EIS11 PBBA Integrated iOS Merchant Button

Implementation Guide

September 2019
Document Version 3.12.3
GitHub Merchant Library Version 3.1.0
Release R3.1

Copyright statement

Document History

Version		Date	Summary of Changes
Was	Now		
1.0		25-01-2016	Draft
-	2.0	20-05-2016	Released to support Pay by Bank app live service
1.2	2.1	30-11-2016	Updated following Participant and peer review
1.4	2.2	15-02-2017	Add new PBBA configuration key
1.5	2.3	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 2.2 M-COMM journey - Bulleted text below the first paragraph updated to include more information on making a payment using the Pay by Bank app ● Figure 2 - App Picker – image updated ● Section 3.1.2 Supported Languages - revised to include Objective-C, Swift 2.3 and Swift 3.0 ● Section 3.2 Source code – Note added ● Section 3.4 Pay by Bank app Integrated iOS Merchant Button – figure 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 Section 3.3.1 - Updated the CocoaPods integration information Section 4 - Update the Pay by Bank Branded iOS Merchant Button Library in a Merchant application - Added
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	Appendices A.1 - Pay by Bank app Button and pop-up configuration – removed.
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

Version		Date	Summary of Changes
Was	Now		
3.12.2		27-05-2019	• Section 3.1.1 - Certified devices and OS versions - table updated
			• Section 3.1.2 - Supported languages – revised for clarity
			• Section 3.2 - Source code - revised for clarity
			• Section 3.4.1 - Update the library using CocoaPods - revised for clarity
			• Section 3.5.1. - Pay by Bank app button component - revised for clarity
			• Section 3.5.1.1 - 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 - Error pop-ups - revised for clarity
			• Section 3.5.1.4 - Network error pop-ups - added
			• Section 3.5.1.5 - Request expired error pop-ups - added
			• Section 3.5.1.6 - Pay by Bank app code generation error pop-ups - added
			• Section 3.5.1.7 - More bout Pay by Bank app pop-ups - added
			• Section 3.5.1.8 - Integrated custom iOS merchant button layouts - added
			• Section 3.5.1.9 - Integrated PBBA Custom iOS Merchant Button with PBBA logo - added
			• Section 3.5.1.10 - Integrated PBBA Custom iOS Merchant Button with More about Pay by Bank app link - added
			• Section 3.5.1.11 - Integrated PBBA Custom iOS Merchant Button with bank logos - added
			• Section 3.5.1.12 - Integrated PBBA Custom iOS Merchant Button with bank logos and More about Pay by Bank app link - added
			• Section 4.2.1 - Implement the Integrated iOS Merchant Button - revised for clarity
			• Section 4.2.3.1 - Integrate pop-up for Pay by Bank app M-COMM journey - revised for clarity
			o How to implement the pop-up callback - revised for clarity
			• Section 4.2.6 - Implementing Integrated iOS Pay by Bank app Custom Merchant Button Layouts - added
			• Section 5 - Sample code - screenshot revised for clarity
3.12.3		20-08-2019	Changes to this version:
			• Section 1 - Introduction - Important and Note - revised for clarity

Version		Date	Summary of Changes
Was	Now		
3.12.3		20-08-2019	• Section 3.1 - Certified devices and OS versions - iPad Mini 4 - removed
			• Section 3.2 - Source code:
			○ Revised for clarity
			○ Note - removed
			• Section 3.3.1 - CocoaPods integration - revised for clarity
			• Section 3.4 - Update the Pay by Bank Branded iOS Merchant Button Library in a Merchant application - revised for clarity
			• Section 3.5.1.2 - E-COMM pop-ups - Note - added
			• Section 3.5.1.5 - Request expired error pop-ups - Note - added
			• Section 3.5.1.6 - Pay by Bank app code generation error pop-ups - Note - added
			• Section 3.5.1.7 - More about Pay by Bank app pop-ups - Note - added
			• Section 3.5.1.11 - Integrated PBBA Custom iOS Merchant Button with bank logos - removed
			• Section 3.5.1.12 - Integrated PBBA Custom iOS Merchant Button with bank logos and More about Pay by Bank app link - removed
			• Section 4.2.6 - Implementing Integrated iOS Pay by Bank app Custom Merchant Button Layouts - Integrated Custom iOS Merchant button layout type table- revised for clarity

Contents

Tables

Figures

1   About this document

1.1   Introduction

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

Important	This version of EIS11 PBBA Integrated iOS 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	The 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 iOS Merchant Button. See section 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.
Notes	Provides more information about a topic.

Convention	Description
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 iOS Merchant Button*

● *PBBA Branded Android Merchant Button*

● *PBBA Integrated Android 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) iOS Merchant Button Library supports two different models:

1. PBBA Branded iOS Merchant Button with PBBA pop-up:

    The standard PBBA iOS Merchant Button with PBBA pop-up. This is covered in the *PBBA Branded iOS Merchant Button* document and should be consulted alongside this document.

2. PBBA Integrated iOS Merchant Button with PBBA pop-up:

    Merchants and Distributors can integrate their 0payment button with the PBBA Integrated iOS Merchant Button Library. This is covered in 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 taps a Pay by Bank app Button which starts the payment. This document covers the integrated Pay by Bank app Merchant Button:

    *   If this is the first time Pay by Bank app has been used on the device and there is at least one Pay by Bank app enabled CFI App installed on the device then the Pay by Bank app pop-up will appear asking the Consumer to either continue his 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 user 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 will be dealt with by the custom iOS App Picker which will be implemented by CFI app.

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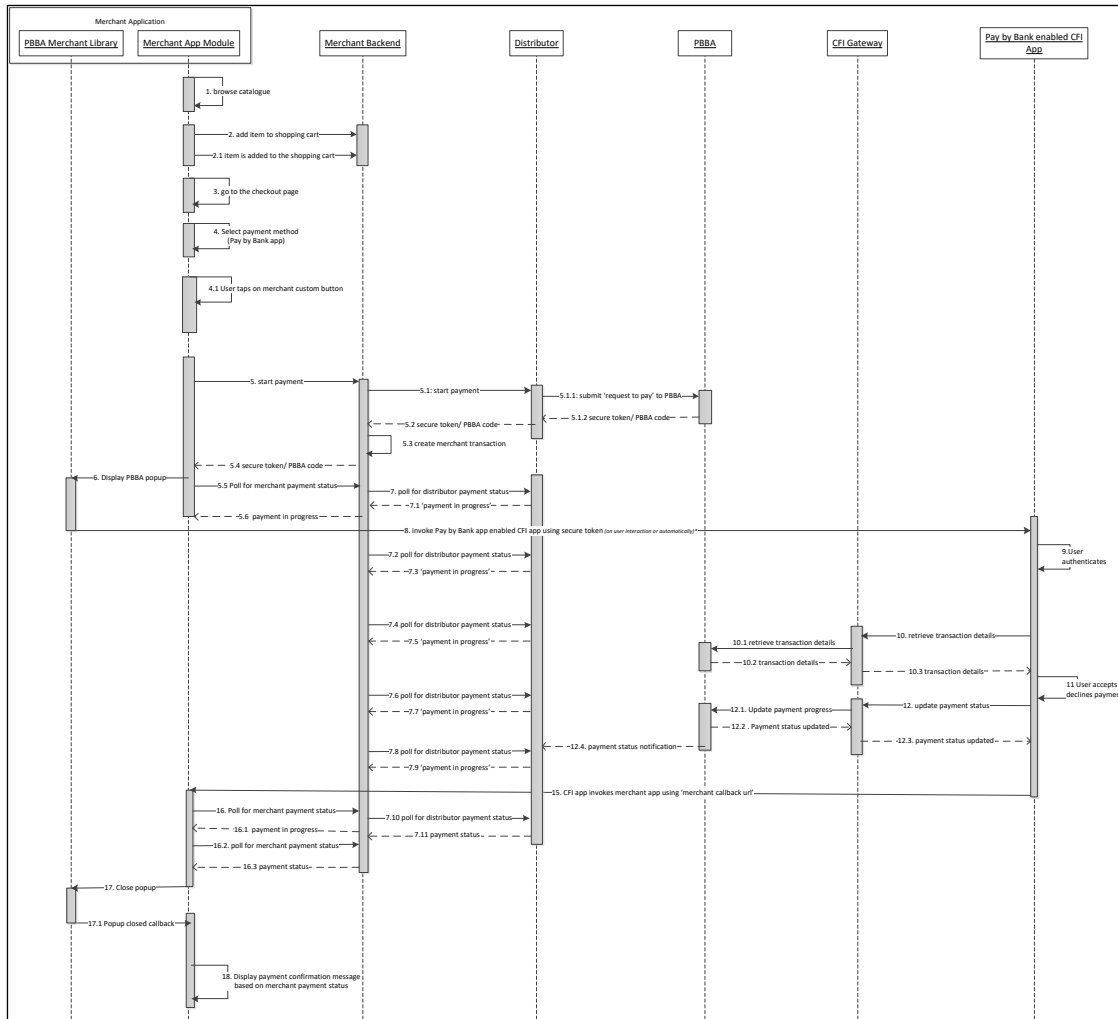
Figure 1:   Interaction between the components of the M-COMM journey

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 the 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.

Sample Screen of Native iOS App Picker with two demo CFI Apps (Bank Too and Bank 3)
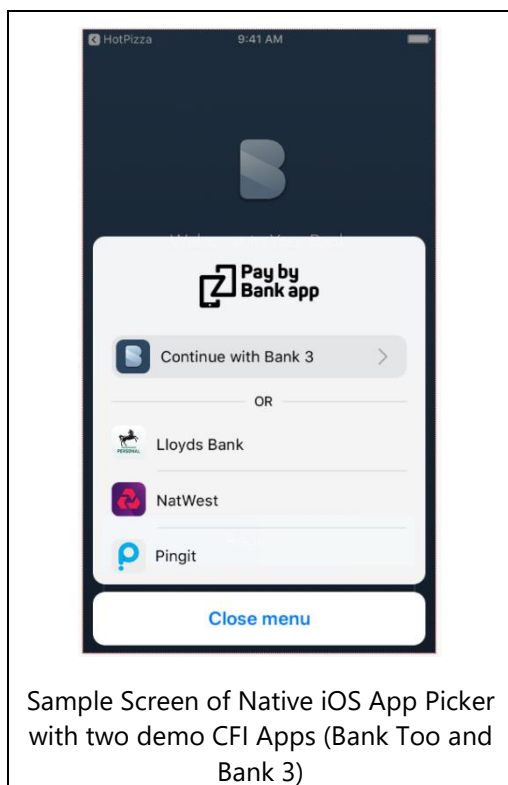
Figure 2:   App Picker – sample screen

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 on a user interface element which starts the payment

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

● The Merchant App displays a six letter Pay by Bank app code

● On another device, the Consumer opens a Pay by Bank app enabled CFI App and enters the six letter Pay by Bank app 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.
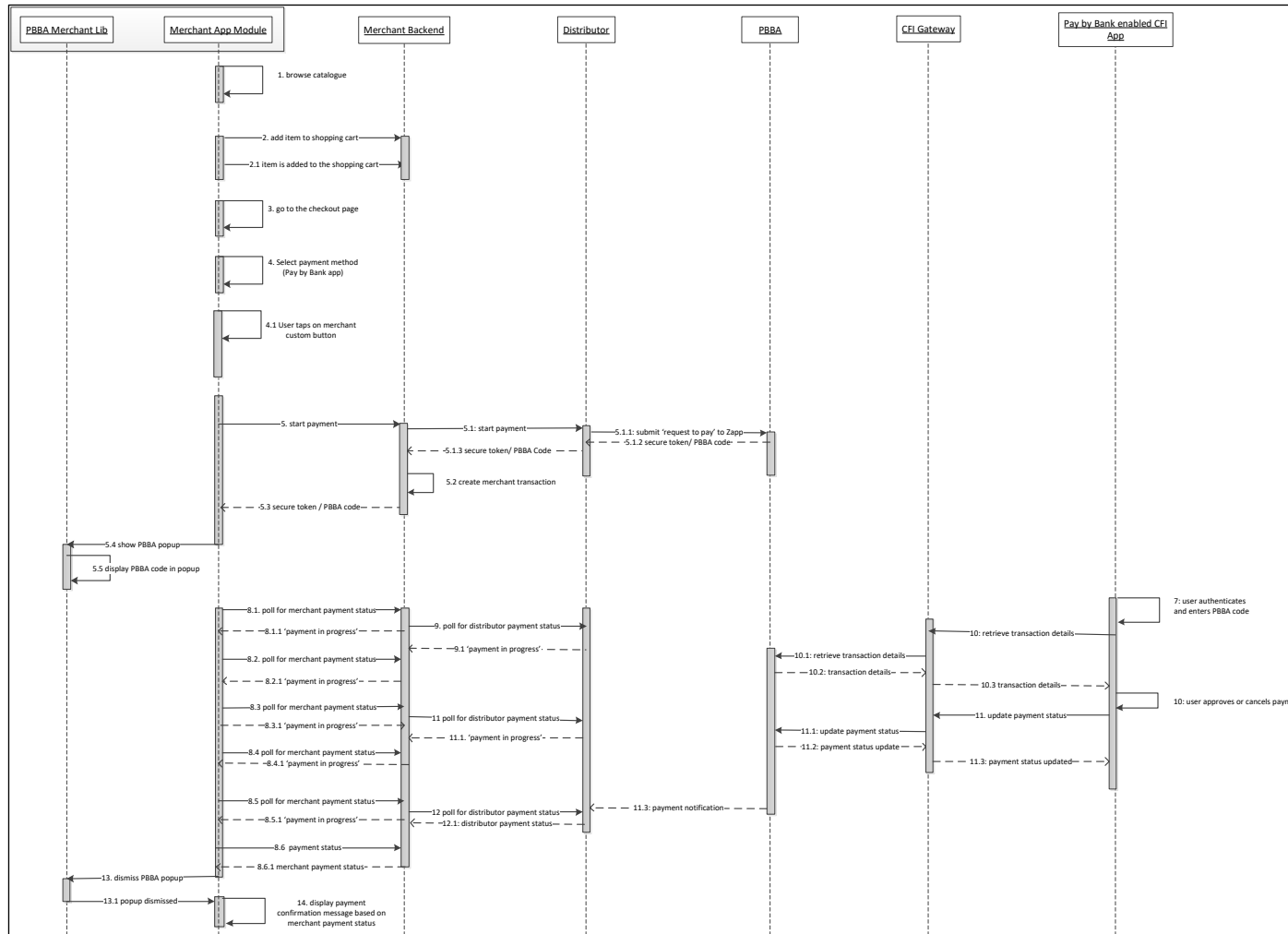
Figure 3:   Interaction between the components of the E-COMM PBBA code journey

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 iOS 8 and up and is certified with the iOS devices provided below:

3.1.1   Certified devices and OS versions

	Operating System/ Browser	OS Version	Comments
Web			
	Windows/Chrome	10	Windows resolutions
	Windows/Edge	10	1024 X 768, 1280 X 800, 1280 X 1024, 1366 X 768
	Windows/IE	10	
	Windows/Firefox	10	1440 x 900,1680 X 1050,1600 X 1200,1920 X 1200
	Windows/Chrome	7	1920 X 1080,2048 X 1536
	Mac OS/Safari	Mojave	Mac - Resolutions
	Mac OS/Chrome	Mojave	1024 X 768, 1280 x 960, 1280 X 1024 1600 x 1200, 1920 X 1080
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	

Table 1:   Certified devices and OS versions

3.1.2   Supported Languages

The PBBA Merchant Button Library is written in Objective-C and was validated with applications written in the following languages:

	Objective-C
Supported Languages	Swift 2.3
	Swift 3.0
	Swift 4.2

Table 2:   Supported Languages

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. Merchant Library is available on GitHub : https://github.com/vocalinkzapp/pbba-merchant-button-library-ios

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

The Pay by Bank app Integrated iOS Merchant Button Library is for iOS Merchant applications to integrate Pay by Bank app payments.

	CocoaPods integration	Manual integration using a static library/framework	Manual integration as subproject
Pay by Bank app Merchant Button Library with Merchant App developed in Objective-C	YES	YES	YES
Pay by Bank app Merchant Button Library with Merchant App developed in Swift 2.3	YES	YES	YES
Pay by Bank app Merchant Button Library with Merchant App developed in Swift 3.0	YES	YES	YES

Table 3:   Pay by Bank app Merchant Button Library integration types

3.3.1   CocoaPods integration

Add the following line to the Podfile if the CocoaPods dependency manager is being used.

Objective-C projects

```
pod 'ZappMerchantLib', :git => 'https://github.com/vocalinkzapp/pbba-merchant-button-library-ios.git'
```

Swift projects

```
use_frameworks!

pod 'ZappMerchantLib', :git => 'https://github.com/vocalinkzapp/pbba-merchant-button-library-ios.git'
```

3.3.2   Manual integration

3.3.2.1   Build the framework manually from the source and include the output binary into the user project

Objective-C projects

Follow these procedure steps to generate and integrate the Zapp Merchant Lib framework into the user project.

Procedure steps

1.   Open the **ZappMerchantLib.xcworkspace** file.

     Note   Ensure the workspace is opened – **not** the project file.

2.   Select **ZappMerchantSDK** scheme from scheme selection drop-down.

3.   Run the selected scheme.

4. The folder with build artefacts is exported to **~/Desktop folder**. The output folder now contains the following artefacts:

ZappMerchantLib.framework	Static framework.
ZappMerchantLibResources.bundle	Resource bundle.
clibZappMerchantLib.a	Static library (optional, to be used instead of ZappMerchantLib.framework).
Headers	Folder to be used together with libZappMerchantLib.a static lib (optional).
ZappMerchantLib-docset.zip	The documentation set.

Note   If the framework documentation is to be exported along with the build artefacts then install the appledoc tool (brew install appledoc).

5. Add the generated framework **ZappMerchantLib.framework** to the **Link Binary With Libraries** section of the **Build Phases** project.

6. Add the **-ObjC linker** flag to the **Other Linker Flags** field in the **Build Settings** project.

7. Add **ZappMerchantLibResources.bundle** to the **Copy Bundle Resources** section of the Build Phases project.

Swift projects

The integration steps for Swift projects are the same as for Objective-C plus but with one additional step:

8. Add the **<ZappMerchantLib/ZappMerchantLib.h>** import statement to the **MerchantApp-Bridging-Header.h** file.

Note   It is important to add the library import statement to the **Objective-C Bridging Header** as it is a static framework which does not define a Swift module.

3.3.2.2   Add the Pay by Bank app Integrated iOS Merchant Button Library as subproject

Objective-C projects

Alternatively, you can add the Pay by Bank app Integrated iOS Merchant Button Library as a subproject to your app project:

Procedure steps

1. Download the **.zip** version of the library from Github.

2. **Unarchive** the project and copy it to the **project folder**.

3. Open the project in Xcode then go to the **File** menu and select the **Add File to "..."** option. Browse for the **ZappMerchantLib.xcodeproj** file.

4. Go to the **Link Binary With Libraries** section of the project **Build Phases** and press the plus (**+**) button. From the list add the **libZappMerchantLib.a** static lib to the linked libraries.

5. Add the **-ObjC linker** flag to the **Other Linker Flags** field in the **Build Settings** project.

6. Add **PATH_TO_MERCHANT_LIB_ROOT_DIRECTORY** to the **Header Search Paths** field in the **Build Settings** project.

7. Go to the **Copy Bundle Resources** section of the **Build Phases** project and press the plus (**+**) button. From the list add the **ZappMerchantLibResources.bundle** to your resources. If there is no **ZappMerchantLibResources.bundle** in the list, drag on drop it from the **Products** folder in the **ZappMerchantLib subproject**.

Swift projects

The integration steps for Swift projects are the same as for Objective-C plus one additional step:

Procedure steps

1. Add the **<ZappMerchantLib/ZappMerchantLib.h>** import statement to the **MerchantApp-Bridging-Header.h** file.

    Note   It is important to add the **library import** statement to the **Objective-C Bridging Header** as it is a static library which does not define a Swift module.

3.3.3   Integrated Pay by Bank app iOS Merchant Button Library structure

The figure below shows the folder structure of the iOS Merchant Button Library.
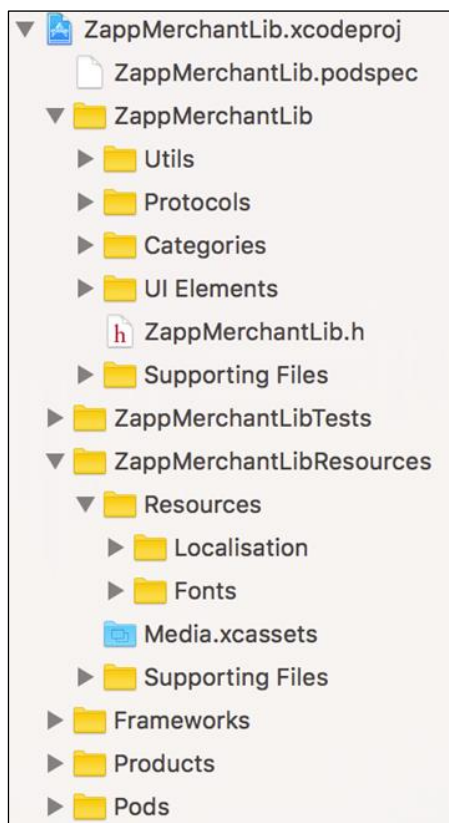
Figure 4:   Integrated Pay by Bank app iOS Merchant Button Library structure

The folders comprise:

ZappMerchantLib.podspec	The CocoaPods spec for the library.
ZappMerchantLib	Objective-C source files of the library.
ZappMerchantLibTests	The unit tests of the library.
ZappMerchantLibResources	The Library assets. These comprise: ● Fonts ● Strings files ● Images
Frameworks	The system frameworks to which the library should be linked.
Products	The project products.
Pods	The custom pods used for unit testing (e.g. Kiwi).

3.4   Update the Pay by Bank Branded iOS Merchant Button Library in a Merchant application

Update of the Zapp 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   Update the library using CocoaPods

Open the Podfile and update the version of the library defined there:

```
pod 'ZappMerchantLib', 'x.x.x', :git => 'https://github.com/vocalinkzapp/pbba-
merchant-button-library-ios.git'
```

In the sample above x.x.x represents the desired library version.

If version is not indicated then simply run 'pod update' command.

3.4.2   Removing an old version that was integrated as a precompiled binary

1. Open the project which integrates the Pay by Bank iOS Merchant Button Library;

2. Find and remove **ZappMerchantLib.framework** and **ZappMerchantLibResources.bundle** files from the project.

3.4.3   Removing an old version that was integrated as subproject

3. Open the project which integrates the Pay by Bank iOS Merchant Button Library;

4. Find and remove the **ZappMerchantLib** subproject from the main project.

5. Go to project builds settings and find the Header Search Paths field. Remove the **PATH_TO_MERCHANT_LIB_ROOT_DIRECTORY** from that field.

3.4.4   Adding the new version as a dependency

Follow the steps from section Integrate the Pay by Bank app Branded iOS Merchant Button Library into a Merchant application to add the new version as a dependency.

3.5   Pay by Bank app Integrated iOS Merchant Button

The pop-up component is the only component provided by the Pay by Bank app Integrated iOS Merchant Button Library. The payment button is provided by the Merchant Application.

The following diagram describes the Merchant Application structure, the Pay by Bank app Merchant Library components and their interactions.
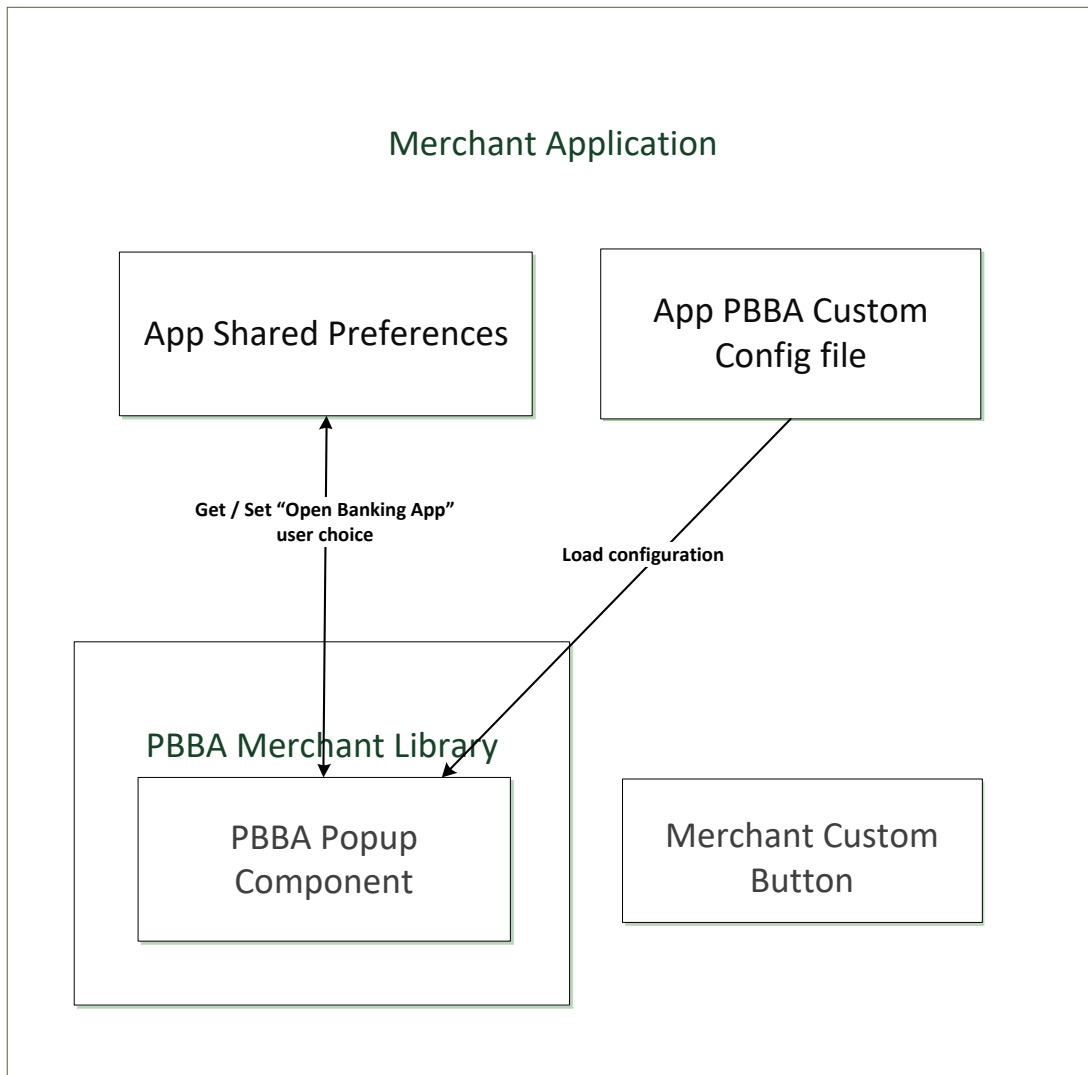
`App Shared Preferences' shown in the diagram above, is the `NSUserDefaults' storage of the Merchant App where the Merchant Button Library saves the flag to remember that the Consumer has tapped the `Open banking app' button on the Pay by Bank app pop-up. This flag is saved under the key `com.zapp.bankapp.remembered'.

3.5.1   Pay by Bank app Button component

This component represents all the Pay by Bank app pop-ups which are shown to Consumers as part of the Pay by Bank app 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 pop up is 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 Integrated iOS Merchant Button Library 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 pop-up with a code is opened specifying the step by step instructions to help the Consumer to complete their purchase on another device.
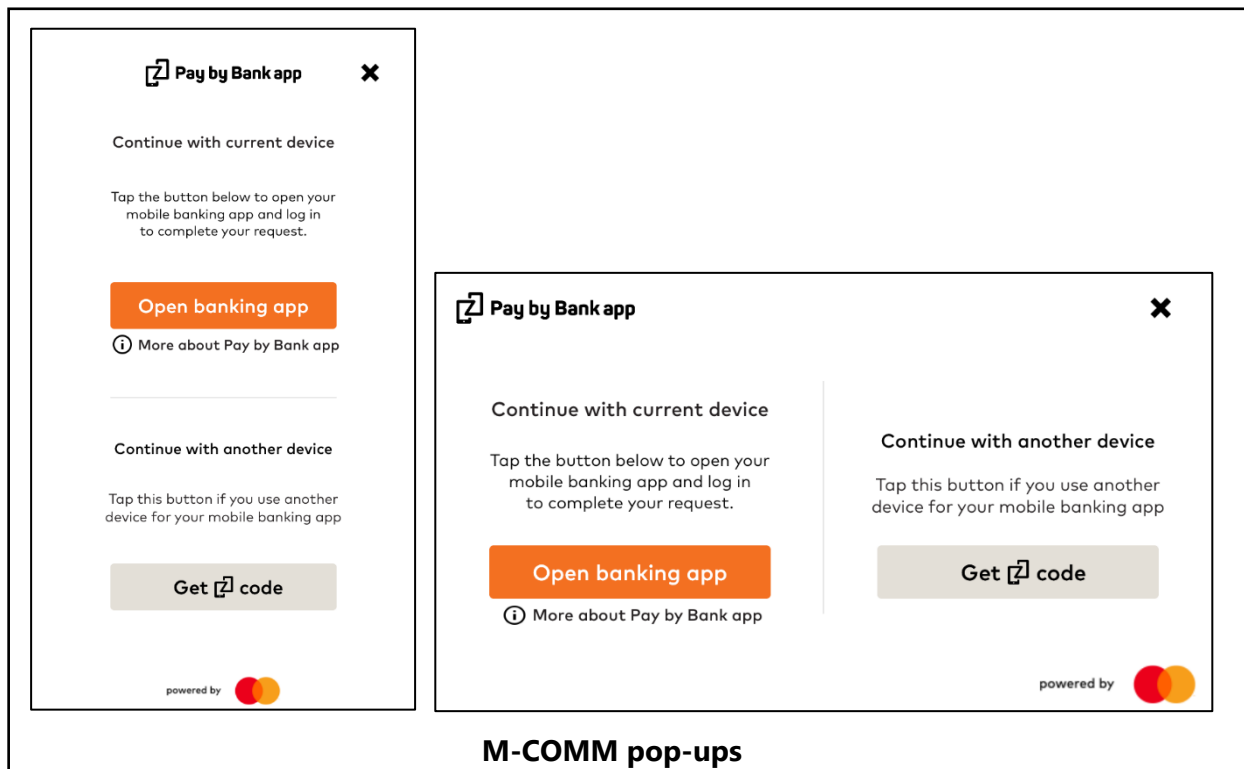
M-COMM pop-ups

3.5.1.2   E-COMM pop-ups

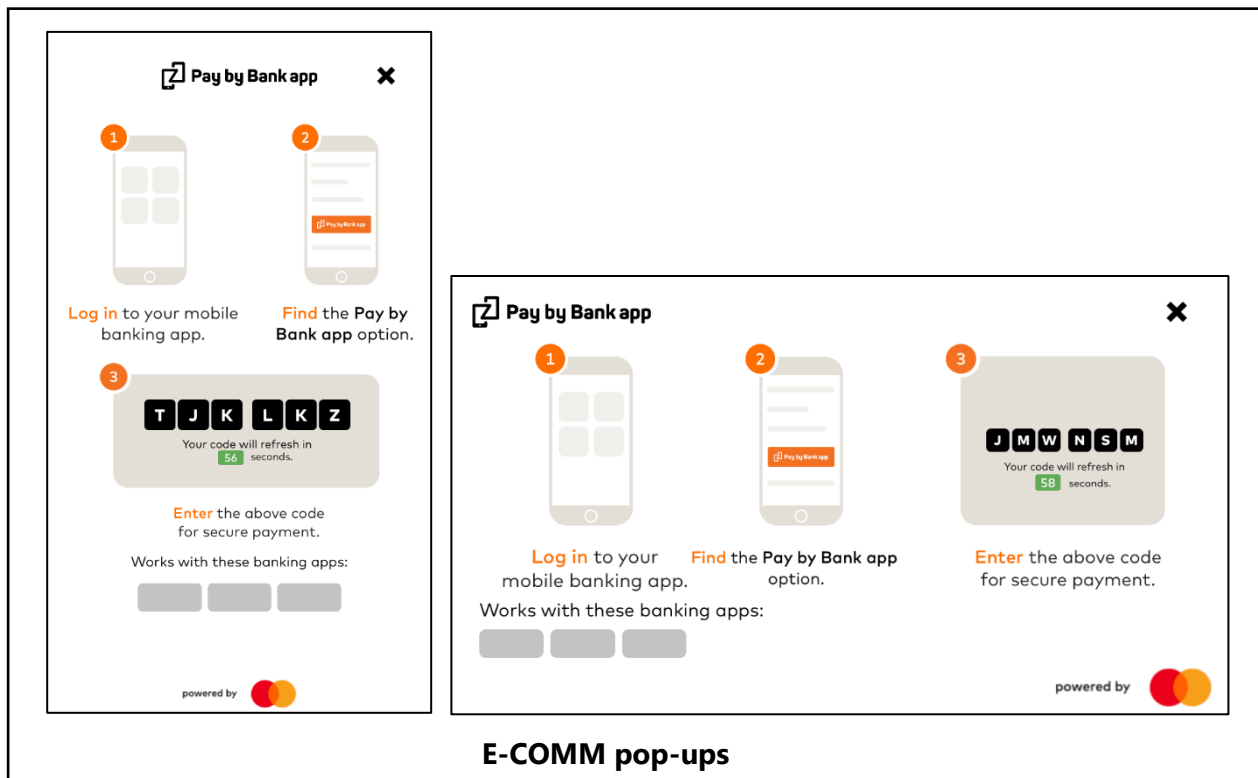If there are no Pay by Bank app supported CFI Apps installed on the device, the following pop-up is displayed directly on click of Pay by Bank app integrated iOS merchant button. This pop-up includes a code and step by step instructions to help the Consumer to complete their purchase on another device.

E-COMM pop-ups

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 CFIs which make available Pay by Bank app through their banking apps (in the spaces indicated by the grey boxes in the above 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 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.

3.5.1.3   Error pop-ups

If an error occurs during the payment process, the consumer is shown with the following error pop-ups based on the error type.

3.5.1.4   Network error pop-ups
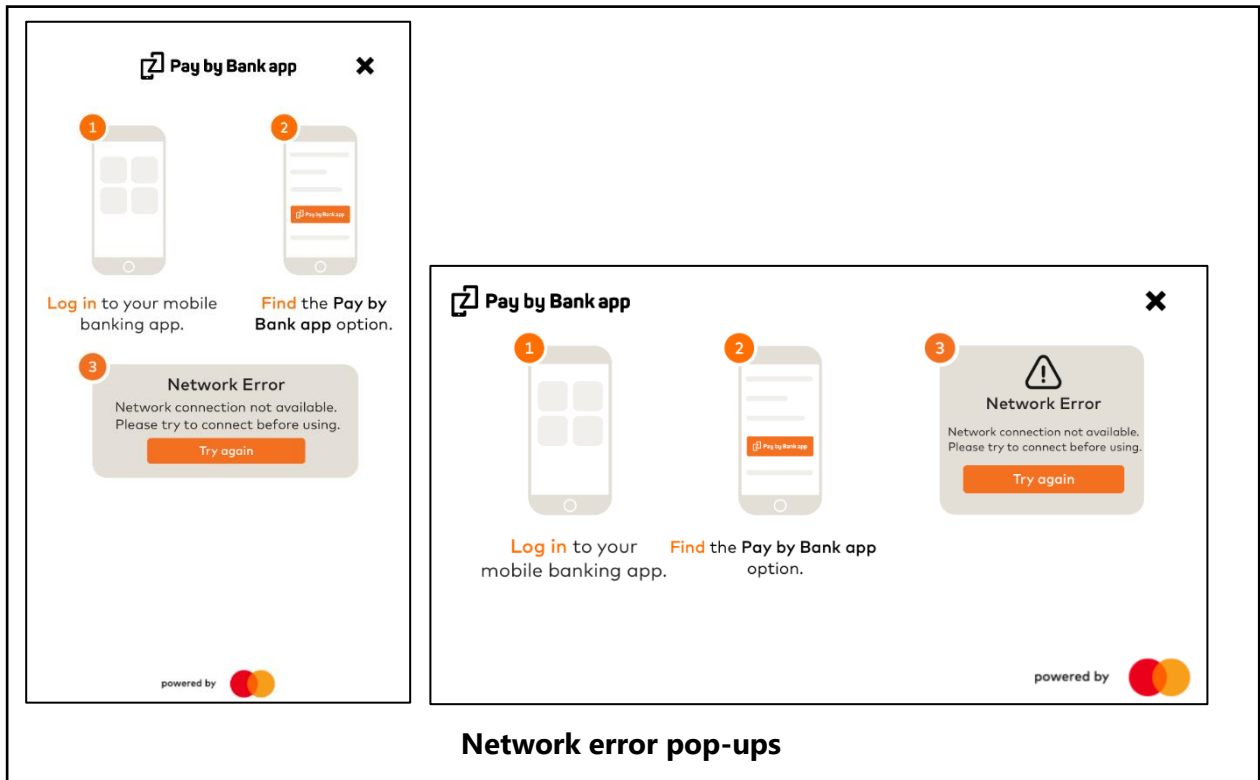
Network error pop-ups

3.5.1.5   Request expired error pop-ups
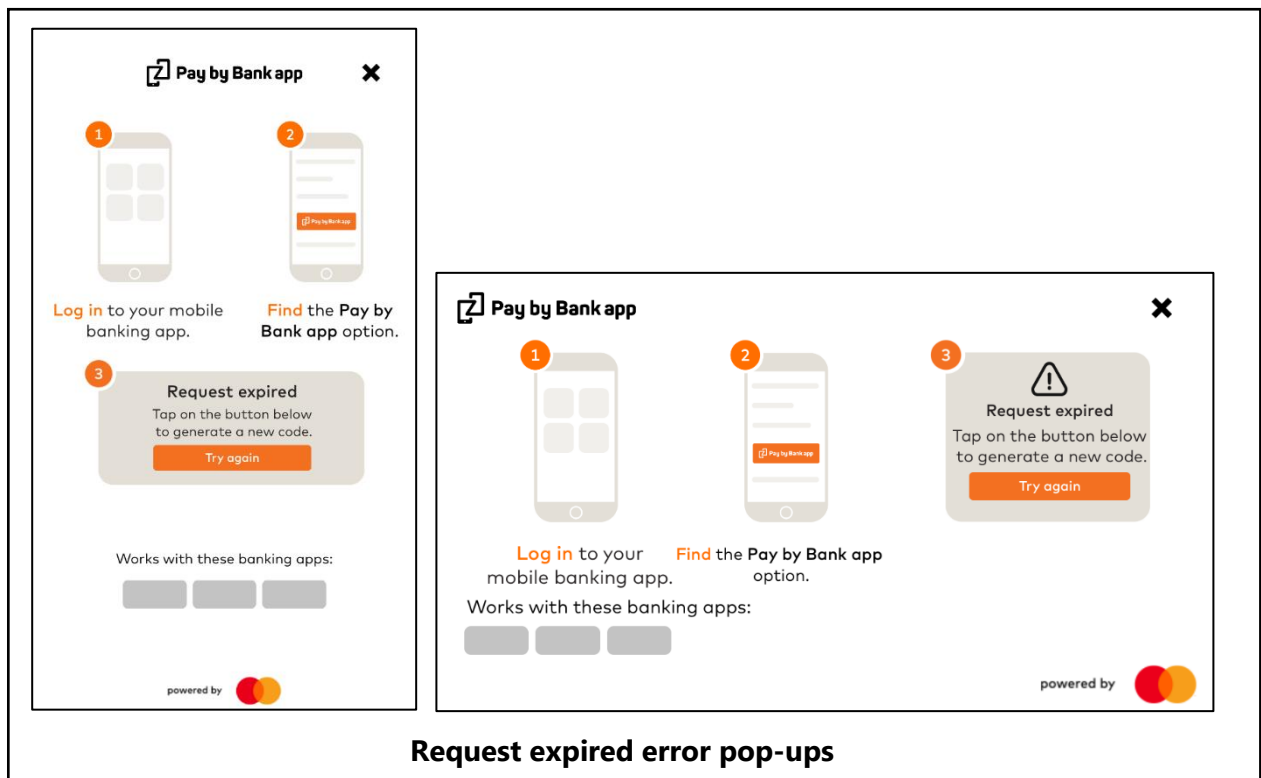
Request expired error pop-ups

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

3.5.1.6   Pay by Bank app code generation error pop-ups
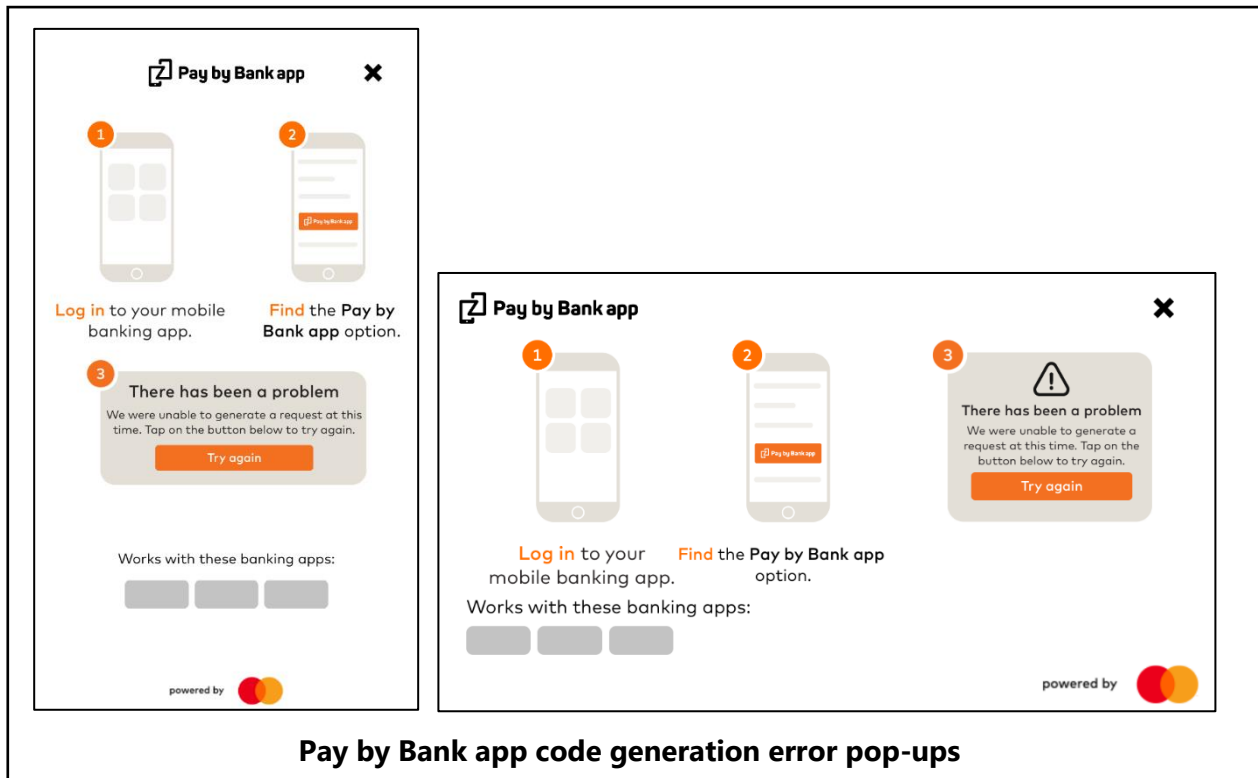
Pay by Bank app code generation error pop-ups

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

3.5.1.7   More about Pay by Bank app pop-ups
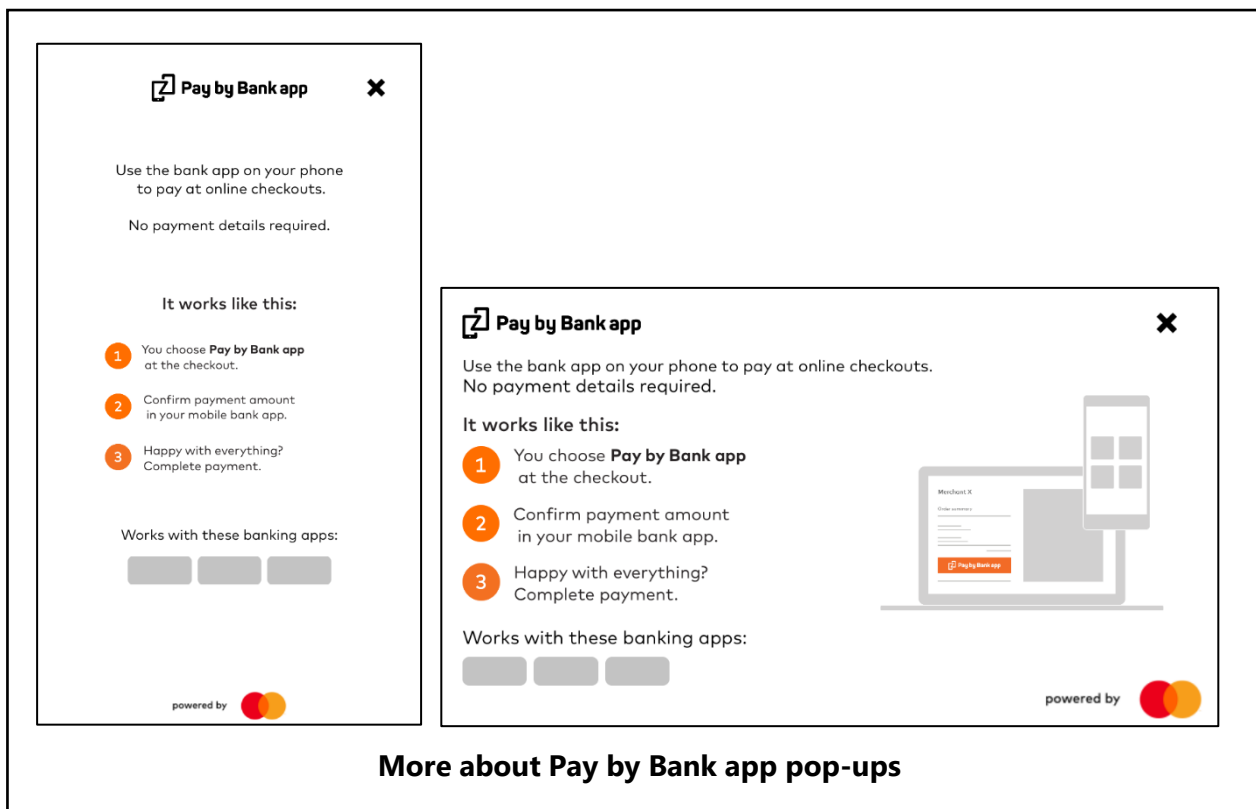
More about Pay by Bank app pop-ups

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

3.5.1.8   Integrated custom iOS merchant button layouts

Integrated Custom iOS 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

3.5.1.9   Integrated PBBA Custom iOS Merchant Button with PBBA logo

3.5.1.10   Integrated PBBA Custom iOS 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 iOS Merchant Button Library.

4.2 Integration Steps of an M-COMM journey

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

Procedure steps

1. Integrate the Pay by Bank app Integrated iOS 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 Implement the Integrated iOS Merchant Button

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

```
@interface ViewController ()

@property (nonatomic, weak) IBOutlet UIButton *paymentButton;

@end

@implementation ViewController


- (IBAction)paymentButtonDidPress:(UIButton *)paymentButton
{
    // Start the submit payment request

}

@end
```

4.2.2 Submit payment request to Merchant backend

The paymentButtonDidPress: method is called when the Consumer taps the Integrated iOS Merchant Button. The Merchant has to submit a payment request to the Merchant backend.. The format / protocol to use is left to the discretion of 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 iOS provides a feature to display the Pay by Bank app branded pop-ups for the payment Transaction:

```
#import <ZappMerchantLib/PBBAAppUtils.h>


// Display PBBA popup

[PBBAAppUtils showPBBAPopup:popupPresenter secureToken:secureToken brn:brn
expiryInterval:expiryInterval delegate:popupDelegate];
```

Parameter name	Parameter description	Parameter source
popupPresenter	The instance of view controller which will present the Pay by Bank app pop-up.	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 period.	< Consult Distributor Documentation >
expiryInterval	The time interval	< Consult Distributor Documentation >
popupDelegate	The Pay by Bank app pop-up delegate instance.	Provided by the Merchant App.

Table 4:   Parameters for `show Pay by Bank app pop-up' method call

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 Pay by Bank app code automatically.

Important   In order for the library to be able to check if a Pay by Bank app enabled CFI App is installed, it is required to add **LSApplicationQueriesSchemes** key entry to the application **Info.plist** file with the string value: **zapp**. It specifies the common Pay by Bank app URL scheme, used by the Pay by Bank app enabled CFI Apps.

How to implement the pop-up callback

1. A callback will be received when the PBBA Pop-up is dismissed.

2. A sample code of how this callback should be implemented is as follows:

```
#pragma mark - PBBAPopupViewControllerDelegate

- (void)pbbaPopupViewControllerRetryPaymentRequest:(PBBAPopupViewController
*)pbbaPopupViewController
{
    // Not applicable for successful responses (only error popup)
}

- (void)pbbaPopupViewControllerDidCloseByUser:(PBBAPopupViewController
*)pbbaPopupViewController
{
    // Will be called when the Consumer taps on the top-right corner of the
popup and dismissed it.
}

- (void)pbbaPopupViewControllerDidExpire:(PBBAPopupViewController
*)pbbaPopupViewController
{
    // Will be called when the popup expired
}
```

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

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

```
#import <ZappMerchantLib/PBBAAppUtils.h>

// Display PBBA error popup
[PBBAAppUtils showPBBAErrorPopup:popupPresenter errorCode:errorCode
errorTitle:errorTitle errorMessage:errorMessage delegate:popupDelegate];
```

Parameter name	Parameter description	Parameter source
popupPresenter	The instance of view controller which will present the Pay by Bank app pop-up.	Provided by the Merchant App.
errorCode	The error code to be displayed along with error message in the Pay by Bank app pop-up.	Provided by the Merchant App.
errorTitle	The error title to be displayed along with error message in the Pay by Bank app pop-up.	Provided by the Merchant App.
errorMessage	The error message to be displayed in the Pay by Bank app Pop-up.	Provided by the Merchant App.
popupDelegate	The Pay by Bank app pop-up delegate instance.	Provided by the Merchant App.

Table 5:   Parameters for `show Pay by Bank app error pop-up' method call

The length of the strings displayed in the Pay by Bank app 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 Pay by Bank app 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:

```
#pragma mark – PBBAPopupViewControllerDelegate

- (void)pbbaPopupViewControllerRetryPaymentRequest:(PBBAPopupViewController
*)pbbaPopupViewController
{
    // 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
}

- (void)pbbaPopupViewControllerDidCloseByUser:(PBBAPopupViewController
*)pbbaPopupViewController
{
    // Will be called when the Consumer taps on the top-right corner of the
popup and dismissed it.
}
```

4.2.4   Get payment status from Merchant backend

The Merchant should implement the logic of 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 (that is, 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 iOS Pay by Bank app Custom Merchant Button Layouts

The Integrated iOS Custom Merchant button should be implemented using getCustomUXConfigurationsWithWidth:(CGFloat)width

andType:(PBBACustomUXType)customUXType class method of PBBAAppUtils class which returns the Custom merchant button layout component. The rendering and the tap event handling of the iOS Custom Merchant Button layout should be provided by the Merchant Application.

The size for the custom merchant button layout component differs based on the PBBACustomUXType enum value. The minimum width for the custom button layout is 200 pts.

Integrated Custom iOS Merchant button layout types:

Layout	Enum value
Integrated PBBA Custom iOS Merchant Button with the PBBA logo	PBBACustomButtonTypeNone
Integrated PBBA Custom iOS Merchant Button with More about Pay by Bank app link	PBBACustomButtonTypeMoreAbout

Note Merchant's 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 Pay by Bank app custom layout component integration.

```
#import "ZappMerchantLib.h"

@interface ViewController ()

@property (nonatomic, weak)IBOutlet UIView *integratedPBBAButton;

@end

@implementation ViewController

-(void) viewDidLoad
{
    [super viewDidLoad];
    self.integratedPBBAButton =  [PBBAAppUtils
getCustomUXConfigurationsWithWidth:200 andType:(PBBACustomUXType)indexPath.row];
}
    @end
```

+(UIView*)getCustomUXConfigurationsWithWidth:(CGFloat)width
andType:(PBBACustomUXType)customUXType

Parameter name	Parameter description	Parameter source
width	CGFloat param to update the button width.	Provided by the Merchant App.
customUXType	Enum to get the integrated PBBA button with CFI logos, More About Pay by Bank app Button. typedef enum { PBBACustomButtonTypeMoreAboutAndBankLogos, PBBACustomButtonTypeBankLogos, PBBACustomButtonTypeMoreAbout, PBBACustomButtonTypeNone, } PBBACustomUXType;	Provided by the Merchant App.

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 Pay by Bank app enabled CFI App installed on the device and displays the E-COMM version of the Pay by Bank app 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 <ZappMerchantLib/PBBAAppUtils.h>
```

```
// Check if Consumer's device supports PBBA payments
if ([PBBAAppUtils isCFIAppAvailable]) {
    // The device has PBBA enabled CFI App installed
} else {
    // The device does not have PBBA enabled CFI App installed
}
```

Note   This is an optional utility API which might be used for testing purpose.

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

The Merchant Button Library provides a feature to invoke the Pay by Bank app enabled CFI App outside the Pay by Bank app pop-up.

Important   The Operator recommends **not** using this function unless it is completely necessary:

```
#import <ZappMerchantLib/PBBAAppUtils.h>

// Invoke the PBBA CFI App
[PBBAAppUtils openBankingApp:secureToken];
```

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

Parameter name	Parameter description	Parameter source
secureToken	The unique token that identifies the payment request.	< Consult Distributor Documentation >

Table 6:   Parameters for `open banking app' method call

Note   This is an optional utility API which might be used for testing purpose.

5   Sample code

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

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

```
#import <ZappMerchantLib/ZappMerchantLib.h>

#import "MerchantPaymentViewController.h"

@interface MerchantPaymentViewController () <PBBAPopupViewControllerDelegate>

@property (nonatomic, weak) IBOutlet UIButton *customMerchantButton;

@property (nonatomic, strong) MerchantNetworkService networkService;
@property (nonatomic, strong) MerchantPaymentDetails paymentDetails;

@end

@implementation MerchantPaymentViewController

- (IBAction)customMerchantButtonDidPress:(UIButton *)customMerchantButton
{
    [self submitPayment];
}

- (void)submitPayment
{
    // The Merchant App uses the network service to make an async HTTP request to the Merchant gateway.
    // Merchant network service receives the response e.g. in JSON format, parses it to an object which is called e.g.
    // Transaction and returns this object in the provided callback. An error object is returned if payment request fails.
    [self.networkService submitPaymentRequest:self.paymentDetails completion:^(Transaction *Transaction, NSError *error) {

        if (error) {
            [self didReceiveError:error];
        } else {
            [self didReceiveTransaction:Transaction];
        }
```

```objc
    }];
}

- (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
    // an enum type e.g. PaymentStatus
    [self.networkService getPaymentStatus:^(PaymentStatus status, NSError *error) {

        if (error) {
            [self didReceiveError:error];
        } else {
            [self didReceivePaymentStatus:status];
        }
    }];
}

#pragma mark - Request Callbacks

- (void)didReceiveTransaction:(Transaction *Transaction)
{
    // In case of standard m-Comm:
    //    If it is the very first time payment then the PBBA popup will appear.
    //    If this is not the first payment then 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:self
                    secureToken:Transaction.secureToken
                            brn:Transaction.brn
                 expiryInterval: Transaction.retrievalExpiryInterval
                       delegate:self];


}

- (void)didReceiveError:(NSError *error)
{
    [PBBAAppUtils showPBBAErrorPopup:self
                           errorCode:nil
                          errorTitle:NSLocalizedString(@"Error", nil)
                        errorMessage:error.localizedDescription
                            delegate:self];
}
```

```
- (void)didReceivePaymentStatus:(PaymentStatus status)
{
    if (status == PaymentStatusInProgress) {
        [self getPaymentStatus];
    } else
        // For m-Comm payment finished, display Merchant payment status page
        // For e-Comm payment finished, dismiss PBBA Popup
    }
}

#pragma mark - PBBAPopupViewControllerDelegate

- (void)pbbaPopupViewControllerRetryPaymentRequest:(PBBAPopupViewController *)pbbaPopupViewController
{
    // This method is called only when the Consumer clicks on the 'Pay by Bank app' button
    // on the PBBA error popup
    [self submitPayment];
}

- (void)pbbaPopupViewControllerDidCloseByUser:(PBBAPopupViewController *)pbbaPopupViewController
{
    // This method is called on any PBBA popup when the Consumer taps the dismiss button
    // on the top-right corner of the PBBA popup
    self.customMerchantButton.enabled = YES;
}

- (void)pbbaPopupViewControllerDidExpire:(PBBAPopupViewController *)pbbaPopupViewController
{
    // This method is called only when the Popup has expired
    // on the PBBA error popup
    [self showErrorPopupForExpiredTransaction];
}

@end
```

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 iOS Merchant Button. Pay by Bank app does not mandate that the native library must be used for Hybrid apps.