JavaScript SDK

Overview

The Checkout SDK provides a simplified way for businesses to accept payment from their customers. The SDK offers three primary methods of integration: **Redirect**, **Iframe**, and **Modal**. This allows developers to choose the integration method that best suits their needs.

Installation

To use the Hubtel Checkout SDK to collect payment, install the NPM Package or include the the CDN link in your HTML file:

NPM | CDN

```bash
npm i @hubteljs/checkout
```

Methods of Integration

The JavaScript SDK supports three primary methods of integration:

1. Redirect Integration: This method opens the checkout page in a new tab or window.

2. Iframe Integration: This method embeds the checkout page within an iframe on your webpage.

3. Modal Integration: This method opens the checkout page in a modal popup.

Pre-Checkout

> **Note**
>
> Before calling the Checkout SDK, most apps would call a Pre-Checkout API provided by their backend team to create the order for the client app to handle the payment. The implementation might differ. Also note that clientReferences must never be duplicated for any transaction. **DO NOT** add 'Basic' to your encoded credentials.

Sample Pre-checkout Request:

```json
{
  "amount": "1",
  "phoneNumber": "0200000000"
}
```

Sample Pre-checkout Response:

```json
{
  "message": "success",
  "code": "200",
```

```
  "data": {
    "description": "Payment of GHS 1.00 for (18013782) (MR JOHN DOE-233200000000) ",
    "clientReference": "f03dbdcf8d4040dd88d0a82794b0229f",
    "amount": 1.0,
    "customerMobileNumber": "233200000000",
    "callbackUrl": "https://yourcallbackurl.com"
  }
}
```

The response from the Pre-Checkout API usually contains all information needed for a successful checkout. Once retrieved the data is then passed onto the Hubtel Checkout SDK to begin the checkout.

Redirect Integration

The redirect method opens the checkout page in a new tab or window. This is done by calling the **redirect** method of the Checkout SDK instance and passing the purchase information and configuration options as parameters.

Example Usage

javascript Copy

```javascript
// Import the Checkout SDK if you are using the npm package
import CheckoutSdk from "@hubteljs/checkout";

// Initialize the Checkout SDK
const checkout = new CheckoutSdk();

//Purchase information
const purchaseInfo = {
  amount: 50,
  purchaseDescription: "Payment of GHS 50.00 for (18013782) (MR JOHN DOE-233000000000)",
  customerPhoneNumber: "233000000000",
  clientReference: "unique-client-reference-12345",
};

// Configuration options
const config = {
  branding: "enabled",
  callbackUrl: "https://yourcallbackurl.com",
  merchantAccount: 11334,
  basicAuth: "your-basic-auth-here",
};

// A function to redirect to the payment page
const redirect = () => {
  checkout.redirect({
    purchaseInfo,
    config,
  });
};
```

Iframe Integration

The iframe integration method embeds the checkout page within an iframe on your webpage. This method is suitable for developers who want to keep their users on the same page while making a payment.

Example Usage

In your HTML file, add a div element with the id **hubtel-checkout-iframe** where you want the checkout to be rendered.

html Copy

```html
<body>
  ...
```

```
  <!-- Add this element to where you want the checkout to be rendered. By default the checkout will use the available  height and width of it's container -->

    <div id="hubtel-checkout-iframe"></div>
  ...

</body>
```

In your JavaScript file, call the **initIframe** method of the Checkout SDK instance and pass the purchase information, configuration options, iframe style and callBacks for event handling as parameters.

javascript Copy

```javascript
// Import the Checkout SDK if you are using the npm package
import CheckoutSdk from "@hubteljs/checkout";

// Initialize the Checkout SDK
const checkout = new CheckoutSdk();

//Purchase information
const purchaseInfo = {
  amount: 50,
  purchaseDescription:
    "Payment of GHS 50.00 for (18013782) (MR JOHN DOE-233000000000)",
  customerPhoneNumber: "233000000000",
  clientReference: "unique-client-reference-12345",
};

// Configuration options
const config = {
  branding: "enabled",
  callbackUrl: "https://yourcallbackurl.com",
  merchantAccount: 11334,
  basicAuth: "your-basic-auth-here",
};

const iframeStyle = {
    width: '100%',
    height: '100%',
    border: 'none',
  };

// A function to render the iframe on the page where the div with id 'hubtel-checkout-iframe' is located
const openIframe = () => {
    checkout.initIframe({ purchaseInfo, config, iframeStyle, callBacks: {
      onInit: () => console.log('Checkout initialize Initialized'),
      onPaymentSuccess: (data) => console.log('Payment Success', data),
      onPaymentFailure: (data) => console.log('Payment Failure', data),
      onLoad: () => console.log('Iframe Loaded'),
      onFeesChanged: (fees) => console.log('Fees Changed', fees),
      onResize: (size) => console.log('Iframe Resized', size?.height),
    }})
};
```

Modal Integration

The modal integration method opens the checkout page in a modal popup. This is done by calling the **openModal** method of the Checkout SDK instance and passing the purchase information, configuration options and callBacks for event handling as parameters.

Example Usage

javascript Copy

```javascript
// Import the Checkout SDK if you are using the npm package
```

```
import CheckoutSdk from "@hubteljs/checkout";

// Initialize the Checkout SDK
const checkout = new CheckoutSdk();

// Purchase information
const purchaseInfo = {
  amount: 50,
  purchaseDescription:
    "Payment of GHS 5.00 for (18013782) (MR SOMUAH STA ADANE-233557913587)",
  customerPhoneNumber: "233557913587",
  clientReference: "unique-client-reference-12345",
};

// Configuration options
const config = {
  branding: "enabled",
  callbackUrl: "https://yourcallbackurl.com",
  merchantAccount: 11334,
  basicAuth: "your-basic-auth-here",
};

// A function to open the payment modal
const initPay = () => {
  checkout.openModal({
    purchaseInfo,
    config,
    callBacks: {
      onInit: () => console.log("Iframe initialized: "),
      onPaymentSuccess: (data) => {
        console.log("Payment succeeded: ", data);
        // You can close the popup here
        checkout.closePopUp();
      },
      onPaymentFailure: (data) => console.log("Payment failed: ", data),
      onLoad: () => console.log("Checkout has been loaded: "),
      onFeesChanged: (fees) =>
        console.log("Payment channel has changed: ", fees),
      onResize: (size) =>
        console.log("Iframe has been resized: ", size?.height),
      onClose: (size) => console.log("The modal has closed ", size?.height),
    },
  });
};
```

Parameters

Parameter	Type	Description
purchaseInfo	Object	Contains details about the purchase.
amount	Number	The purchase amount.
purchaseDescription	String	A brief description of the purchase.
customerPhoneNumber	String	The customer’s phone number.
clientReference	String	A unique reference for the purchase.
config	Object	Contains configuration options.
branding	String	Set to **"enabled"** to show the business name and logo or **"disabled"** to hide them.
callbackUrl	String	The URL to which the SDK will send the response.
merchantAccount	Number	The merchant account ID. Find this here.

Parameter	Type	Description
basicAuth	String	Basic auth credentials. Find this here.
integrationType	String	Specifies the integration type. Default is "External" which is the type used for external integration.
callacks	Object	Callback functions for various events.
onInit	Function	Called when the iframe is initialized.
onPaymentSuccess	Function	Called when the payment is successful.
onPaymentFailure	Function	Called when the payment fails.
onLoad	Function	Called when the iframe is loaded.
onFeesChanged	Function	Called when the fees change.
onClose	Function	Called when the the modal is closed.
onResize	Function	Called when the iframe is resized.

Transaction Status Check

It is **mandatory** to implement the **Transaction Status Check API** as it allows merchants to check for the status of a transaction in rare instances where a merchant does not receive the final status of the transaction from Hubtel **after five (5) minutes**.

To check the status of a transaction, send an HTTP **GET** request to the below URL, with either one or more unique transaction identifiers as parameters.

It is also **mandatory** to parse your **POS Sales ID** for Status Check requests in the endpoint. Find your **POS Sales ID** here.

Note

Only requests from whitelisted IP(s) can reach the endpoint. Requests from IP addresses that have not been whitelisted will return a **403 Forbidden** error response or a **timeout**. Submit your public IP(s) to your Retail Systems Engineer to be whitelisted.

We permit a maximum of 4 IP addresses per service.

API Endpoint	https://api-txnstatus.hubtel.com/transactions/{POS_Sales_ID}/status
Request Type	**GET**
Content Type	**JSON**

REQUEST PARAMETERS

Parameter	Type	Requirement	Description
clientReference	String	Mandatory (preferred)	The client reference of the transaction specified in the request payload.
hubtelTransactionId	String	Optional	Transaction ID from Hubtel after successful payment.
networkTransactionId	String	Optional	The transaction reference from the mobile money provider.

Note

Although either one of the unique transaction identifiers above could be passed as parameters, **clientReference** is recommended to be used often.

SAMPLE REQUEST

HTTP

Copy

```
GET /transactions/11684/status?clientReference=fhwrthrthejhjmt HTTP/1.1
Host: api-txnstatus.hubtel.com
Authorization: Basic QmdfaWghe2Jhc2U2NF9lbmNvZGUoa2hzcW9seXU6bXVhaHdpYW8pfQ==
```

RESPONSE PARAMETERS

Parameter	Type	Description
message	String	The description of response received from the API that is related to the ResponseCode.
responseCode	String	The response code of the API after the request.
data	Object	An object containing the required data response from the API.
date	String	Date of the transaction
status	String	Status of the transaction i.e.: **Paid**, **Unpaid** or **Refunded**.
transactionId	String	The unique ID used to identify a Hubtel transaction (**from Hubtel**).
externalTransactionId	String	The transaction reference from the mobile money provider (**from Telco**).
paymentMethod	String	The mode of payment.
clientReference	String	The reference ID that is initially provided by the client/API user in the request payload (**from merchant**).
currencycode	String	Currency of the transaction; could be **null**.
amount	Float	The transaction amount.
charges	Float	The charge/fee for the transaction.
amountAfterCharges	Float	The transaction amount after charges/fees deduction.
isFulfilled	Boolean	Whether service was fulfilled; could be **null**.

SAMPLE RESPONSE (Paid)

200 OK

Copy


```
{
    "message": "Successful",
    "responseCode": "0000",
    "data": {
        "date": "2024-04-25T21:45:48.4740964Z",
        "status": "Paid",
        "transactionId": "7fd01221faeb41469daec7b3561bddc5",
        "externalTransactionId": "0000006824852622",
        "paymentMethod": "mobilemoney",
        "clientReference": "1sc2rc8nwmchngs9ds2f1dmn",
        "currencyCode": null,
        "amount": 0.1,
        "charges": 0.02,
        "amountAfterCharges": 0.08,
        "isFulfilled": null
    }
}
```

SAMPLE RESPONSE (Unpaid)

200 OK

Copy


```
{
    "message": "Successful",
    "responseCode": "0000",
    "data": {
        "date": "2024-04-25T21:45:48.4740964Z",
        "status": "Unpaid",
        "transactionId": "7fd01221faeb41469daec7b3561bddc5",
        "externalTransactionId": "0000006824852622",
        "paymentMethod": "mobilemoney",
        "clientReference": "1sc2rc8nwmchngs9ds2f1dmn",
        "currencyCode": null,
        "amount": 0.1,
        "charges": 0.02,
        "amountAfterCharges": 0.08,
        "isFulfilled": null
    }
}
```


Response Codes

The Hubtel Sales API uses standard HTTP error reporting. Successful requests return HTTP status codes in the 2xx. Failed requests return status codes in 4xx and 5xx. Response codes are included in the JSON response body, which contain information about the response.

Response Code	Description	Required Action
0000	This ResponseCode in the callback means the transaction has been processed successfully.	None
2001	• Transaction failed due to an error with the Payment Processor. Please review your request or retry in a few minutes • The MTN Mobile Money user has reached counter or balance limits, has insufficient funds or is missing permissions. • MTN Exception: Account Holder with FRI Not Found • Transaction id is invalid • Decline - General decline of the card. No other information provided by the issuing bank.	• Customer either entered no or invalid PIN • Mobile network not able to parse your request • USSD session timeout • Having strange characters (&*!%@) in your descriptio • Ensure that the number provided matches the chann
4000	Validation errors	Validation errors. Something is not quite right with this reques
4070	We're unable to complete this payment at the moment. Please try again later. Fees not set for given conditions.	Ensure you are passing the required minimum amount or cor