCode(params)`**:
  - Generates a Razorpay UPI QR code for a transaction.
  - Downloads the QR code image, reads it using `Jimp`, and extracts the UPI string.
  - Updates the UPI string with transaction-specific details.
  - **Use case**: Initiate UPI payments using Razorpay.
- **`fetchLast10MinsPayments(req, res)`**:
  - Retrieves all payments made in the last 10 minutes from Razorpay.
  - Updates the transaction status in the database.
- **`readQrCode(filename)`**:
  - Reads a QR code image file and extracts the UPI string.
- **`updateUPIString(upiString, updates)`**:
  - Updates specific fields of a UPI string with new values (e.g., transaction details).

Code Reference

Conclusion

The controllers documented above are integral to the functioning of the LightSpeedPay payment gateway. They cover a wide range of functionalities, from managing payment gateways like Razorpay and PayU, to handling affiliate commissions, pricing plans, and transaction status updates. Each controller interacts with other controllers and models to ensure smooth operation across the entire payment lifecycle.