Bakong QR Pay Integration

DOCUMENT HISTORY

Version	Date	Prepared By	Description
1.0	28.12.2020	Moeung Theara	
1.0.1	28.01.2021	Moeung Theara	- Adding SDK system interaction flow. - Update product flow diagram. - Adding header parameter in api document.

Table of Contents

Abbreviation **2**

Introduction **3**

Overview 3

Purpose 3

Scope 3

Product Flow **3**

How it works 4

System Interaction Flow 5

Integration Process **9**

Prerequisites: 9

EMVCo QR Generation: 9

Implementing API: 11

Abbreviation

Abbreviation	Meaning
API	Application Programming Interface
QR Code	Refer to the Quick Response code for Cambodia which compiled with standard QR code called KHQR.
Customer	Refer to individual who use bakong/bank app to make payment via Bakong.

Introduction

Overview

QR code is the best option of payment for the quick response. Consumers can pay for their goods/services by using any mobile apps to scan QR code that are displayed by merchants in a faster and easier way. So, let’s start integration with Bakong dynamic QR code payments which offer more security than credit cards, as several high-profile data breaches have illustrated due to all the transferred data being encrypted.

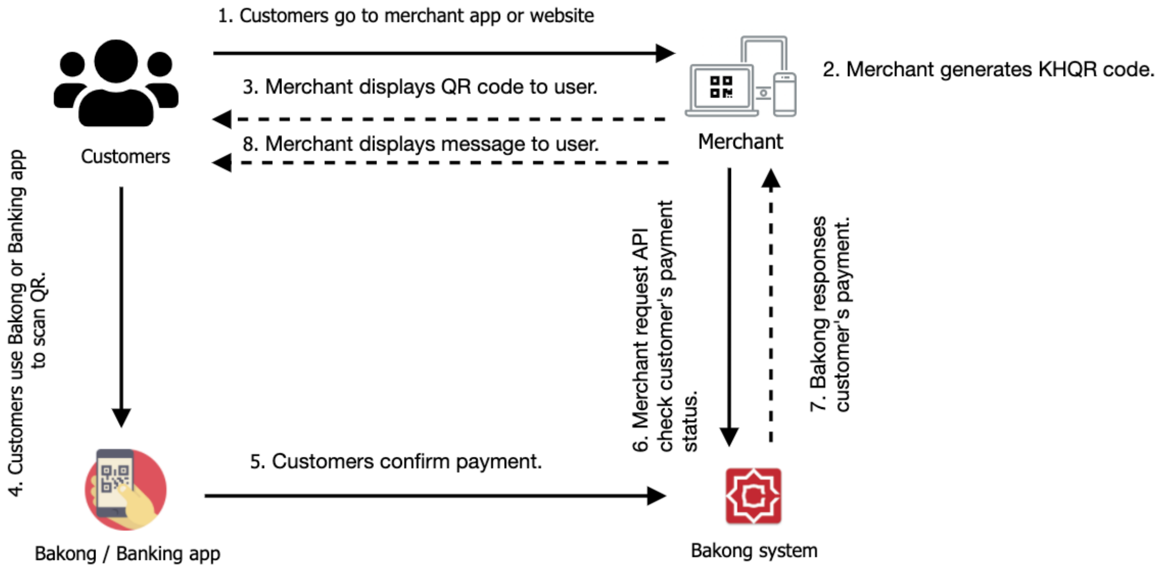
Purpose

This document describes the detailed specification of how to integrate Bakong dynamic QR code payment which is offered by National Bank of Cambodia. The expected readers are NBC technical team, third-party technical, POS service provider, software developers, etc. This can be used as reference for any interest related to the KHQR Integration.

Scope

This document contains the complete description of the QR pay integration specification including: Product Flow, System interaction flow, Integration Process and API List.

Product Flow


1. Customers go to merchant app or website
2. Merchant generates KHQR code.
3. Merchant displays QR code to user.
8. Merchant displays message to user.
Customers
Merchant
4. Customers use Bakong or Banking app to scan QR.
6. Merchant request API check customer's payment status.
7. Bakong responses customer's payment.
5. Customers confirm payment.
Bakong / Banking app
Bakong system

How it works

1. Customers make payment via a merchant app, website, or dynamic invoice.

2. Merchant generates KHQR code by using KHQR SDK which is provided by the **National bank of Cambodia** or they can generate QR by themself following the QR EmvCo standard at their back-end side then respond to the client side via API.

3. Merchant displays QR to the user along with expiration time.

4. Users scan QR code via the Bakong/Bank app that supports KHQR (In general all Bakong members support it).

5. Users confirm payment. After the payment succeeds, the user will get the notification.

6. Merchant set time interval to request API to check transaction of customer's payment at Bakong side in specific duration by following QR expiration time (it shall complete within 10s at most or the merchant can go back to manual check with their customer by asking to show the receipt within Bakong/Bank app).

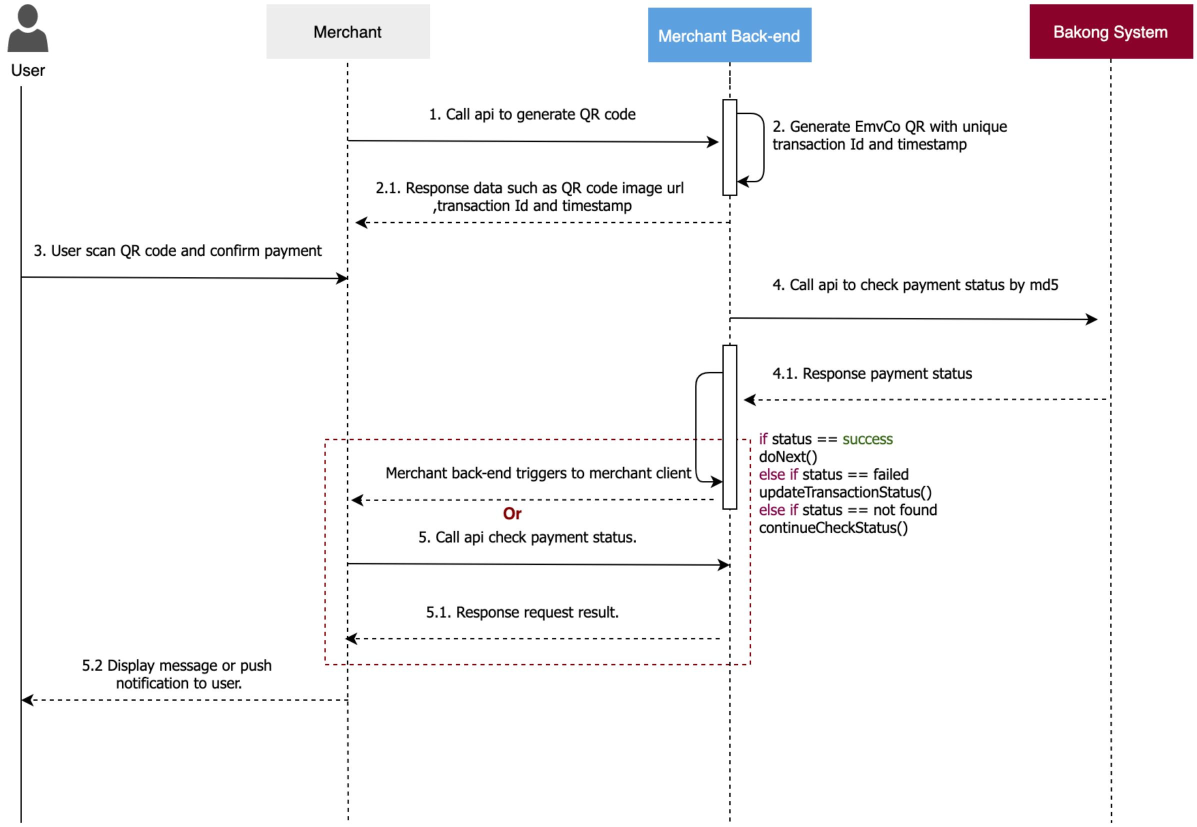
7. Bakong response payment status to merchant such as success, failed or expired.

8. Merchant displays messages to the customer base on status success, failed or expired.

System Interaction Flow

This section provides high-level information about the system interaction by 2 options as below

➢ **Generate QR at merchant back-end side:**


User
Merchant
Merchant Back-end
Bakong System
1. Call api to generate QR code
2. Generate EmvCo QR with unique transaction Id and timestamp
2.1. Response data such as QR code image url ,transaction Id and timestamp
3. User scan QR code and confirm payment
4. Call api to check payment status by md5
4.1. Response payment status
Merchant back-end triggers to merchant client
Or
5. Call api check payment status.
5.1. Response request result.
5.2 Display message or push notification to user.

```
if status == success
doNext()
else if status == failed
updateTransactionStatus()
else if status == not found
continueCheckStatus()
```

1. Users go to the merchant app/website to make payment then merchants call API to their own integration back-end to create QR code.

2. Merchants generate EMVCo QR code with unique transaction id, timestamp and any additional data if needed. Then response data back to the merchant such as QR code image URL, transaction id, QR expiration datetime , Channel name...

Note:

- Expiration datetime can be defined based on integrator business.
- Transaction Id must be unique for every time you generate QR code.

2.1 The merchant displays QR code.

3. Users use the Bakong/Banking app to scan QR and make payments. After payment is successful, Bakong will notify the user.

4. Merchants set a time interval for a specific second to call API Get transaction with MD Hash to the Bakong system to check payment results. In case payment status is still not found and it reaches QR expiration time then kill request and update payment status.

   Note:

   - MD hash generated from md5 encryption of the whole QR string.
   - The timer along with the time-out which shall not exceed 10mins. Or you may need to go back to the manual process by checking if the customer money is deducted or not? by asking the customer to show the transaction on their mobile app.

   4.1 Bakong Response payment result to system integrator.

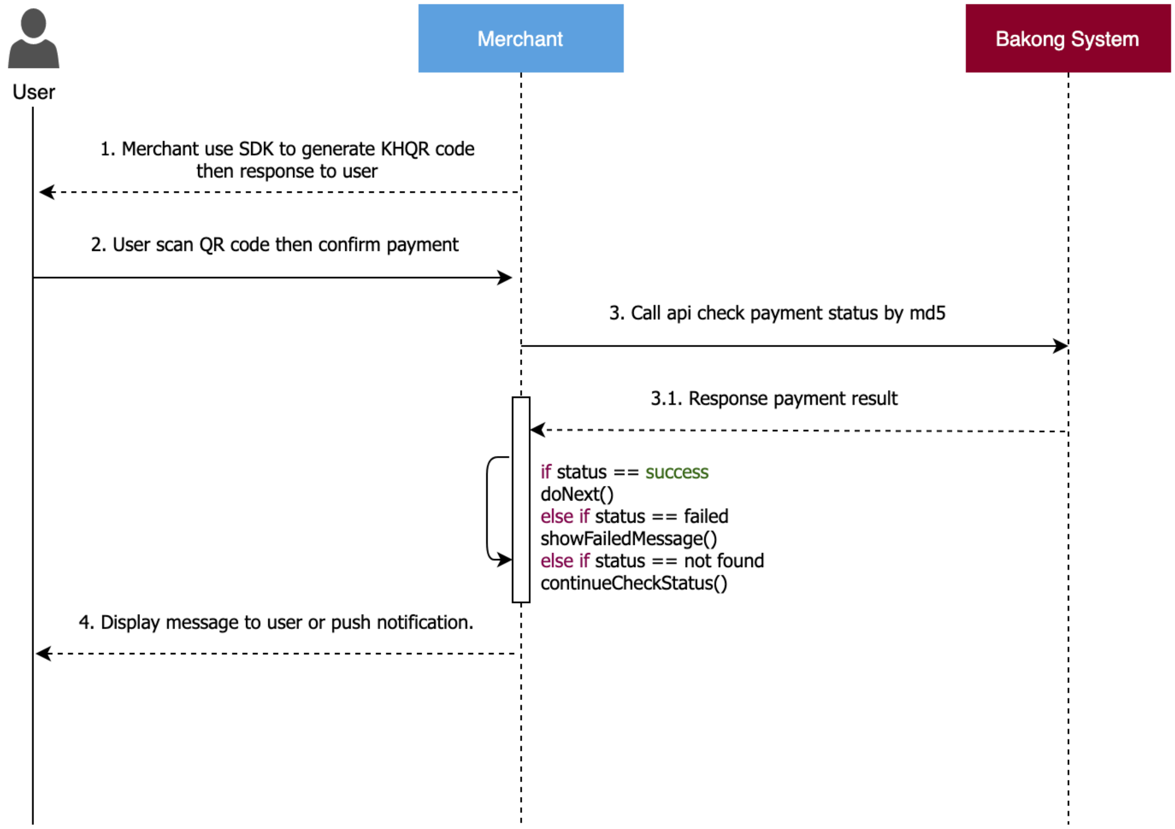
   - If status is **success** or **failed** then integrator back-end updates transaction status and can perform any additional action at their side such as print-out the bill or display a message to the user.
   - else if status is **not found** and QR **doesn't expire** then still keep process checking payment status to Bakong with api mentioned in step 4 above.
   - else if status is **not found** and QR **expired** then integrator back-end updates transaction status and shows a message to users to retry payment again with new QR code.

   After getting response from the Bakong side, Merchants can trigger their merchant clients.

5. If there is no trigger function from the merchant back-end then merchant clients have to call API to check result status to their own system integrator in every specific second if status is **success** , **failed** or expired, display message to the user. else if status is not found and QR doesn't expire then still keep process checking payment status.

   5.1 Integrator system response request result to merchant app.
   5.2 Merchants display messages to users based on request result success, failed or expired. Moreover they also can push notifications to the user.

➢ Generate QR at the client side by using KHQR SDK:


Merchant
Bakong System
User
1. Merchant use SDK to generate KHQR code then response to user
2. User scan QR code then confirm payment
3. Call api check payment status by md5
3.1. Response payment result
if status == success
doNext()
else if status == failed
showFailedMessage()
else if status == not found
continueCheckStatus()
4. Display message to user or push notification.

1. Users go to the merchant store/app/website to make payment then merchants use KHQR sdk which is provided by **National Bank of Cambodia** to generate and display QR code to users.

   Note:
   - For display QR code should have an expiration time for making payment.
   - QR expiration time can be defined based on integrator business.

2. Users use the Bakong/Banking app to scan QR and make payments. After payment is successful, Bakong will notify the user.

3. Merchant sets a time interval for a specific second to call API Get transaction with MD Hash to Bakong system to check payment result. In case payment status is still not found and it reaches QR expiration time then kill request and update payment status.

   Note:
   - The request timer for calling API check payment status can be defined by your own.

- The timer along with the time-out which shall not exceed 10mins. Or you may need to go back to the manual process by checking if the customer money is deducted or not? by viewing the screen of the customer mobile app.

3.1. Bakong response payment results to merchants.

- If status is **success** or **failed** then integrator back-end updates transaction status then can perform any additional action at their side such as display message to user.
- else if status is **not found** and QR **doesn't expire** then still keep process checking payment status to Bakong with api mentioned in step 4 above.
- else if status is **not found** and QR **expired** then integrator back-end updates transaction status and shows a message to users to retry payment again with new QR code.

4. Merchants display messages to users based on request result success, failed or expired. Moreover they also can push notifications to the user.

Integration Process

Prerequisites:

Before starting integration with Bakong QR code payment function, Please ready for some follow preparations as below:

- Bakong account which is registered under commercial bank.
- Read API specification and integration flow. Be cleared about the process.

EMVCo QR Generation:

Please refer to the document KHQR Guideline.
For more understanding about EMVco QR please visit
https://www.emvco.com/emv-technologies/qrcodes/

Additional Data

- For additional Payload Data Objects

Data Object	Meaning
Bill Number	The invoice number or bill number. This number could be provided by the merchant or could be an indication for the mobile application to prompt the consumer to input a Bill Number. For example, the Bill Number may be present when the QR Code is used for bill payment.
Mobile Number	The mobile number could be provided by the merchant or could be an indication for the mobile application to prompt the consumer to input a Mobile Number. For example, the Mobile Number to be used for multiple use cases, such as mobile top-up and bill payment.
Store Label	A distinctive value associated with a store. This value could be provided by the merchant or could be an indication for the mobile application to prompt the consumer to input a Store Label. For example, the Store Label may be displayed to the consumer on the mobile application identifying a specific store.

Loyalty Number	Typically, a loyalty card number. This number could be provided by the merchant, if known, or could be an indication for the mobile application to prompt the consumer to input their Loyalty Number.
Reference Label	Any value as defined by the merchant or acquirer in order to identify the transaction. This value could be provided by the merchant or could be an indication for the mobile app to prompt the consumer to input a transaction Reference Label. For example, the Reference Label may be used by the consumer mobile application for transaction logging or receipt display.
Customer Label	Any value identifying a specific consumer. This value could be provided by the merchant (if known), or could be an indication for the mobile application to prompt the consumer to input their Customer Label. For example, the Customer Label may be a subscriber ID for subscription services, a student enrolment number, etc.

- For Adding Transaction Id

Data Object	Printable Format	Meaning
Unreserved template (ID "91") . Globally Unique Identifier . Merchant Account Information	"9132” “0016A011223344998877” “0105Smart"	Globally Unique Identifier = A011223344998877 Third-party company name

- For Adding TimeStamp

Data Object	Printable Format	Meaning
Unreserved template (ID "99") Globally Unique Identifier	“9917” “0013161406568381”	“99” is an Id, “17” is the length of globally unique identifier. “00” is sub Id, ”13” is the lenght of timestamp, “161406568381” is timestamp value.

- Additional Data Field Template

Name (Format; ID; Length)	Template	Description	Values
Mobile Number F: ans ID: "02" L: var. up to "25"	"62"	Mobile phone number to be used for multiple use cases, such as mobile top- up and bill payment.	Please refer to . 4.8
Purpose of Transaction F: ans ID: "08" L: var. up to "25"	"62"	Any value as defined by the merchant or acquirer in order to define the purpose of the transaction.	Please refer to . 4.8
Reference Label F: ans ID: "05" L: var. up to "25"	"62"	Any value as defined by the merchant or acquirer in order to identify the transaction.	Please refer to . 4.8
Store Label F: ans ID: "03" L: var. up to "25"	"62"	A distinctive number associated with a store.	Please refer to . 4.8
Terminal Label F: ans ID: "07" L: var. up to "25"	"62"	A distinctive number associated with a terminal in the store.	Please refer to . 4.8
Bill Number F: ans ID: "01" L: var. up to "25"	"62"	The invoice number or bill number.	Please refer to . 4.8

Example:

00020101021229300012D156000000000510A93FO3230Q31280012D1560000000103081
234567852044111580 2CN5914BEST TRANSPORT6007BEIJING64200002ZH0104 最佳运 输
0202 北京
540523.7253031565502016233030412340603***0708A60086670902ME91320016A011223
3449988770708123456786304A13A

Implementing API:

For getting transaction payment result with md5 and hash, please find the API document detail as below:

Name	Method	URL	Description

Check transaction by md5	POST	{{url}}/v1/check_transaction_by_md5	Checking the transaction using 32 length string hash.
Check transaction by hash	POST	{{url}}/v1/check_transaction_by_hash	Checking the transaction by hash

HTTP code status

Code	Status
200	Accepted
400	Bad Request
401	Unauthorization
404	Not Found
500	Internal Server Error

Custom error code status

Response Code	Status
0	Success
1	Failed

Check transaction by md5

API Specifications

HTTP Method	POST
Name	Check transaction by md5
Endpoint	{{url}}/v1/check_transaction_by_md5
Description	

Request

- Request Parameters

Parameter name	Type	Mandatory	Description
Header parameters			
Authorization	String	1	Must be “Bearer <access token>”
Content-Type	String	1	Must be “application/json”
Body Parameters			
md5	String	1	md5 hash got from QR string encryption. **Validation** - Min: 1 - Max: 255

Response

- Response Parameter

Parameter	Data Type	Mandatory	Description
responseCode	int	1	
responseMessage	string	1	
data	object	1	

- Data object

Parameter	Data Type	Mandatory	Description
hash	string	1	
fromAccountId	string	1	
toAccountId	string	1	
currency	string	1	
amount	float	1	
description	string	0	

- Sample Response data

Success

```
{
  "responseCode": 0,
  "responseMessage": "Getting transaction successfully.",
  "data": {
    "hash": "8465d722d7d5065f2886f0a474a4d34dc6a7855355b611836f7b6111228893e9",
    "fromAccountId": "rieu_dhqj_1984@devb",
    "toAccountId": "bridge_account@devb",
    "currency": "USD",
    "amount": 1.0,
    "description": "testing bakong generator"
  }
}
```

Failed

```
{
   "data": null,
   "errorCode": 1,
  "responseCode": 1,
   "responseMessage": "Transaction failed.”
}
```

Not Found

```
{
  "data": null,
  "errorCode": 1,
  "responseCode": 1,
  "responseMessage": "Transaction could not be found. Please check and try again."
}
```

Check transaction by hash

API Specifications

HTTP Method	POST
Name	Check transaction by hash
Endpoint	{{url}}/v1/check_transaction_by_hash
Description	

Request

- Request Parameters

Parameter name	Type	Mandatory	Description
Header parameters			
Authorization	String	1	Must be “Bearer <access token>”
Content-Type	String	1	Must be “application/json”
Body Parameters			
hash	String	1	**Validation** - Min: 1 - Max: 255

Response

- Response Parameter

Parameter	Data Type	Mandatory	Description
responseCode	int	1	
responseMessage	string	1	
data	object	1	

- Data object

Parameter	Data Type	Mandatory	Description
hash	string	1	
fromAccountId	string	1	
toAccountId	string	1	
currency	string	1	
amount	float	1	
description	string	0	

- Sample Response sample

```
Success
{
  "responseCode": 0,
  "responseMessage": "Getting transaction successfully.",
  "data": {
    "hash": "8465d722d7d5065f2886f0a474a4d34dc6a7855355b611836f7b6111228893e9",
    "fromAccountId": "rieu_dhqj_1984@devb",
    "toAccountId": "bridge_account@devb",
    "currency": "USD",
    "amount": 1.0,
    "description": "testing bakong generator"
  }
}
```

```
Failed
{
   "data": null,
   "errorCode": 1,
  "responseCode": 1,
   "responseMessage": "Transaction failed.”
}
```

```
Not Found
{
  "data": null,
  "errorCode": 1,
  "responseCode": 1,
  "responseMessage": "Transaction could not be found. Please check and try again."
}
```

End of Document