KHQR SDK Integration

DOCUMENT HISTORY

Version	Date	Prepared By	Description
1.0	14.12.2020	Moeung Theara	
1.0.1	04.01.2021	Moeung Theara	- Alignment content. - Remove the Error Handling part. - Make all contents in Requirement, Installation and Usage part more consistent.
1.1	25.01.2021	Moeung Theara	- Add a timestamp to the QR code string. - Add md5 hash string to response data.
1.1.1	01.02.2021	Moeung Theara	- Add Bill Number, Store Label, Terminal Label in QR
1.2	17.02.2021	Moeung Theara	Adding decode KHQR information function.

Table of Contents

Introduction

Overview

The standardization of KHQR code specification will help promote wider use of mobile retail payments in Cambodia and provide consistent user experience for merchants and consumers. It can enable interoperability in the payment industry. A common QR code would facilitate payments among different schemes, e-wallets and banks and would encourage small merchants to adopt KHQR code as payment method.

KHQR is created for retail or remittance in Cambodia and Cross-Border. It only requires a single QR for receiving transactions from any payment provider through Bakong including Bakong App. For more detail please refer to **Prokas KHQR Code Specification** in Cambodia.

Purpose

This document describes the detailed specification of how to use KHQR SDK offered by **National Bank of Cambodia**. The expected readers are NBC technical team and third-party technical team. This can be used as reference for any interest related to the KHQR SDK.

Scope

This document contains the complete description of the KHQR SDK specification including: Features, Requirement, Installation Guide, Usage and FAQ.

Control Version

Version	SDK	Description
0.1.1	- iOS - Java for android - C#	
1.0.0.1	- iOS - Java for android - C#	- Add a timestamp to QR code string. - Add md5 hash string to response data.
1.0.0.2	- iOS - Java for android - C#	- Add Bill number, Store label and terminal label to QR
1.0.0.3 (**current**)	- iOS - Java for android - C#	Add decode KHQR information function.

Features

- Generate KHQR.
- Verification (Valid or Invalid).
- Decode KHQR Information.

Requirement

iOS

- Development target 11.0 +
- Dependency management using Cocoapod

Java for android

- Minimum sdk version 16
- Target sdk version 30
- Source compatibility Java 8
- Target compatibility Java 8

C#

- NetStandard 1.6
- .Net Framework 4.0 and later
- .NET Core 1.0 and later
- Xamarin.iOS
- Xamarin.Android

Installation

iOS

CocoaPods

- **How to**

    1. Add source to podfile

    ```
    source "https://sambo:ycfXmxxRbyzEmozY9z6n@gitlab.nbc.org.kh/khqr/khqr-ios-pod.git"
    ```

    2. Add pod name to podfile

    ```
    pod "BakongKHQR"
    ```

    3. Run command

    ```
    pod install
    ```

Java for android

Gradle

- **How to**

1. Add the token to $HOME/.gradle/gradle.properties

```
authToken=-XdUnXqa1VrUkaS-Xz-f
```

2. Add the JitPack repository to your root build.gradle

```
allprojects {
            repositories {
              maven {
         url 'https://jitpack.io'
         credentials { username authToken }
      }
  }
}
```

3. Add the dependency

```
dependencies {
          implementation 'kh.org.nbc.gitlab.khqr:sdk-java:1.0.0.3
}
```

C#

Nuget

- **How to**

    1. Using Visual Studio Package Manager

    ```
    Install-Package Kh.Org.Nbc.BakongKHQR
    ```

    2. Using Dotnet Cli

    ```
    dotnet add package Kh.Org.Nbc.BakongKHQR
    ```

    3. Include in csproj

    ```
    <ItemGroup>
      <PackageReference Include="Kh.Org.Nbc.BakongKHQR" Version="1.0.0.3" />
      <!-- ... other packages goes here -->
    </ItemGroup>
    ```

Install Manually

If you prefer not to use nuget, you can integrate this SDK into your project manually via dll file.

- **How to**

    1. Go to Nuget page of Kh.Org.Nbc.BakongKHQR
       https://www.nuget.org/packages/Kh.Org.Nbc.BakongKHQR
    2. Download package (nupkg file).
    3. Extract the package using any unzip tool.
    4. Copy dll file based on your platform from the following folder.

       For .NetFramework 4.0 copy dll from this path

       ```
       lib/net40/kh.org.nbc.bakong_khqr.dll
       ```

For .NetFramework 4.5 copy dll from this path

```
lib/net45/kh.org.nbc.bakong_khqr.dll
```

For .NetFramework 4.6 copy dll from this path

```
lib/net46/kh.org.nbc.bakong_khqr.dll
```

Other Platforms (.NetStandard 1.6) copy from this path

```
lib/netstandard1.6/kh.org.nbc.bakong_khqr.dll
```

Other Platforms (.NetStandard 2.0) copy from this path

```
lib/netstandard2.0/kh.org.nbc.bakong_khqr.dll
```

Please consult the NetStandard support table for further explanation.
https://docs.microsoft.com/en-us/dotnet/standard/net-standard#net-implementation-support

Usage

Request Parameter

- **Generate KHQR**

Parameter	Type	Mandatory	Length	Description
MerchantType	KHQRMerchantType	Y		Individual, Merchant
BakongAccountID	String	Y	32	
Currency	KHQRCurrency	N		Default: KHR
Amount	Double	N	13	
MerchantName	String	Y	25	
BillNumber	String	N	25	
StoreLabel	String	N	25	
TerminalLabel	String	N	25	

- **Verification KHQR**

Parameter	Type	Mandatory	Length	Description
qrCode	String	Y		

- **Decode KHQR**

Parameter	Type	Mandatory	Length	Description
qrCode	String	Y		

iOS

- **Generate KHQR**

  Swift

```
let khqr = BakongKHQR(
```

```
                                accountId: "john_smith@devb",
                                name: "John Smith",
                                currency: .Usd,
                                setAmount: 100,
                                qrType: .Individual
                            );
let additionalData = AdditionalData(
                                billNumber: "#12345",
                                storeLabel: "Coffee Khlaing",
                                terminalLabel: "Cashier1")
khqr?.setAdditionData(additionalData)
let response = khqr?.generate()
let khqrData = response?.data as? KHQRData
print("data: \(khqrData?.qr)")
print("md5: \(khqrData?.md5)")
```

Objective C

```
BakongKHQR * khqr = [[BakongKHQR alloc]initWithAccountId:
                                @"john_smith@devb"
                                name: @"John Smith"
                                currency: Usd
                                setAmount: 100
                                qrType: Individual
                            ];
AdditionalData * additionalData = [[AdditionalData alloc] initWithBillNumber: @"#12345"
storeLabel: @"Coffee Khlaing" terminalLabel: @"Cashier1"];
[khqr setAdditionData: additionalData];
KHQRResponse* khqrResponse = [khqr generate];
KHQRData * khqrData = (KHQRData *) khqrResponse.data;
NSLog(@"data: %@", khqrData.qr);
NSLog(@"md5: %@", khqrData.md5);
```

Result

```
data:
00020101021229190015john_smith@devb52045999530384054051000.05802KH5910John
Smith6010Phnom Penh62400106#123450314Coffee
Khlaing0708Cashier1991700131612840868892630 47C1E

md5: 299e139a91ff8c353f9b47ff3fb2731a
```

- **Verification KHQR**

Swift

```
let qrCode =
"00020101021229190015john_smith@devb52045999530384054051000.05802KH5

910John Smith6010Phnom Penh6304BF30"

let response = khqr?.verify(qrCode)

let crcValidation = response?.data as? CRCValidation

print("valid : \(crcValidation?.valid)")
```

Objective C

```
NSString *qrCode =
@"00020101021229190015john_smith@devb52045999530384054051000.05802KH5910John
Smith6010Phnom Penh6304BF30";

KHQRResponse * response = [khqr verify: qrCode];

CRCValidation* crcValidation = (CRCValidation *) response.data;

NSLog(@"valid: %d", crcValidation.valid);
```

Result

```
valid: 1
```

- **Decode KHQR**

Swift

```
let decodeResponse =
BakongKHQR.decodeQr("00020101021229190015john_smith@devb5204599953038405405100.05802KH5910John Smith6010Phnom Penh6304BF30")
let decodeData = decodeResponse?.data as? KHQRDecodeData
decodeData?.printAll()
```

Objective C

```
KHQRResponse* decodeResponse = [BakongKHQR decodeQr: @"00020101021229190015john_smith@devb5204599953038405405100.05802KH5910John Smith6010Phnom Penh6304BF30"];
KHQRDecodeData* decodeData = (KHQRDecodeData *) decodeResponse.data;
[decodeData printAll];
```

Result:

```
payloadFormatIndicator: 01
pointOfInitiationMethod:  12
merchantType: 0
bakongAccountID: john_smith@devb
merchantCategoryCode: 5999
countryCode: KH
merchantName: John Smith
merchantCity:  Phnom Penh
transactionCurrency: 840
transactionAmount:  100.0
billNumber: null
storeLabel: null
terminalLabel: null
```

timestamp: null

CRC: BF30

Java for android

- **Generate KHQR**

```
BakongKHQR bakongKHQR = new BakongKHQR();
bakongKHQR.setKHQRMerchantType(KHQRMerchantType.INDIVIDUAL);
bakongKHQR.setBakongAccountID("john_smith@devb");
bakongKHQR.setCurrency(KHQRCurrency.USD);
bakongKHQR.setAmount(100.0);
bakongKHQR.setMerchantName("John Smith");
bakongKHQR.setBillNumber("#12345");
bakongKHQR.setStoreLabel("Coffee Khlaing");
bakongKHQR.setTerminalLabel("Cashier1");

KHQRResponse<KHQRData> khqrResponse = bakongKHQR.generate();
String qrCode = khqrResponse.getData().getQr();
String md5 = khqrResponse.getData().getMd5();
 System.out.println("data:" + qrCode);
 System.out.println("md5:" + md5);
```

Result

```
data:00020101021229190015john_smith@devb52045999530384054051000.05802KH5910John Smith6010Phnom Penh62400106#123450314Coffee Khlaing0708Cashier1991700131612837496600630482 3A

md5:bfe5b662d481a128656bc9ebc5d2b6ba
```

- **Verification KHQR**

```
String qrCode =
00020101021229190015john_smith@devb52045999530384054051000.05802KH5910John
Smith6010Phnom Penh99176813161153841240063O4F13C;

BakongKHQR bakongKHQR = new BakongKHQR();

boolean valid = bakongKHQR.verify(qrCode);

System.out.print(“valid:” + valid);
```

Result

```
valid:true
```

- **Decode KHQR**

```
String qr =
"00020101021229190015john_smith@devb52045999530384054051000.05802KH5910John
Smith6010Phnom Penh6304BF30";

KHQRResponse<KHQRDecodeData> khqrDecodeResponse = BakongKHQR.decode(qr);

System.out.println("data:"+khqrDecodeResponse.getData());
```

Result:

```
data:{

payloadFormatIndicator=01,

pointOfInitiationMethod=12,

merchantType=INDIVIDUAL,

bakongAccountID=john_smith@devb,

merchantCategoryCode=5999,

countryCode=KH,

merchantName=John Smith,

merchantCity=Phnom Penh,

transactionCurrency=USD,
```

```
transactionAmount=100.0,
billNumber=null,
storeLabel=null,
terminalLabel=null,
timestamp=null,
crc=BF30
}
```

C#

- **Generate KHQR**

```
var khqr = new BakongKHQR()
                .SetMerchantType(KHQRMerchantTypes.Individual)
                .SetBakongAccountID("john_smith@devb")
                .SetCurrency(KHQRCurrency.USD)
                .SetAmount(100)
                .SetMerchantName("John Smith")
                .SetBillNumber("#12345")
                .SetStoreLabel("Coffee Khlaing")
                .SetTerminalLabel("Cashier1");
var response = khqr.Generate();
var qrString = response.Data.QR;
var md5 = response.Data.MD5;
Console.WriteLine("data:" + qrString);
Console.WriteLine("md5:" + md5);
```

Result

```
data:00020101021229190015john_smith@devb5204599953038405405100.05802KH5910John Smith6010Phnom Penh62400106#123450314Coffee Khlaing0708Cashier1991700131612839711534630 4110C
md5:8f8ada3f0308fdf079bb0750332dc88
```

- **Verification KHQR**

```
var khqr = 
“data:00020101021229190015john_smith@devb52045999530384054051000.05802KH5910John Smith6010Phnom Penh6304BF30”;

var reponse = BakongKHQR.Verify(khqr);

var valid = reponse.Data.Valid;

Console.WriteLine("valid:" + valid);
```

Result:

```
valid:True
```

- **Decode KHQR**

```
var decodeData = BakongKHQR.Decode("00020101021229190015john_smith@devb52045999530384054051005802KH5910John Smith6010Phnom Penh6304BF30");
var options = new JsonSerializerOptions()
{
    WriteIndented = true
};
Console.WriteLine(JsonSerializer.Serialize(decodeData, options));
```

Result:


```
{
 "Status": {
  "Code": 0,
  "ErrorCode": null,
  "Message": null
 },
 "Data": {
  "PayloadFormatIndicator": "01",
  "PointOfInitiationMethod": "12",
  "MerchantType": 0,
  "BakongAccountID": "john_smith@devb",
  "MerchantCategoryCode": "5999",
  "CountryCode": "KH",
  "MerchantName": "John Smith",
  "MerchantCity": "Phnom Penh",
  "TransactionCurrency": 840,
  "TransactionAmount": "100.0",
  "BillNumber": null,
  "StoreLabel": null,
  "TerminalLabel": null,
```

```
    "Timestamp": null,
    "CRC": "BF30"
  }
}
```

Standard Response

Response Format

```
{   "status": {
      "code": 0,
      "errorCode": null,
      "message": 'This is message'
   },
   "data": {
      //response based on each function
         qr: "00020101021230190015john_smith@devb52045999530384054051000.05802KH5910John
         Smith6010Phnom Penh99176813161153707061063044CF4",
         md5: "92f96188ce415bfcc8f60e3bf18196ec"
   }
}
```

Response Key and Code with definition

Key	Type	Value	Message
code	int		
		0	Success
		1	Failed
errorCode	int		
		1	Bakong Account ID cannot be null or empty
		2	Merchant name cannot be null or empty
		3	Bakong Account ID is invalid
		4	Amount is invalid
		5	Merchant type cannot be null or empty
		6	Bakong Account ID Length is invalid

		7	Merchant Name Length is invalid
		8	KHQR provided is invalid
		9	Currency type cannot be null or empty
		10	Bill Number Length is invalid
		11	Store Label Length is invalid
		12	Terminal Label Length is invalid
message	String		The message describes the result.
data	Any Object		The response will be different based on function
qr	String		emv qr code provided by KHQR library.
md5	String		hash value of bakong KHQR to verify transaction status.

FAQ

1. **If my bank wants to know more or inquire about KHQR, who should I talk to?**

   KHQR team of Association of Bank in Cambodia

2. **If my bank wants to know more or inquire about How To Implement KHQR By Using Bakong As a Payment Switch, who should I talk to?**

   Bakong team of National Bank of Cambodia