EIS11 PBBA Integrated Android Merchant Button

Implementation Guide

September 2023

Document Version 3.12.6

GitHub Merchant Library Version 3.1.2

Release R3.1

Document History

Version		Date	Summary of Changes
Was	Now		
1.0		22-11-2016	Draft
1.2	1.1	25-11-2016	Updated sequence diagrams, peer reviews
1.3	2.0	30-11-2016	Final draft following Participant and peer review
1.5	2.1	16-02-2017	Add new PBBA configuration key
1.6	2.2	01-03-2017	Updated Custom to Integrated
2.0	3.0	03-05-2017	• Document format revised for R3 • Corrected version of library published to GitHub
3.1		01-07-2017	Changes to this version: • Document History revised to ensure versions align to Releases • Section 3.3. Integrating the Pay by Bank app Integrated Android Merchant Button Library into a Merchant application - updated • Section 3.4 Updating the Pay by Bank app Branded Android Merchant Button Library in a Merchant application – Added • Appendix A.1 Pay by Bank app pop-up configuration – Note updated • Appendix A.2 Hybrid Merchant Apps – Added
3.2		23-10-2017	Note added clarifying the polling mechanism on: • Section 2.2 – M-COMM journey • Section 2.3 – E-COMM journey
3.3		01-11-2017	Released to support Pay by Bank app R3 live service.
3.4 - 3.11		01-10-2018	Document version omitted to align with Pay by Bank app Release 3.1
3.12		08-10-2018	Released to support Pay by Bank app R3.1 live service.
3.12.1		17-12-2018	Changes to this version: • Section 3.3.2 – Integrate the library source – reference to Appendix – removed. • Section 3.3.3 – Branded PBBA Android Merchant Button Library Source – pbbaCustomConfig.properties – removed. • A.1 – Appendices- Pay by Bank app Button and pop-up configuration – removed.
3.12.2		12-02-2019	Introducing version 3.0.0 of the PBBA integrated android merchant button library with updated user interface.

Version		Date	Summary of Changes
Was	Now		
3.12.2		27-05-2019	Changes to this version in relation GitHub Merchant Library Version 3.0.0: (as Published in June 2019) • Section 1.1 - Introduction - Note on Level AA standard - added • Section 3.1 - Certified devices and OS versions - table updated • Section 3.1.1 - Phones - table removed • Section 3.1.2 - Tablet - table removed • Section 3.1.3 - Third party components used - becomes section 3.1.1 • Section 3.2 – Source code- link to GitHub - amended • Section 3.3.1 - Integrate the library from jCenter plugin repository - Android Studio - Procedure - 'com.zapp.library:merchant:1.0.0 becomes 'com.zapp.library:merchant:3.0.0 • Section 3.3.2 - Integrate the library source - Android Studio – Procedure : ○ `zapp-merchant-library-1.1.0' **becomes** `zapp-merchant-library-3.1.0' ○ compile project(':zapp-merchant-library-1.1.0') **becomes** compile project(':zapp-merchant-library-3.0.0') • Section 3.5.1.1 - Section title Error pop-ups **becomes** M-COMM pop-ups ○ description and screenshot revised for clarity • Section 3.5.1.2 - E-COMM pop-ups - description and screenshot revised for clarity • Section 3.5.1.3.1 - Network error pop-ups - added • Section 3.5.1.3.2 - Request expired pop-ups - added • Section 3.5.1.3.3 - Pay by Bank app code generation error pop-ups - added • Section 3.5.1.4 - More about Pay by Bank app pop-ups - added • Section 3.5.2 - Integrated Custom Android Merchant Button Layouts - added • Section 3.5.2.1 - Integrated PBBA Custom Android Merchant Button with the PBBA logo - added • Section 3.5.2.2 - Integrated PBBA Custom Android Merchant Button with More about Pay by Bank app link - added

Version		Date	Summary of Changes
Was	Now		
3.12.2		27-05-2019	• Section 3.5.2.3 - Integrated PBBA Custom Android Merchant Button with bank logos - added • Section 3.5.2.4 - Integrated PBBA Custom Android Merchant Button with bank Logos and More about Pay by Bank app link - added • Section 4.2.3.1 - Integrate pop-up for Pay by Bank app M-COMM journey - revised for clarity o Parameter name - timeout - added o How to implement the Pop-up callback - revised for clarity • Section 4.2.3.3 - Integrate More about Pay by Bank app pop-up - added • Section 4.2.6 - Implementing Integrated Custom Android Merchant Button layouts - added • Section 5 - Sample code - screenshot revised for clarity • Figure 2 - App Picker – sample screen - updated
3.12.3		20-08-2019	Changes to this version: • Section 1 - Introduction - Important and Note - revised for clarity • Section 3.1 - Certified devices and OS versions - iPad Mini 4 - removed • Section 3.2 - Source code: o Revised for clarity o Note - removed • Section 3.3 - Integrating the Pay by Bank app Integrated Android Merchant Button Library into a Merchant application - `zzapp.library:merchant:3.0.0 - *changed to* 'zapp.library:merchant:x.x.x' • Section 3.3.1 - Integrate the library from jCenter plugin repository - Note - removed • Section 3.3.2 - Integrate the library source - Note - added • Section 3.4.1 - Removing an old version that was integrated from jCenter plugin repository - Note - added • Section 3.4.2 - Removing an old version that was integrated as a library source - Note - added • Section 3.5.1.2 - E-COMM pop-ups - Note - added

Version		Date	Summary of Changes
Was	Now		
3.12.3		20-08-2019	• Section 3.5.1.3.3 - Pay by Bank app code generation error pop-ups - Note - added • Section 3.5.1.4 - More about Pay by Bank app pop-ups - Note - added • Section 3.5.2 - Integrated Custom Android Merchant Button Layouts - revised for clarity • Section 3.5.2.3 - Integrated PBBA Custom Android Merchant Button with bank logos - removed • Section 3.5.2.4 - Integrated PBBA Custom Android Merchant Button with bank Logos and More about Pay by Bank app link - removed • Section 4.2.3.3 - Integrate More about Pay by Bank app pop-up - revised for clarity • Section 4.2.6 - Implementing Integrated Custom Android Merchant Button layouts - table - revised for clarity • Table 3 - Third party components used - revised for clarity
3.12.4		09-07-2021	Minor Changes – 1. Updated integration section with new JFrog Artifactory Repository 2. Added section for 'Package visibility in Android 11 changes'
3.12.5		14-03-2023	Minor Changes: • External GitHub URL change to internal Mastercard GitHub URL • Font replacement Changes - Replace MarkForMC font to OpenSans font • Updated the dependencies, gradle versions and build tool version
3.12.6		13-09-2023	• Added Maven Central Repository option in integration section. • Removed JFrog Artifactory Integration option • Updated gradle version

Contents

Tables

Figures

1 About this document

1.1 Introduction

This documentation describes the Pay by Bank app (PBBA) Merchant Button Library for Android Applications. The document focusses on the Pay by Bank app Integrated Android Merchant Button Library behaviour and code and provides a functional and technical overview for M-COMM and E-COMM Customer journeys using an Integrated Merchant Button.

Important This version of EIS11 PBBA Integrated Android Merchant Button Implementation Guide accompanies Version 3.0 of the Merchant Library which is made available to Distributors on an optional basis as described in Section 21 of the Product and Service Definition document.

NOTE Merchant Button Library has been tested to Level AA of the WCAG 2.0 standards. To ensure adherence to Level AA it is important the Merchant Button Library is implemented as described within this document.

As set out in Section 21 of the *Product and Service Definition* document, the Operator is contemplating the introduction of functionality to display the logos of CFIs which make available Pay by Bank app through their banking apps alongside the Pay by Bank app paymarque. However, this is a roadmap item of functionality which is not currently available and will only be implemented by the Operator if considered desirable following consultation with Participants on how best to deploy such feature. Whilst the description of the Merchant Library Button and its coding may refer to CFI logos, this functionality has been disabled by the Operator within the Zapp system and, as such, no CFI logos will be displayed unless or until the Operator deploys this feature.

Implementation support is available on request.

1.2 Audience

This document is intended to be used by external Participants to support the implementation and subsequent use of the Pay by Bank app.

1.3 Scope

The scope of this document covers the implementation of the Pay by Bank app Integrated Android Merchant Button. See section 1.5 Associated documents for more related information outside the scope of this document.

1.4 Document conventions

The following conventions are specific to this document and are used throughout.

Convention	Description
Important	Highlights important text in the document.

Convention	Description
Notes	Provides more information about a topic.
Number Title text	Hyperlink to another section in the document.
Italics	Indicates a document name.
`Courier New`	Indicates code / command.

1.5 Associated documents

The following provides additional information on topics covered in this document.

- *Brand Guidelines*
- *PBBA Branded Android Merchant Button*
- *PBBA Branded iOS Merchant Button*
- *PBBA Integrated iOS Merchant Button*
- *PBBA Branded Web Merchant Button*
- *PBBA Integrated Web Merchant Button*
- *Zapp Glossary*

2 Functional overview

2.1 Introduction

The Pay by Bank app (PBBA) Android Merchant Button supports two different models:

1. Pay by Bank app Branded Android Merchant Button with Pay by Bank app pop-up

   The standard Pay by Bank app Android Merchant Button with an integrated pop-up. This is described in this document.

2. Pay by Bank app Integrated Android Merchant Button with Pay by Bank app pop-up

   Merchants and Distributors can integrate their payment button with the Pay by Bank app Integrated Android Merchant Button. The additional considerations are covered in the *PBBA Branded Android Merchant Button Implementation Guide* document and should be consulted alongside this document.

Contact your Distributor for any Distributor specific implementation updates or amendments.

2.2 M-COMM journey

The Merchant App and the Pay by Bank app CFI App are on the same device. A sample Consumer journey includes the following steps:

- The Consumer selects a Pay by Bank app method and taps on a user interface element which starts the payment:

  Note This document assumes that this user interface element is a Merchant Integrated Button.

  - If this is the first time PBBA has been used on the device and there is at least one PBBA enabled CFI App installed on the device then the PBBA pop-up will appear asking the Consumer to either continue their payment on the same device by pressing `Open banking app' or get the PBBA code to pay on another device
  - If this is not the first payment on the device and the Consumer has selected 'open banking app' from before, the Pay by Bank app enabled CFI App on the device is directly invoked
  - If there are multiple mobile banking apps then a choice of which one should open is determined by the Android OS
- The Consumer can approve or cancel the Transaction
- When the payment has been completed, the Merchant App displays the payment confirmation page

The following sequence diagram shows the interaction between the components of the M-COMM journey.

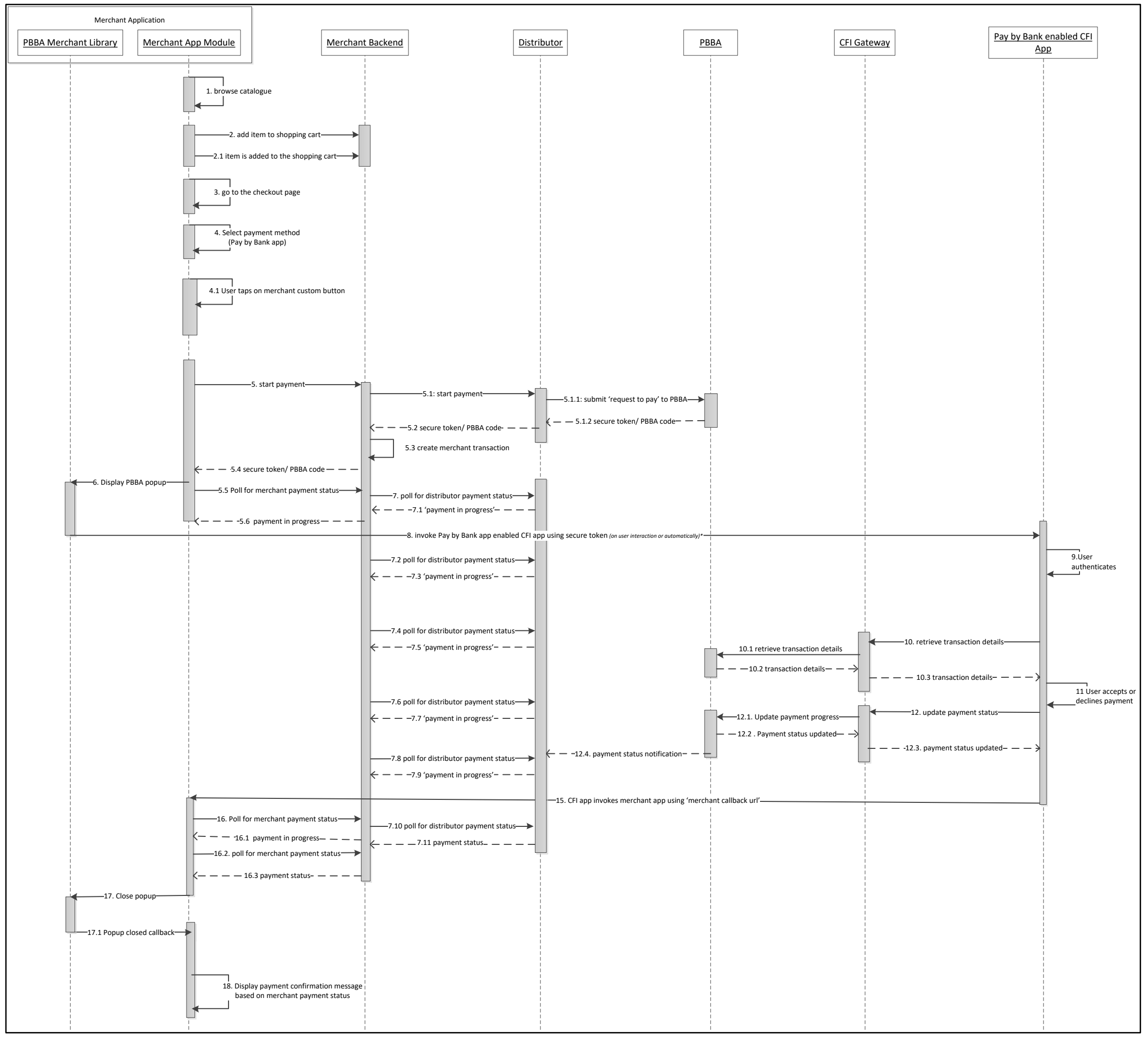

Merchant Application
PBBA Merchant Library
Merchant App Module
Merchant Backend
Distributor
PBBA
CFI Gateway
Pay by Bank enabled CFI App
1. browse catalogue
2. add item to shopping cart
2.1 item is added to the shopping cart
3. go to the checkout page
4. Select payment method (Pay by Bank app)
4.1 User taps on merchant custom button
5. start payment
5.1: start payment
5.1.1: submit 'request to pay' to PBBA
5.1.2 secure token/ PBBA code
5.2 secure token/ PBBA code
5.3 create merchant transaction
5.4 secure token/ PBBA code
6. Display PBBA popup
5.5 Poll for merchant payment status
7. poll for distributor payment status
7.1 'payment in progress'
5.6 payment in progress
8. invoke Pay by Bank app enabled CFI app using secure token (on user interaction or automatically)*
7.2 poll for distributor payment status
7.3 'payment in progress'
9.User authenticates
7.4 poll for distributor payment status
7.5 'payment in progress'
10. retrieve transaction details
10.1 retrieve transaction details
10.2 transaction details
10.3 transaction details
11 User accepts or declines payment
7.6 poll for distributor payment status
7.7 'payment in progress'
12. update payment status
12.1. Update payment progress
12.2 . Payment status updated
12.3. payment status updated
12.4. payment status notification
7.8 poll for distributor payment status
7.9 'payment in progress'
15. CFI app invokes merchant app using 'merchant callback url'
16. Poll for merchant payment status
7.10 poll for distributor payment status
16.1 payment in progress
7.11 payment status
16.2. poll for merchant payment status
16.3 payment status
17. Close popup
17.1 Popup closed callback
18. Display payment confirmation message based on merchant payment status

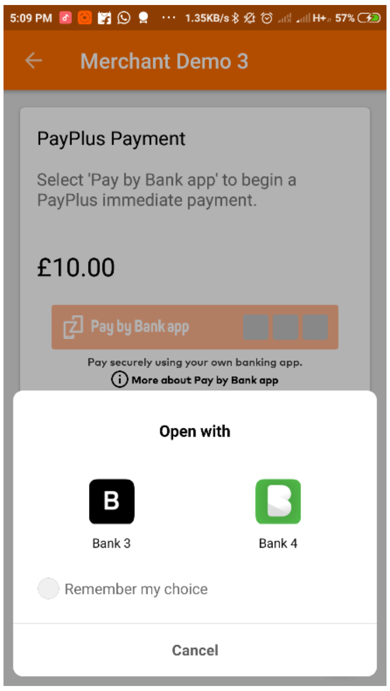
Figure 1: Interaction between the components of the M-COMM journey

Step 8 "Invoke Pay by Bank app enabled CFI App using secure token" describes the action when the Merchant Library opens the CFI App. In case of first time payment, this action happens when the Consumer taps on the 'Open Banking app' Button displayed on the M-COMM pop-up. The Merchant Library automatically opens the CFI App and no user interaction is required.

Note The polling mechanism depicted in Figure 1 is as an example implementation only. If polling is used, appropriate decoupling should be implemented between the website and backend, to support completion of a payment in case of polling failure in the website/application. Since PBBA uses a push based mechanism, it is recommended that a push based mechanism is used between the Distributor and the Merchant. The Merchant should contact their Distributor for any Distributor specific implementation updates, API definitions or amendments.

2.2.1 App Picker

Where more than one Pay by Bank app enabled CFI App is installed on the same device as the Merchant App an App Picker is displayed (see Figure 1: Interaction between the components of the M-COMM journey Figure 2 below) where the Consumer can select which CFI App they would like to use to complete the Pay by Bank app payment.



PayPlus Payment

Select 'Pay by Bank app' to begin a PayPlus immediate payment.

£10.00

Pay securely using your own banking app.

Sample Screen of Native Android App Picker with two demo CFI Apps (Bank 4 and Bank 3)

Figure 2: App Picker – sample screen

2.3 E-COMM journey

The Merchant App and the Pay by Bank app CFI App are on different devices.

A sample Consumer journey includes the following steps:

- The Consumer selects a Pay by Bank app method and taps the button (see note below) which starts the payment

  Note This document assumes the user interface element is a Merchant Integrated Button.

- The Merchant App displays a six letter PBBA code
- On another device, the Consumer opens a Pay by Bank app enabled CFI App and enters the six letter PBBA code to retrieve the Transaction
- The Consumer can approve or cancel the Transaction
- When the Consumer completes the payment journey on the CFI App, the Merchant App (on the first device) displays the payment confirmation page

The following sequence diagram shows the interaction between the components of the E-COMM journey.

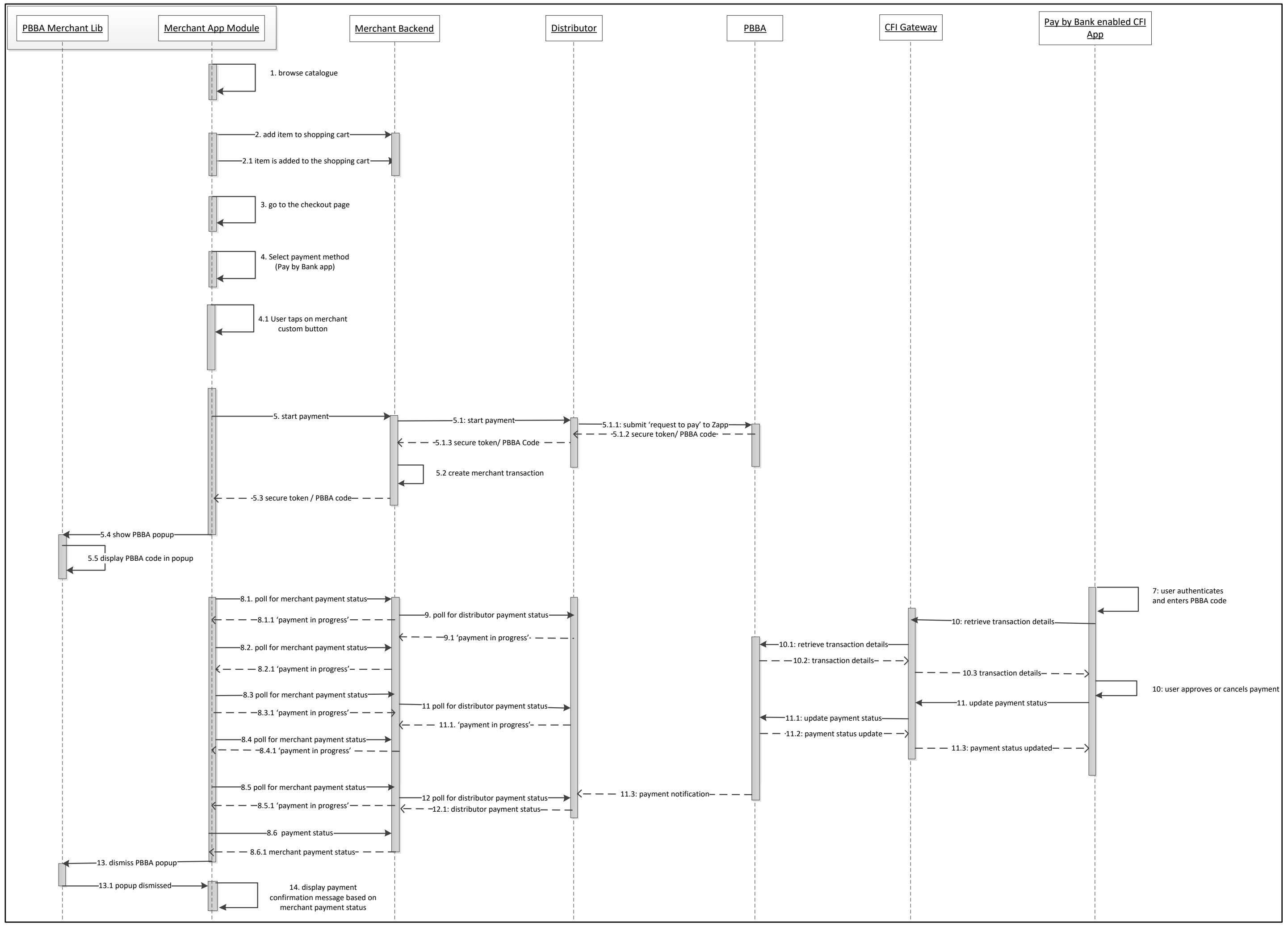

PBBA Merchant Lib
Merchant App Module
Merchant Backend
Distributor
PBBA
CFI Gateway
Pay by Bank enabled CFI App
1. browse catalogue
2. add item to shopping cart
2.1 item is added to the shopping cart
3. go to the checkout page
4. Select payment method (Pay by Bank app)
4.1 User taps on merchant custom button
5. start payment
5.1: start payment
5.1.1: submit 'request to pay' to Zapp
5.1.2 secure token/ PBBA code
5.1.3 secure token/ PBBA Code
5.2 create merchant transaction
5.3 secure token / PBBA code
5.4 show PBBA popup
5.5 display PBBA code in popup
7: user authenticates and enters PBBA code
8.1. poll for merchant payment status
8.1.1 'payment in progress'
9. poll for distributor payment status
9.1 'payment in progress'
10: retrieve transaction details
10.1: retrieve transaction details
10.2: transaction details
10.3 transaction details
10: user approves or cancels payment
8.2. poll for merchant payment status
8.2.1 'payment in progress'
8.3 poll for merchant payment status
8.3.1 'payment in progress'
11 poll for distributor payment status
11.1. 'payment in progress'
11. update payment status
11.1: update payment status
11.2: payment status update
11.3: payment status updated
8.4 poll for merchant payment status
8.4.1 'payment in progress'
8.5 poll for merchant payment status
8.5.1 'payment in progress'
11.3: payment notification
12 poll for distributor payment status
12.1: distributor payment status
8.6 payment status
8.6.1 merchant payment status
13. dismiss PBBA popup
13.1 popup dismissed
14. display payment confirmation message based on merchant payment status

Figure 3: Interaction between the components of the E-COMM PBBA Code journey

Note The polling mechanism depicted is as an example implementation only. If polling is used, appropriate decoupling should be implemented between the website and backend, to support completion of a payment in case of polling failure in the website/application. Since PBBA uses a push based mechanism, it is recommended that a push based mechanism is used between the Distributor and the Merchant. The Merchant should contact their Distributor for any Distributor specific implementation updates, API definitions or amendments.

3 Technical overview

3.1 Certified devices and OS versions

The Library supports Android 2.3 Gingerbread (API 9) and up and is certified with the Android devices provided below.

	Operating System/ Browser	OS Version	Comments
Web			
	Windows/Chrome	10	Windows resolutions 1024 X 768, 1280 X 800, 1280 X 1024, 1366 X 768 1440 x 900,1680 X 1050,1600 X 1200,1920 X 1200 1920 X 1080,2048 X 1536
	Windows/Edge	10	
	Windows/IE	10	
	Windows/Firefox	10	
	Windows/Chrome	7	
	Mac OS/Safari	Mojave	Mac - Resolutions 1024 X 768, 1280 x 960, 1280 X 1024 1600 x 1200, 1920 X 1080
	Mac OS/Chrome	Mojave	
Mobile (iOS)			
	iPhone 5S	10.3.3	
	iPhone 6	11.4	
	iPhone 6S Plus	12.1.2	
	iPhone 7	10.2.0	
	iPhone 7 plus	11.4.0	
	iPhone X	12.1.4	
	iPhone XR	12.0.1	
	iPhone XS Max	12.1.2	
Mobile (Android)			
	Samsung Galaxy S8	8.0.0	
	Samsung Galaxy S7 edge	6.0.1	
	Google Pixel	9	
	Samsung Galaxy S9	8.0.0	
iPAD and Tablet			
	Samsung Tab2	4.1.2	
	Galaxy Tab S4	5.1.1	
	iPad Pro 12.0	12.1.1	

	Operating System/ Browser	OS Version	Comments
	iPad Pro 10.5	12.1.1	
	iPad Mini 2	12.3.0	

3.1.1 Third party components used

Third Party Components used in Merchant Library	Version
Android support library	Version 31 Dependencies : implementation 'com.android.support:appcompat-v7:26.1.0' implementation 'com.android.support-annotations:28.0.0' implementation 'com.android.support-v4 :26.1.0' Build Tools Version : 29.0.3

Table 1: Third party components used

3.2 Source code

To get the source code for the Pay by Bank app Merchant Library, clone the repository (or download it as ZIP package) from GitHub. This document corresponds to the Merchant Button Library releases on GitHub: https://github.com/Mastercard/pbba-merchant-button-library-android

3.3 Integrating the Pay by Bank app Integrated Android Merchant Button Library into a Merchant application

ZappMerchantLibrary is for Android Merchant applications to integrate Pay by Bank app payments.

In the case of an existing integration with an older version of the library – go to section 3.4 Updating the Pay by Bank app Branded Android Merchant Button Library in a Merchant application.

3.3.1 Integrate the library from Maven Central Repository

Android Studio – Procedure

1. Open the App level **build.gradle** file.
2. Update app level ***build.gradle*** file configuration with maven central repository if absent in to the `allprojects' and `buildscript' sections.

```
repositories {
//other repositories
 mavenCentral()
```

```
        }
```

3. Open the **build.gradle** file for the module uses the PBBA Merchant Button Library (the App module).
4. Add the line **implementation 'com.mastercard.pbba:merchant-sdk:x.x.x'** to the dependencies section of the build.gradle file.
5. Save the **build.gradle** file and make a build so the new dependency is downloaded and resolved.

NOTE **x.x.x** represents the desired library version.

3.3.2 Integrate the library source

Download the source code for the library from GitHub then import it to the Merchant App and make the necessary configuration.

Verify that the library code is compatible with the Merchant's application. Check the library's android version compatibility, gradle version requirements, and any dependencies.

Android Studio – Procedure

1. Select **File** / **New** / **Import Module**.
2. Browse the directory of the source code of the downloaded Merchant Library in to the **Source directory** field.
3. Click on **OK**.
4. Enter `zapp-merchant-library-x.x.x' in to the **Module name** field (ensure the **Import** checkbox is **ON**).
5. Click on **Finish**.
6. Add dependency **implementation project(':zapp-merchant-library-x.x.x')** in gradle file of the Merchant App module.

NOTE **x.x.x** represents the desired library version.
NOTE Resolve any potential dependency conflicts between the library and the merchant's application dependencies. Ensure that all required dependencies are correctly managed and compatible.

3.3.3 Pay by Bank app Integrated Android Merchant Button Library structure

The figure below shows the folder structure of the Android Merchant Button Library.

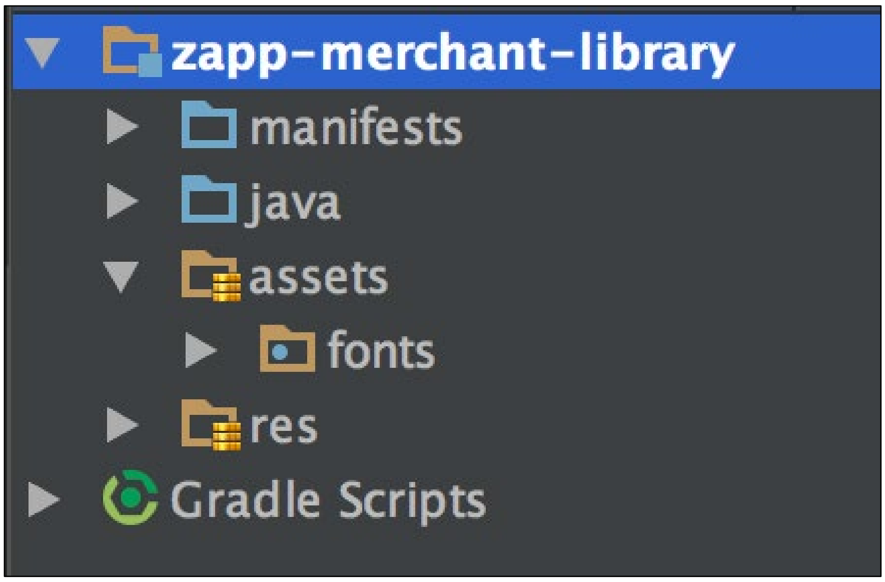


Figure 4: Integrated Pay by Bank app Android Merchant Button Library structure

The folders comprise:

Manifests	This contains an empty manifest file (required for the Android library)
Java	The Android / Java source files of the Library
Assets	Fonts – the PBBA custom font files.
Res	The resource files (images, dimensions, strings)
Gradle scripts	The Gradle build scripts

3.3.4 Package visibility in Android 11 changes at Merchant side

(https://developer.android.com/about/versions/11/privacy/package-visibility)

If Merchant's Android application uses android 11 as *targetSdkVersion* | *compileSdkVersion* which is 30 or above and integrated android merchant button library in their code, then Merchant app needs to declare below intents in **<*queries*>** tag in their main ***AndroidManifest.xml***.

```
<queries>
    <intent>
        <action android:name="android.intent.action.VIEW" />
        <data android:scheme="zapp" />
    </intent>
</queries>
```

Note: When targeting API level 30 and adding a **<*queries*>** element to the app, use the latest available release of the Android Gradle plugin.

3.4 Updating the Pay by Bank app Branded Android Merchant Button Library in a Merchant application

Update of the Merchant Library includes removing the old version of the library as a dependency and adding the new version of the library as a dependency.

3.4.1 Transitioning repository from JFrog Artifactory to Maven Central

PBBA Android merchant button library which is available to PBBA merchants via JFrog artifactory will be discontinued and the same library would be made available through Maven Central. This section provide the details on the changes which PBBA merchants need to make in order to ensure the dependency added in their applications is updated for downloading PBBA Android merchant button library from Maven Central.

Step 1 : Update app level build.gradle file configuration

```
repositories {
     //Remove the JFrog dependency from the `allprojects' and `buildscript' sections
     ~~maven {~~
          ~~url 'https://vocalinkzapp.jfrog.io/artifactory/merchant-library-maven/'~~
     ~~}~~
      //Add Maven Central dependency if absent
       mavenCentral()
}
```

Step 2 : Update build.gradle file for the module uses the PBBA Merchant Button Library (the App module).

```
dependencies {
     //Remove old dependency
      ~~implementation 'com.zapp.library:merchant:x.x.x'~~
     //add below dependency
      implementation 'com.mastercard.pbba:merchant-sdk:x.x.x'
    //x.x.x represents the desired library version.
}
```

3.4.2 Removing an old version that was integrated from Maven Central repository.

Android Studio – Procedure

1. Open the **build.gradle** file for the module uses the PBBA Merchant Button Library (the App module).
2. Go to the **dependencies** section and remove the following line:
   'implementation 'com.mastercard.pbba:merchant-sdk:y.y.y'

NOTE **y.y.y** represents the old version of the library which was integrated from Maven Central repository.

3.4.3 Removing an old version that was integrated as a library source

Android Studio – Procedure

1. Open the **build.gradle** file for the module that uses the PBBA Merchant Button Library (the App module).

2. Remove dependency **implementation project(:zapp-merchant-library-y.y.y)** in gradle file of the Merchant App module.
3. Open the settings.gradle file and remove: **zapp-merchant-library-y.y.y**.
4. Perform a sync with gradle by clicking **Tools -> Android -> Sync Project with Gradle Files**. A pop-up dialog is displayed asking if the module: **zapp-merchant-library-y.y.y** should be deleted from the IDE. Click on the **OK** button to confirm.
5. Open Project view by clicking **View -> Tool Windows -> Project** then select **Project** view from the drop-down which is defaulted to 'Android'.
6. Select module: **zapp-merchant-library-y.y.y** then right click on it and select the **Delete** option.

NOTE **y.y.y** represents the old version of the library which was integrated as a library source.

3.4.4 Adding the new version as a dependency

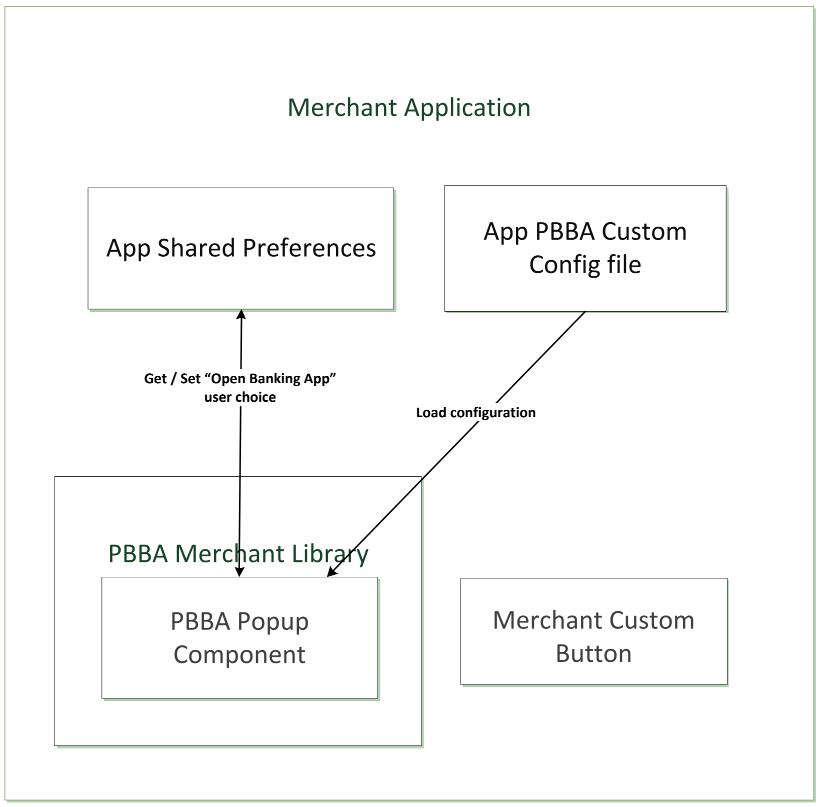
Follow the procedure in section 3.3 Integrating the Pay by Bank app Integrated Android Merchant Button Library into a Merchant application to add the new version as a dependency.

3.5 Pay by Bank app Integrated Android Merchant Button

The pop-up component is the only component provided by the Pay by Bank app Merchant App Library for Android.

The payment button is provided by the Merchant Application. The following diagram describes the Integrated Android Merchant Button implementation when the Merchant Application structure interacts with the Pay by Bank app Integrated Android Merchant Library components.

The following diagram describes the Merchant Application structure, the Pay by Bank app Merchant Library components and their interactions.



The `App Shared Preferences' shown in the diagram above is the default shared preferences of the Merchant App where the Merchant Button Library saves the flag to remember if the user has tapped `Open banking app' Button on the Pay by Bank app pop-up. This flag is saved under the key `openBankingAppButtonClicked'.

3.5.1 Pay by Bank app pop-up component

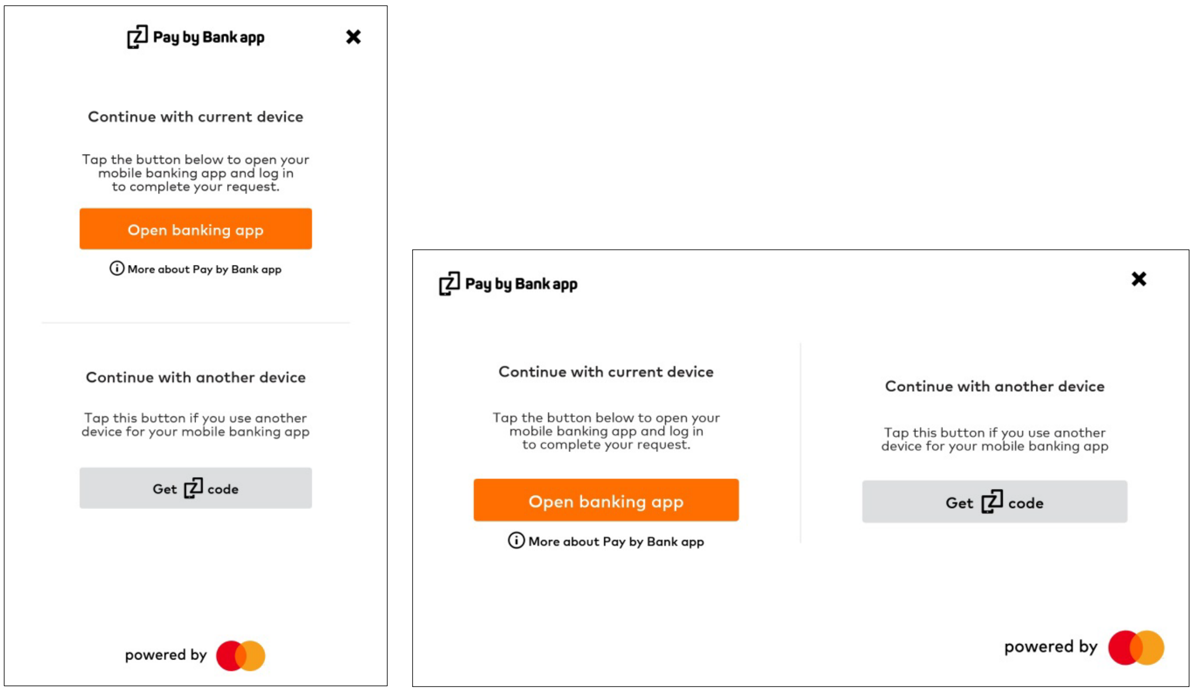
This component represents all the Pay by Bank app pop-ups that are shown to Consumers as part of the PBBA payment. The following section outlines the various pop-ups.

The pop-up examples below use the general `your mobile banking app' reference in the text.

3.5.1.1 M-COMM pop-ups

This is the pop up displayed the very first time the Pay by Bank app Button is tapped and there is at least one CFI App present on the device. Once the Consumer has pressed the 'Open banking app' Button, this preference is remembered by the Pay by Bank app Merchant App Library for Android and the Pay by Bank app enabled CFI App is opened automatically. This preference is cleared when this automatic opening cannot happen because the Consumer has uninstalled all the Pay by Bank app enabled CFI Apps. It also contains More about Pay by Bank app link.

On click of "Get Code" button, the E-COMM pop-up with a Pay by Bank app code is opened specifying the step by step instructions to help the consumer to complete their purchase on another device.



Continue with current device

Tap the button below to open your mobile banking app and log in to complete your request.

Continue with another device

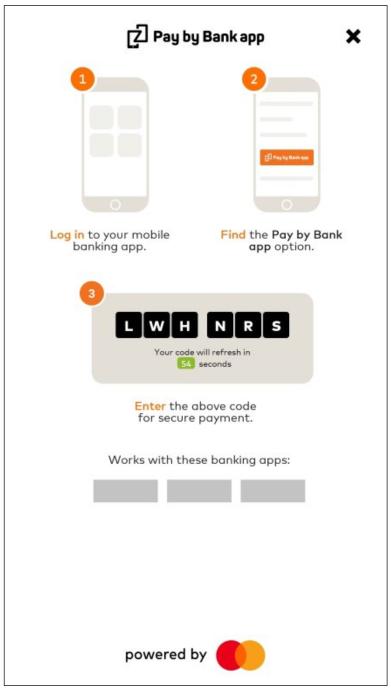
Tap this button if you use another device for your mobile banking app

M-COMM pop-ups

3.5.1.2 E-COMM pop-ups

If there are no Pay by Bank app supported CFI Apps installed on the device, the following pop-up is displayed directly on click of Pay by Bank app Integrated Android merchant button. This pop-up includes a code and step by step instructions to help the Consumer to complete their purchase on another device.

Note: As set out in Section 21 of the *Product and Service Definition* document, the Operator is contemplating the introduction of functionality to display the logos of the CFIs which make available Pay by Bank app through their banking apps (in the spaces indicated by the grey boxes in the following diagrams). However, this is a roadmap item of functionality which is not currently available and will only be implemented by the Operator if considered desirable following consultation with Participants on how best to deploy such feature. As such, the grey boxes and logos (in the following diagrams) will not appear in the current version of the Merchant Button. Furthermore, the words "Works with these banking apps" will not appear in the current version.


Pay by Bank app

1 Log in to your mobile banking app.

2 Find the Pay by Bank app option.

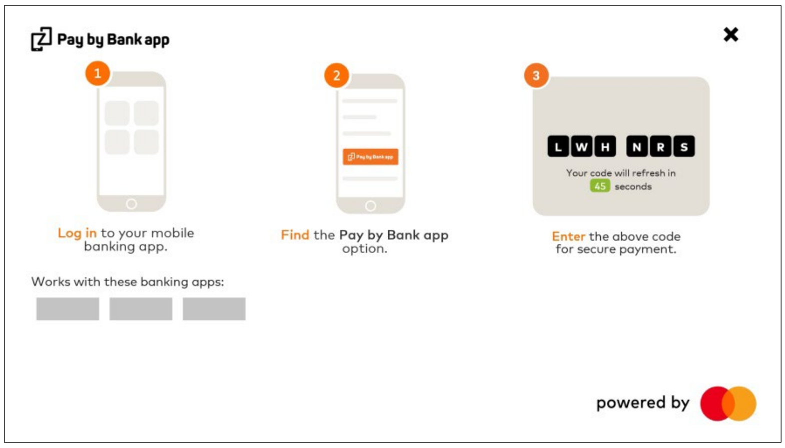
3 L W H N R S

Your code will refresh in 54 seconds

Enter the above code for secure payment.

Works with these banking apps:

powered by


Pay by Bank app

1 Log in to your mobile banking app.

2 Find the Pay by Bank app option.

3 L W H N R S

Your code will refresh in 45 seconds

Enter the above code for secure payment.

Works with these banking apps:

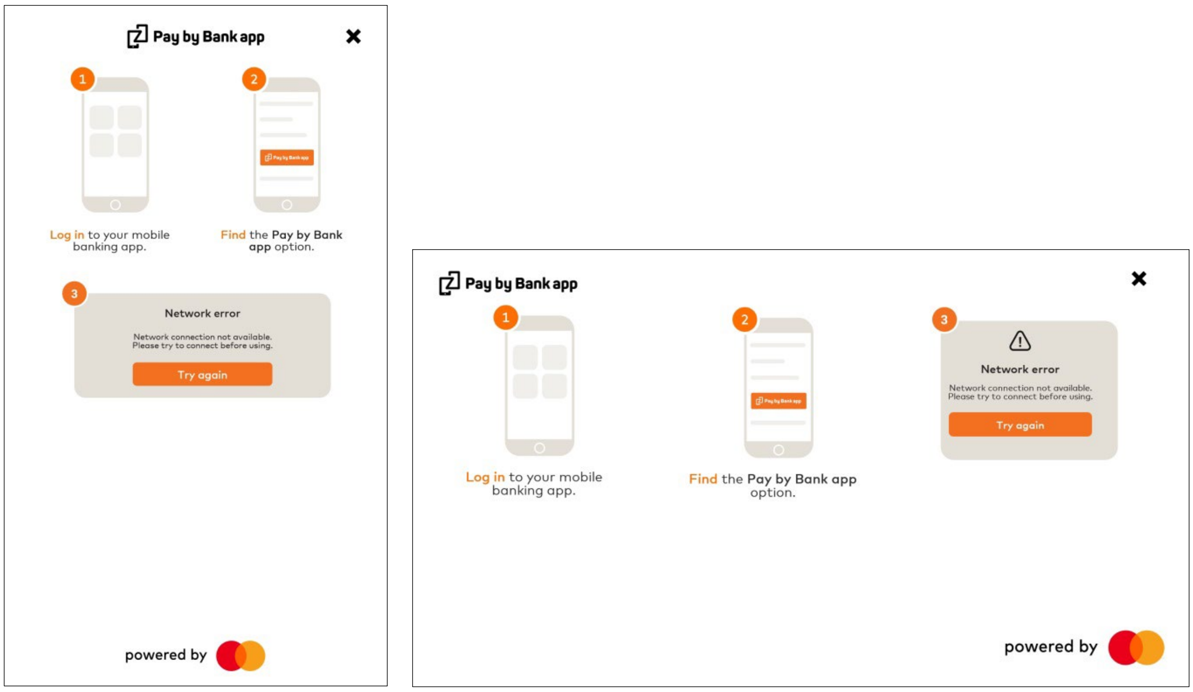
powered by

E-COMM pop-ups

3.5.1.3 Error pop-ups

If an error occurs during the payment process, the Merchant can use the Pay by Bank app pop-up to present error messages to the Consumer (this is not mandated and the Merchant can use its own method to display error messages to the user).

3.5.1.3.1 Network error pop-ups

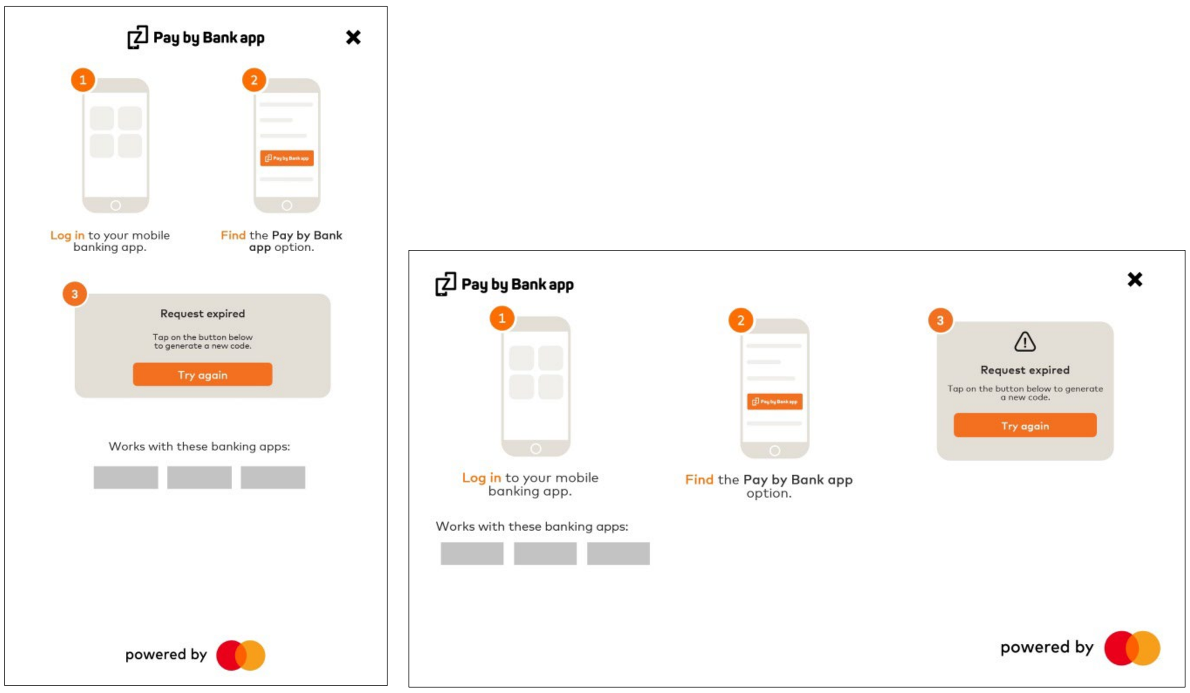

Pay by Bank app
1
Log in to your mobile banking app.
2
Find the Pay by Bank app option.
3
Try again
powered by

Network error

Network connection not available. Please try to connect before using.

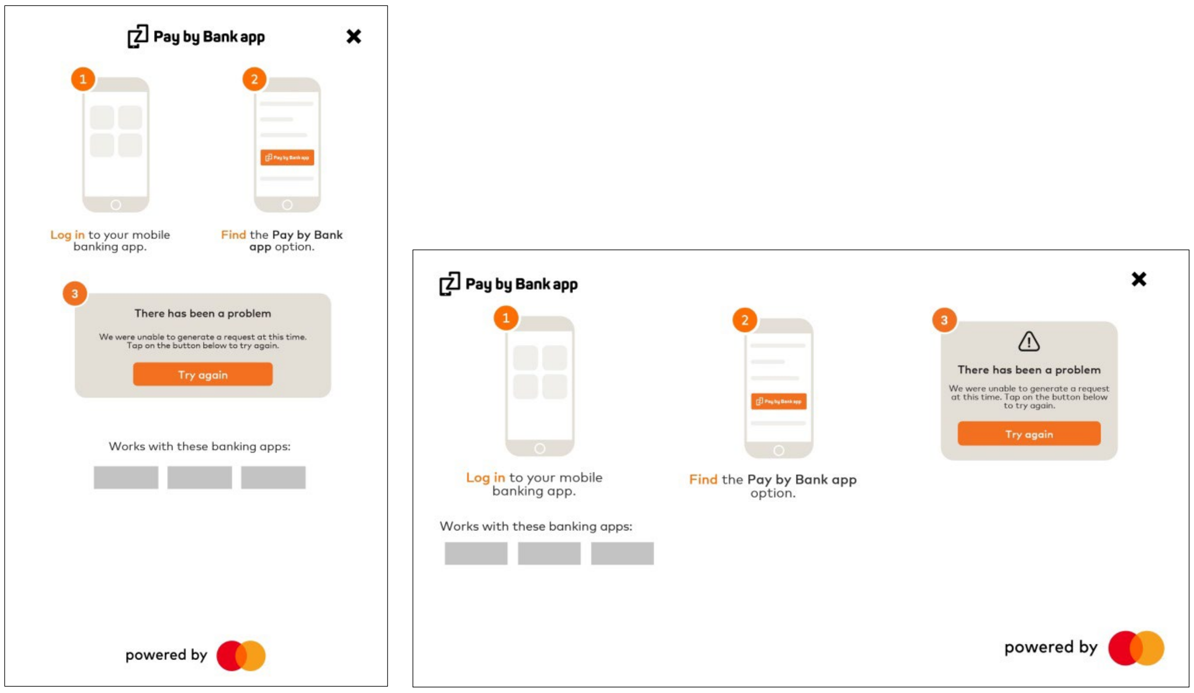
Network error pop-ups

3.5.1.3.2 Request expired pop-ups


Pay by Bank app
1
Log in to your mobile banking app.
2
Find the Pay by Bank app option.
3
Request expired
Tap on the button below to generate a new code.
Try again
Works with these banking apps:
powered by
Pay by Bank app
1
Log in to your mobile banking app.
2
Find the Pay by Bank app option.
3
Request expired
Tap on the button below to generate a new code.
Try again
Works with these banking apps:
powered by

Request expired pop-ups

3.5.1.3.3 Pay by Bank app code generation error pop-ups

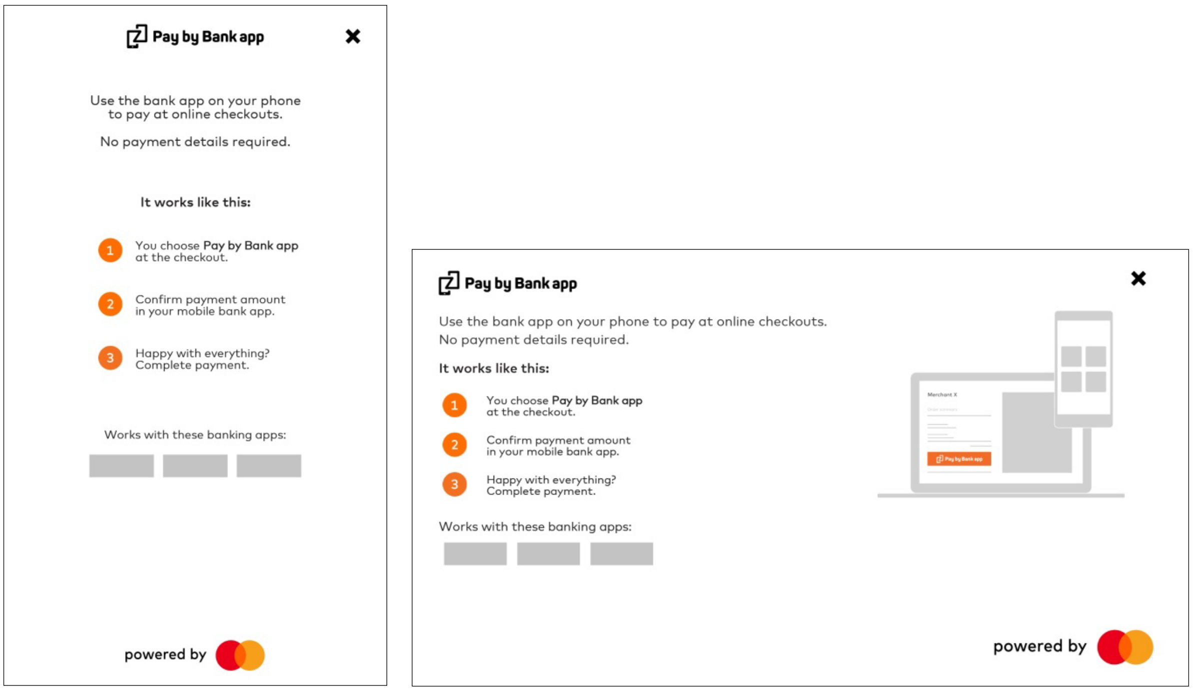

Pay by Bank app
1
Log in to your mobile banking app.
2
Find the Pay by Bank app option.
3
There has been a problem
We were unable to generate a request at this time. Tap on the button below to try again.
Try again
Works with these banking apps:
powered by
Pay by Bank app
1
Log in to your mobile banking app.
2
Find the Pay by Bank app option.
3
There has been a problem
We were unable to generate a request at this time. Tap on the button below to try again.
Try again
Works with these banking apps:
powered by

Pay by Bank app code generation pop-ups

Note: As outlined above, the words "Works with these banking apps" and the grey boxes beneath will not appear in the current version of the Merchant Button.

3.5.1.4 More about Pay by Bank app pop-ups

This popup is displayed to the consumer on click of More about Pay by Bank app link to provide more details about the pay by bank app payment journey.



Pay by Bank app

Use the bank app on your phone to pay at online checkouts.

No payment details required.

It works like this:

1 You choose Pay by Bank app at the checkout.

2 Confirm payment amount in your mobile bank app.

3 Happy with everything? Complete payment.

Works with these banking apps:

powered by

Pay by Bank app

Use the bank app on your phone to pay at online checkouts.
No payment details required.

It works like this:

1 You choose Pay by Bank app at the checkout.

2 Confirm payment amount in your mobile bank app.

3 Happy with everything? Complete payment.

Works with these banking apps:

powered by

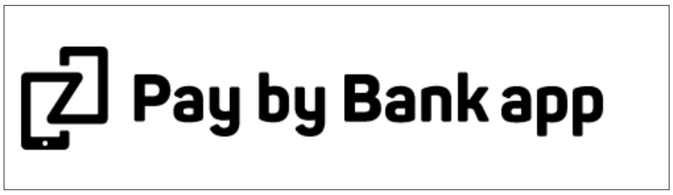
More About Pay by Bank app pop-ups

Note: As outlined above, the words "Works with these banking apps" and the grey boxes beneath will not appear in the current version of the Merchant Button.

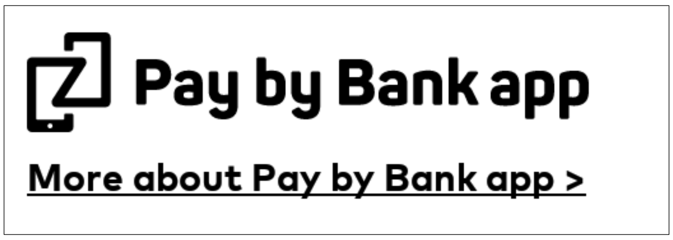
3.5.2 Integrated Custom Android Merchant Button Layouts

Integrated Custom Android Merchant Button can be displayed in four different layouts. This component is packaged with Pay by Bank app logo and More about Pay by Bank app link with informative text. More about Pay by Bank app link guides through the Pay by Bank app payment journey

3.5.2.1 Integrated PBBA Custom Android Merchant Button with the PBBA logo



3.5.2.2 Integrated PBBA Custom Android Merchant Button with More about Pay by Bank app link



4 Usage

4.1 Introduction

This chapter describes the recommended way to use the Pay by Bank app Integrated Android Merchant Button Library.

4.2 Integration Steps of an M-COMM journey

The steps to integrate the Pay by Bank app Integrated Android Merchant Button of the M-COMM journey in the Merchant App are as follows:

Procedure

1. Implement the Integrated Android Merchant Button.
2. Submit payment request to Merchant backend.
3. Integrate the Pay by Bank app pop-up (which can invoke the Pay by Bank app CFI App directly).
4. Get payment status from Merchant backend.
5. Dismiss the Pay by Bank app pop-up.
6. Display Merchant payment confirmation screen.

4.2.1 Implementing the Integrated Android Merchant Button

The Integrated Android Merchant Button should be implemented by the Merchant. The rendering and the tap event handling of the Integrated Android Merchant Button should be provided by the Merchant Application.

```
//in the Activity (or Fragment) of the payment method selection screen...
//(...)

//in the .onCreate(Bundle) method of the Activity...
final Button button = (CustomMerchantButton)
findViewById(R.id.custom_merchant_button);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(final View v) {
    //the Merchant App can start submitting the payment request
    }
});
```

4.2.2 Submit payment request to Merchant backend

The .onClick event handler of the Integrated Android Merchant Button is called when the Consumer taps the button. The Merchant has to submit a payment request to the Merchant backend. The format / protocol to use is left open to the Merchant.

4.2.3 Integrate the Pay by Bank app pop-up

4.2.3.1 Integrate pop-up for Pay by Bank app M-COMM journey

The Merchant App Library for Android provides a feature to display the Pay by Bank app branded pop-ups for the payment Transaction:

```
import com.zapp.library.merchant.util.PBBAAppUtils;
//(...)
// display PBBA Popup (provide an android.support.v4.app.FragmentActivity,
secureToken, BRN and callback parameters)
PBBAAppUtils.showPBBAPopup(/* activity */this,secureToken,brn,callback,timeout);
```

Parameter name	Parameter description	Parameter source
activity	The fragment activity in the Merchant App.	Provided by the Merchant App
secureToken	The unique token that identifies the payment request.	< Consult Distributor Documentation >
brn	The six character code that identifies the payment request for the duration of retrieval timeout.	< Consult Distributor Documentation >
callback	PBBACallback implementation that receives callback events. Please see section 'How to implement the pop-up callback' for sample code on how to implement the callback.	Provided by the Merchant App
timeout	The timeout value in milliseconds	< Consult Distributor Documentation >

Table 2: Response to Request to Pay mapping

If the Consumer has tapped the 'Open banking app' Button before and the device has a Pay by Bank app enabled CFI App installed, this API call invokes to the CFI App immediately (without displaying the pop-up). If there is no Pay by Bank app enabled CFI App installed on the Consumer's device, the Pay by Bank app pop-up displays the PBBA code automatically

How to implement the Pop-up callback

1. A callback will be received when the Pay by Bank app pop-up is dismissed.
2. A sample code of how this callback should be implemented is as follows:

```
import com.zapp.library.merchant.ui.PBBAPopupCallback;
//(...)
// display PBBA Popup (provide an android.support.v4.app.FragmentActivity,
secureToken, BRN and callback parameters)
Final PBBAPopupCallback mPopupCallback = new PBBAPopupCallback() {
    onRetryPaymentRequest() {
        //not applicable for successful responses
    }

    onDismissPopup() {
        //will be called on the PBBA popup when the Consumer taps on the top-
right corner of the popup and
        //and dismissed it
    }
    onStartTimer(){
//will be called on the PBBA popup when is starting the countdown timer
    }
```

```
     onEndTimer(){
//will be called on the PBBA popup when is finished the countdown timer
     }

}
```

4.2.3.2 Integrate Pay by Bank app pop-up for error handling

The Pay by Bank app Merchant Button Library provides a feature to display the Pay by Bank app branded error pop-up for the payment Transaction:

```
import com.zapp.library.merchant.util.PBBAAppUtils;

//(...)

// display PBBA error Popup (provide an android.support.v4.app.FragmentActivity,
errorCode (optional), errorTitle (optional), errorMessage and callback
parameters)
PBBAAppUtils.showPBBAErrorPopup(this, errorCode, errorTitle, errorMessage,
callback);
```

Parameter name	Parameter description	Parameter source
activity	The fragment activity in the Merchant App.	Provided by the Merchant App
errorCode	The error code to display. It is appended to the errorMessage and is an optional parameter.	Provided by the MerchantAapp if a custom error or it can be the error received from the Distributor gateway.
errorTitle	The error title to display. If not provided, a standard error title will be used.	Provided by the Merchant.
errorMessage	The error message to display.	Provided by the Merchant.
callback	PBBACallback implementation that receives callback events. Please see section 'How to implement the pop-up callback' for sample code on how to implement the callback.	Provided by the Merchant App

Table 3: Error mapping

The length of the strings displayed in the PBBA error dialog is not limited but the recommended lengths of these strings are:

- Maximum 25 characters for the error title
- Maximum 75 characters for the error message
- Maximum 5 characters for the error code

How to implement the pop-up callback

1. A callback will be received when the PBBA pop-up is dismissed or when the PBBA Button inside the error pop-up is tapped.
2. A sample code of how this callback should be implemented is as follows:

```
import com.zapp.library.merchant.ui.PBBAPopupCallback;

//(...)
```

```
// display PBBA Popup (provide an android.support.v4.app.FragmentActivity,
secureToken, BRN and callback parameters)
Final PBBAPopupCallback mPopupCallback = new PBBAPopupCallback() {
    onRetryPaymentRequest() {
        //will be called only when the Consumer taps on the 'Pay by Bank
app' button on the PBBA error popup
        //the Merchant can retry submitting the payment request
    }

    onDismissPopup() {
        //will be called on PBBA Error popup when the Consumer taps on the
top-right corner of the popup
        //and dismissed it
    }
}
```

4.2.3.3 Integrate More about Pay by Bank app pop-up

The Pay by Bank App Merchant Button Library provides a feature to display the More about Pay by Bank app pop-up. This pop-up contain stepwise information about how merchant button works.

```
import com.zapp.library.merchant.util.PBBAAppUtils;
//(...)
// display PBBA About Popup (provide an
android.support.v4.app.FragmentActivity)
PBBAAppUtils.showPBBAPopupAbout(this);
```

4.2.4 Get payment status from Merchant backend

The Merchant should implement the logic for getting the status of the payment. This status change detection can be implemented in various ways and is up to the Merchant. One way to implement it is to poll the Merchant backend for status change when the Merchant App has displayed the Pay by Bank app (PBBA) pop-up and the Merchant App is active (running in the foreground). There is no need to check the payment status while the Merchant App is in the background (for example, because the focus is forwarded to the CFI App). However, the status change detection can start as soon as the PBBA pop-up is displayed because if the Consumer does not have a PBBA enabled CFI App installed, the PBBA pop-up displays the PBBA code and the payment authorisation can happen in a separate device.

4.2.5 Display payment confirmation screen

The Merchant should display their payment confirmation screen once the payment status of the Transaction is known.

4.2.6 Implementing Integrated Custom Android Merchant Button layouts

The Integrated Custom Android Merchant Button should be implemented by the Merchant. The rendering and the tap event handling of the Integrated Android Merchant Button should be provided by the Merchant Application.

The size for the integrated Android Merchant Button differs based on the PBBACustomButtonType enum value.

Integrated Custom Android Merchant button layout types:

Layout	PBBACustomButtonType
Integrated PBBA Custom Android Merchant Button with the PBBA logo	NONE
Integrated PBBA Custom Android Merchant Button with More about Pay by Bank app link	MOREABOUT

Refer the size table below:

Width in dp (minimum)	Height in dp
235	WRAP_CONTENT

NOTE Merchant developer should be aware that if the size values are set to be too small, Pay by Bank app custom button layouts may not appear correctly. The sizes of all the views inside Pay by Bank app custom layout component are fixed and do not scale up or down if the layout size is increased or decreased.

The following code snippet shows an example of Custom Merchant Button layout integration.

```
<com.zapp.library.merchant.ui.view.PBBACustomButton

        android:id="@+id/pbbaCustomButton"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content" />

//in the Activity (or Fragment) of the payment method selection screen...
//(...)

//in the .onCreate(Bundle) method of the Activity...
final PBBACustomButton pbbaCustomButton = (PBBACustomButton)
view.findViewById(R.id.pbbaCustomButton);
pbbaCustomButton.setLoaderManager(getLoaderManager());
pbbaCustomButton.setLayoutType(pbbaCustomButtonType == null ?
PBBACustomButtonType.NONE : pbbaCustomButtonType);
pbbaCustomButton.setOnClickedView(new PBBACustomButton.OnClickedView() {
           @Override
           public void onClickedView() {
                //
            }
    });
```

4.3 Additional consideration for the integration of an E-COMM (two device) journey

There are no additional changes required for an E-COMM journey.

Note The Pay by Bank app Merchant Library framework automatically recognises that there is no PBBA enabled CFI App installed on the device and displays the E-COMM version of the PBBA pop-up.

4.4 Additional API set

4.4.1 API to check if the device has Pay by Bank app enabled CFI App installed

Using this API, the Merchant App can check if the Consumer's device has at least one Pay by Bank app enabled CFI App installed. The Operator recommends not to cache the response but call this API every time they want to check before any Pay by Bank app pop-up invocation.

```
import com.zapp.library.merchant.util.PBBAAppUtils;

//(...)

// check if user's device support PBBA payments (provide an Activity Context as
the single parameter)
if (PBBAAppUtils.isCFIAppAvailable(this)) {
  // the device has PBBA enabled CFI App installed
} else {
  // the device does not have PBBA enabled CFI App installed
}
```

4.4.2 API to invoke the Pay by Bank app enabled CFI App

The Merchant Library provides a feature to invoke the Pay by Bank app enabled CFI App outside the Pay by Bank app pop-up. The Operator recommends not using this function unless it is completely necessary:

```
    import com.zapp.library.merchant.util.PBBAAppUtils;

    //(...)

    // invoke the PBBA CFI App (provide an Activity Context and secureToken
    parameters)
    PBBAAppUtils.openBankingApp(/* activity */ this, secureToken);
```

This invocation happens automatically if the Merchant App calls the PBBA pop-up invocation.

Parameter name	Parameter description	Parameter source
activity	The activity in the merchant app.	Provided by the Merchant App
secureToken	The unique token that identifies the payment request.	< Consult Distributor Documentation >

Table 4: CFI App invocation mapping

4.4.3 Pay by Bank app pop-up orientation change support

The Pay by Bank app pop-ups communicate to the Merchant App through the `callback' parameter of the `showPBBAPopup' API method. In case of orientation change (if the Merchant App supports rotation) this callback implementation usually collects garbage. As the FragmentActivity which provides the activity for the Pay by Bank app pop-up gets destroyed and recreated during orientation change, the new FragmentActivity instance needs to `re-connect' to the Pay by Bank app pop-up in order to be able to receive its callbacks:

```
import com.zapp.library.merchant.util.PBBAAppUtils;

//(...)

@Override
public void onCreate(@Nullable final Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //(...)

    if (savedInstanceState != null) {
        //re-connect to the popup (provide an
android.support.v4.app.FragmentActivity and callback parameters)
        PBBAAppUtils.setPBBAPopupCallback(this, callback);
    }
}
```

Note This call is not applicable if the Merchant App does not support rotation.

5 Sample code

This section describes a frame of a simplified sample implementation of the Merchant App Library for Android for the M-COMM journey. The code displayed in black relates to the PBBA Merchant Button Library. The code displayed in blue is Merchant App related.

Note This is not compilable code as it assumes some of the Merchant specific codebase.

```
import com.zapp.library.merchant.ui.view.PBBAButton;

public class MerchantActivity extends FragmentActivity {

  private CustomMerchantButton mButton;

  private MerchantPaymentDetails mPaymentDetails;

  @Override
  public void onCreate(@Nullable final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        mButton = (CustomMerchantButton) findViewById(R.id.custom_merchant_button);
        mButton.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(final View v) {
                        submitPayment();
                }
        });

        if (savedInstanceState != null) {
                PBBAAppUtils.setPBBAPopupCallback(/* activity */ this, mPbbaPopupCallback);
        }
  }

  private void submitPayment() {
        //the Merchant App executes e.g. an async task which submits the payment request to the Merchant gateway
        //loads its response e.g. in JSON format, parses it to a Java object which is called e.g. Transaction
        //then calls the merchantAppCallback / onPaymentSubmitted callback with this Transaction object
        //or calls the merchantApp / onPaymentSubmitError callback if error happened during submitting the payment
        //to the Merchant backend
        final MerchantBackendSubmitPaymentAsyncTask submitPaymentTask
                = new MerchantBackendSubmitPaymentAsyncTask(mPaymentDetails, mMerchantAppCallback);
        submitPaymentTask.execute();
  }

  private void getPaymentStatus() {
        //here the Merchant App e.g. can wait for 5 seconds not to put heavy load on its backend
        //then make a request to the Merchant gateway, load its response e.g. in JSON format, parse it to
        //a Java object which is called e.g. PaymentStatus
        final MerchantBackendPaymentStatusAsyncTask paymentStatusTask
                = new MerchantBackendPaymentStatusAsyncTask(mMerchantAppCallback);
        paymentStatusTask.execute();
  }

  private final PBBAPopupCallback mPbbaPopupCallback = new PBBAPopupCallback {
        public void onRetryPaymentRequest() {
                //this method is called only when the Consumer clicks on the 'Pay by Bank app' button
                //on the PBBA error popup
                submitPayment();
        }

        public void onDismissPopup() {
                //this method is called on any PBBA popup when the Consumer taps the dismiss button
                //on the top-right corner of the PBBA popup
                stopPollingForPaymentStatus();
                mButton.setEnabled(true);
        }

        public void onStartTimer(){
                //this method is called on BRN PBBA popup when is starting the countdown timer
                startPollingForPaymentStatus();
        }

        public void onEndTimer(){
                //this method is called on BRN PBBA popup when is finished the countdowntimer
                stopPollingForPaymentStatus();
        }
  };

    private final MerchantAppCallback mMerchantAppCallback = new MerchantAppCallback {
        public void onPaymentSubmitted(MerchantTransactionObject Transaction) {
                //in case of standard m-Comm, this will invoke PBBA enabled CFI App automatically without
                //a PBBA Popup displayed
                //in case of standard e-Comm, this will display the PBBA Popup with the PBBA code
                PBBAAppUtils.showPBBAPopup(MerchantActivity.this,
                        Transaction.getSecureToken(),
                        Transaction.getBrn(),
                        mPbbaPopupCallback,
                        Transaction.getRetrievalExpiryInterval()*1000);
                getPaymentStatus();
        }


        public void onPaymentSubmitError(String errorCode) {
                PBBAAppUtils.showPBBAErrorPopup(MerchantActivity.this,
                        errorCode,
                        getErrorTitle(errorCode),
                        getErrorMessage(errorCode),
                        mPbbaPopupCallback);
        }
        public void onPaymentStatusReceived(PaymentStatus status) {
                if (status == IN_PROGRESS) {
                        getPaymentStatus();
                } else if (status == ERROR) {
                        PBBAAppUtils.showPBBAErrorPopup(MerchantActivity.this,
                                status.errorCode,
                                getErrorTitle(status.errorCode),
                                getErrorMessage(status.errorCode),
                                mPbbaPopupCallback);
                } else {
                        //for m-Comm payment finished, display Merchant payment status page
                        //for e-Comm payment finished, dismiss PBBA Popup
                }
        }
  };

}
```

A Appendices

A.1 Hybrid Merchant Apps

In the event, the Merchant offers an app that uses Web Views to serve the checkout page (i.e. a Hybrid app) the Merchant may choose to implement the Web Merchant Button instead of the native Android Merchant Button. Pay by Bank app does not mandate that the native library must be used for Hybrid apps.