



# **EIS11 PBBA Branded Web Merchant Button**

## Implementation Guide

March 2023

Document Version 3.12.5

GitHub Merchant Library Version 3.1.2

Release R3.1



**Copyright statement**

The information contained in this document is confidential and proprietary to Vocalink Limited, its successors or assignees and (if applicable) its prospective or actual customers/partners. The copyright in this document is owned by Vocalink Limited, or its successors or assignees. This document shall not be used, disclosed or copied in whole or in part for any purposes without the express permission of the owner.

© Copyright 2020 Mastercard. All rights reserved

## Document History

Version		Date	Summary of Changes
Was	Now		
1.3		18-11-2015	Final Version 1 Draft
2.0	1.4	19-11-2015	Revised to reflect split of basket setup and Transaction flow: removing all product fields except for a merchant product/basket reference id. Removed APTid references. Changed APTrid references to "secureToken". Removed repetition in the code descriptions.
2.0	1.5	29-11-2015	Removed hosted option as deprecated following browser standards and security reviews.
2.0		20-05-2016	Released to support Pay by Bank app live service.
2.1		23-06-2016	Updated document to add appendix and sequence diagrams.
2.2		06-07-2016	Updated with the Theme information and peer review comments
2.3		25-07-2016	Added information about cookieExpiryDays Added the GitHub location for the web merchant button library Updated the external sharepoint location for the web merchant button library Removed unwanted code as per my previous mail Updated the cookie management URL to <a href="https://www.paybybankapp.co.uk/">https://www.paybybankapp.co.uk/</a>
2.4		11-10-2016	Cleaned up examples to remove unwanted fields. Updated the PacyConnect URL in the examples. Added steps to continue polling upon receipt of a payment not confirmed status.

Version		Date	Summary of Changes
Was	Now		
2.5		21-10-2016	Added the setCookie() description to Appendix A. Added support for Safari
2.6		25-10-2016	Added section 5.5 to Appendix C on cookie management
2.7		07-11-2016	Updated the document after joint review with participant
2.8		08-11-2016	Review complete and Final draft released
2.9		30-11-2016	Merchant Button version 2.0.2 changes updated and peer review
3.0	2.10	11-01-2017	Updated screenshots to show "Pingit" instead of "your banking app".
3.1	2.11	24-01-2017	Updated the document for custom merchant button related changes.
3.2	2.12	01-03-2017	Update re polling for current status.
3.3	2.13	30-03-2017	Updated Links to the latest Web Merchant Button Library
3.0		03-05-2017	<ul style="list-style-type: none"> <li>Document format revised for R3 issue.</li> <li>Added instructions to enable/disable the hover over popup.</li> <li>Added a section with steps to upgrade the library.</li> <li>Updated popup close confirmation box screenshot and description.</li> <li>Consumer edge cases added to Appendix.</li> <li>Update to DDOS section regarding use of Cookie token</li> </ul>
3.1		15-05-2017	A.1.4 Pay by Bank app Cookie Management Component <ul style="list-style-type: none"> <li>Additional text and table added</li> </ul>
3.2		01-07-2017	Changes to this version: <ul style="list-style-type: none"> <li>Document format revised for R3 issue</li> </ul>

Version		Date	Summary of Changes
Was	Now		
			<ul style="list-style-type: none"> <li>• Document History revised to ensure versions align to Releases</li> <li>• Updated Links to the latest Web Merchant Button Library</li> <li>• Section 3.4 – Note amended to explain that although the version number mentions R2, the code itself will work with Release 3 of PBBA</li> <li>• Section 3.6 – The text (Standard Offering) removed from the title and the following paragraph</li> <li>• Section 3.6.4: <ul style="list-style-type: none"> <li>▪ Note following step 4 – Removed</li> <li>▪ Section 3.6.4 step 5e – The screenshots within this step updated to reflect the latest versions</li> </ul> </li> <li>• Table 3 – Reference to 'iPhone 5' changed to 'iPhone SE'</li> <li>• Figure 2 – App Picker – image updated</li> <li>• Figure 5 – Updated screenshot showing version number</li> <li>• Figure 10 – Screen #3 updated to reflect the latest version</li> <li>• Appendix A.8.3 – Consumer starts the payment journey in a non-default browser but is returned to the Merchant website with the device default browser post payment- Added</li> <li>• Appendix A.9 – Hybrid Merchant Apps – Added</li> </ul>
3.3		23-10-2017	<p>Note added clarifying the polling mechanism on:</p> <ul style="list-style-type: none"> <li>• Section 2.2 – M-COMM journey</li> <li>• Section 2.3 – E-COMM journey - Pay by Bank app code journey</li> <li>• Section 2.4 - E-COMM – PayConnect journey</li> </ul>

Version		Date	Summary of Changes
Was	Now		
	3.4	01-11-2017	Released to support Pay by Bank app R3 live service.
	3.5 - 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	References to Pingit removed
	3.12.2	11-02-2019	Introducing version 3.1.0 of the web merchant button library with updated user interface
	3.12.2	27-05-2019	<p>Changes to this version in relation GitHub Merchant Library Version 3.0.0: (as Published in June 2019)</p> <ul style="list-style-type: none"> <li>• Section 1.1 - Introduction - Note on Level AA standard - added</li> <li>• Section 3.2 - Certified Browsers and Devices - table updated</li> <li>• Section 3.6.1 - Button component - screenshot updated</li> <li>• Section 3.6.2 - Popup component - revised for clarity</li> <li>• Section 3.6.2.a. - PBBA Code Popup - added</li> <li>• Section 3.6.2.b. - More-About Popup - added</li> <li>• Section 3.6.4 - Pay by Bank app Branded Merchant Button setup: <ul style="list-style-type: none"> <li>○ Procedure steps 3 - revised for clarity</li> <li>○ Procedure steps 5 - revised for clarity</li> <li>○ Procedure steps 5e - screenshots updated</li> </ul> </li> <li>• Section 4.3 - Integrated Web Merchant Button - screenshot removed</li> <li>• Appendix A.1.2 - Enabling or disabling the hover over pop-up – removed</li> </ul>

Version		Date	Summary of Changes
Was	Now		
			<ul style="list-style-type: none"> <li>Appendix A.1.3 - Pay by Bank app Cookie Management Component becomes A.1.2</li> <li>Appendix A.2 - Pay by Bank app Button implementation sample code - revised for clarity</li> <li>Appendix A.3 - Changing the look and feel of the Button – removed</li> <li>Appendix A.4 - Additional Cookie management Information becomes A.3</li> <li>Table 5 - Web Merchant Button Library download locations - revised for clarity</li> <li>Figure 5 - Branded Pay by Bank app Web Merchant Button Library structure - revised for clarity</li> <li>Figure 9 - Mobile App Cookie - screenshot updated</li> </ul>
3.12.2		27-05-2019	<ul style="list-style-type: none"> <li>Figure 10 - M-COMM Journey on a mobile browser - screenshot updated</li> <li>Figure 11 - M-COMM journey with another device - screenshot updated</li> </ul>
3.12.3		11-09-2019	<p>Changes to this version:</p> <ul style="list-style-type: none"> <li>Section 1 - Introduction - Important and Note - revised for clarity</li> <li>Section 3.2 - Certified Browsers and Devices - iPad Mini 2 and iPad Mini 4 - OS Version added</li> <li>Section 3.4 - Branded Pay by Bank app Web Merchant Button Library structure - revised for clarity</li> <li>Section 3.5.1 - General requirements: <ul style="list-style-type: none"> <li>Table1 - JQuery version - amended</li> <li>&lt;script - amended</li> </ul> </li> </ul>

Version		Date	Summary of Changes
Was	Now		
			<ul style="list-style-type: none"> <li>• Section 3.6.1 - Button component - Note - added</li> <li>• Section 3.6.2.b - More-About Popup. - Note - added</li> <li>• Section 3.6.4 - Pay by Bank app Branded Merchant Button setup: <ul style="list-style-type: none"> <li>○ Procedure steps 2 - Example- revised for clarity</li> <li>○ Procedure steps 3 - revised for clarity</li> <li>○ Procedure steps 5d - Note - added</li> </ul> </li> <li>• Appendices A.1.2 - Pay by Bank app Cookie Management Component - jquery version- amended</li> <li>• Appendices A.2 - Pay by Bank app Button implementation sample code: <ul style="list-style-type: none"> <li>○ Revised for clarity</li> <li>○ Note - revised for clarity</li> </ul> </li> <li>• Appendices A.9 - Configuring Content Security Policy (CSP) - added</li> <li>• Table 5 - Download the latest version table removed and replace with a link</li> <li>• Table 7 - Cookie management component - revised for clarity</li> </ul>
3.12.4		14-10-2022	<ul style="list-style-type: none"> <li>• External GitHub URL change to internal Mastercard GitHub URL</li> </ul>
3.12.5		14-03-2023	Updated font library



## Contents

---

<b>1</b>	<b>About this document</b> .....	<b>12</b>
1.1	Introduction .....	12
1.2	Audience.....	12
1.3	Scope.....	12
1.4	Document conventions.....	12
1.5	Associated documents.....	13
<b>2</b>	<b>Functional overview</b> .....	<b>14</b>
2.1	Introduction .....	14
2.2	M-COMM journey .....	14
2.2.1	App Picker.....	16
2.3	E-COMM – Pay by Bank app code journey.....	17
2.4	E-COMM – PayConnect journey.....	19
<b>3</b>	<b>Technical overview</b> .....	<b>22</b>
3.1	Introduction .....	22
3.2	Certified Browsers and Devices .....	22
3.3	Hosting options.....	23
3.4	Branded Pay by Bank app Web Merchant Button Library structure.....	23
3.5	Technical requirements.....	25
3.5.1	General requirements.....	25
3.5.2	Library hosting requirements.....	25
3.6	Branded Web Merchant Button .....	25
3.6.1	Button component .....	26
3.6.2	Popup component.....	26
3.6.3	Cookie management component.....	29
3.6.4	Pay by Bank app Branded Merchant Button setup .....	32
3.6.5	Pay by Bank app Branded Merchant Button sample code.....	44
<b>4</b>	<b>Additional considerations for M-COMM</b> .....	<b>45</b>
4.1	Prerequisites for M-COMM.....	45
4.2	Mobile App Cookie – retaining PBBA enabled Bank App selection .....	45
4.2.1	Setting hasApp cookie .....	45
4.3	Integrated Web Merchant Button .....	49
<b>A</b>	<b>Appendices</b> .....	<b>50</b>

A.1	Merchant Configurable Properties for the PBBA Branded Web Merchant Button.....	50
A.1.1	Merchant Poll Intervals.....	50
A.1.2	Pay by Bank app Cookie Management Component.....	50
A.2	Pay by Bank app Button implementation sample code.....	53
A.3	Additional Cookie management Information.....	58
A.3.1	Remove all connections to the Mobile Banking Application Consumer function.....	58
A.3.2	DDoS protection.....	58
A.4	Known browser specific requirements.....	58
A.4.1	Internet Explorer.....	58
A.5	Polling for Payment Status.....	60
A.6	Upgrading the library.....	60
A.7	Handling Consumer edge cases.....	60
A.7.1	Consumer selects Pay by Bank app more than once per order.....	60
A.7.2	Consumer retrieves payment in Mobile Banking app but pays with another payment method.....	60
A.7.3	Consumer starts the payment journey in a non-default browser but is returned to the Merchant website with the device default browser post payment.....	61
A.8	Hybrid Merchant Apps.....	61
A.8.1	Hybrid iOS Apps.....	61
A.8.2	Hybrid Android Apps.....	62
A.9	Configuring Content Security Policy (CSP).....	63

## Tables

---

Table 1:	Web Merchant Button library – Setup requirements.....	25
Table 2:	Cookie management component.....	30
Table 3:	PayConnect ID mapping to Request to Pay.....	35
Table 4:	Response to Request to Pay Mapping.....	37
Table 5:	PayConnect cookie setting for Pay Status Authorised.....	43

## Figures

---

Figure 1:	Interaction between the components of the M-COMM journey.....	15
Figure 2:	App Picker – sample screens.....	16

---

Figure 3:	Interaction between the components of the E-COMM PBBA code journey .....	18
Figure 4:	Interaction between the components of the E-COMM PayConnect journey .....	20
Figure 5:	Branded Pay by Bank app Web Merchant Button Library structure .....	24
Figure 6:	PBBA Popup component.....	27
Figure 7:	More about popup. ....	28
Figure 8:	PayConnect Cookie (pcid).....	31
Figure 9:	hasApp Cookie .....	31
Figure 10:	Mobile App Cookie.....	45
Figure 11:	M-COMM Journey on a mobile browser .....	47
Figure 12:	M-COMM journey with another device .....	48

# 1 About this document

---

## 1.1 Introduction

This document describes the Pay by Bank app (PBBA) Merchant Button Library for Web. The focus is on the Pay by Bank app Branded Web Merchant Button behaviour/code and provides a functional and technical overview for M-COMM, E-COMM and E-COMM PayConnect Consumer journeys.

**Important** This version of EIS11 PBBA Branded Web Merchant Button Implementation Guide accompanies Version 3.1.2 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** Version 3.1.2 of 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 logo, 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 Branded Web 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
<b>Important</b>	Highlights important text in the document.
<b>Notes</b>	Provides more information about a topic.
<a href="#">Number Title text</a>	Hyperlink to another section in the document.
<i>Italics</i>	Indicates a document name.
<code>Courier New</code>	Indicates code / command.

## 1.5 Associated documents

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

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

---

## 2 Functional overview

---

### 2.1 Introduction

The Pay by Bank app Web Merchant Button enables Merchants and Distributors to use Pay by Bank app as a payment method. Written in JavaScript, the Web Merchant Button library can be included on any Website by following a few simple steps.

The Pay by Bank app Web Merchant Button supports two different models:

1. Pay by Bank app Branded Web Merchant Button

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

2. Pay by Bank app Integrated Web Merchant Button with Pay by Bank app Popup

Merchants and Distributors can integrate their payment button with the Pay by Bank app Integrated Web Merchant Button. The additional considerations are covered in the *PBBA Integrated Web 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 Website or App is opened on the same device as the Pay by Bank app CFI App. A sample Consumer journey includes the following steps:

- The Consumer clicks on a Pay by Bank app button which starts the payment. This document covers the standard PBBA branded Merchant Button only.
  - If this is the first time Pay by Bank app has been used on the device and there is at least one PBBA enabled CFI App installed on the device then the Pay by Bank app Popup will appear asking the Consumer to either continue his payment on the same device by pressing 'Open mobile banking app' or get the Pay by Bank app code to pay on another device.
  - If this is not the first payment on the device and the Consumer has selected 'Open mobile 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 offered
- The Consumer can approve or cancel the Transaction
- When the payment has been completed, the Merchant App displays the payment confirmation page and also stores the Consumers choice of using Pay by Bank app on the same device on that browser for future payments

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

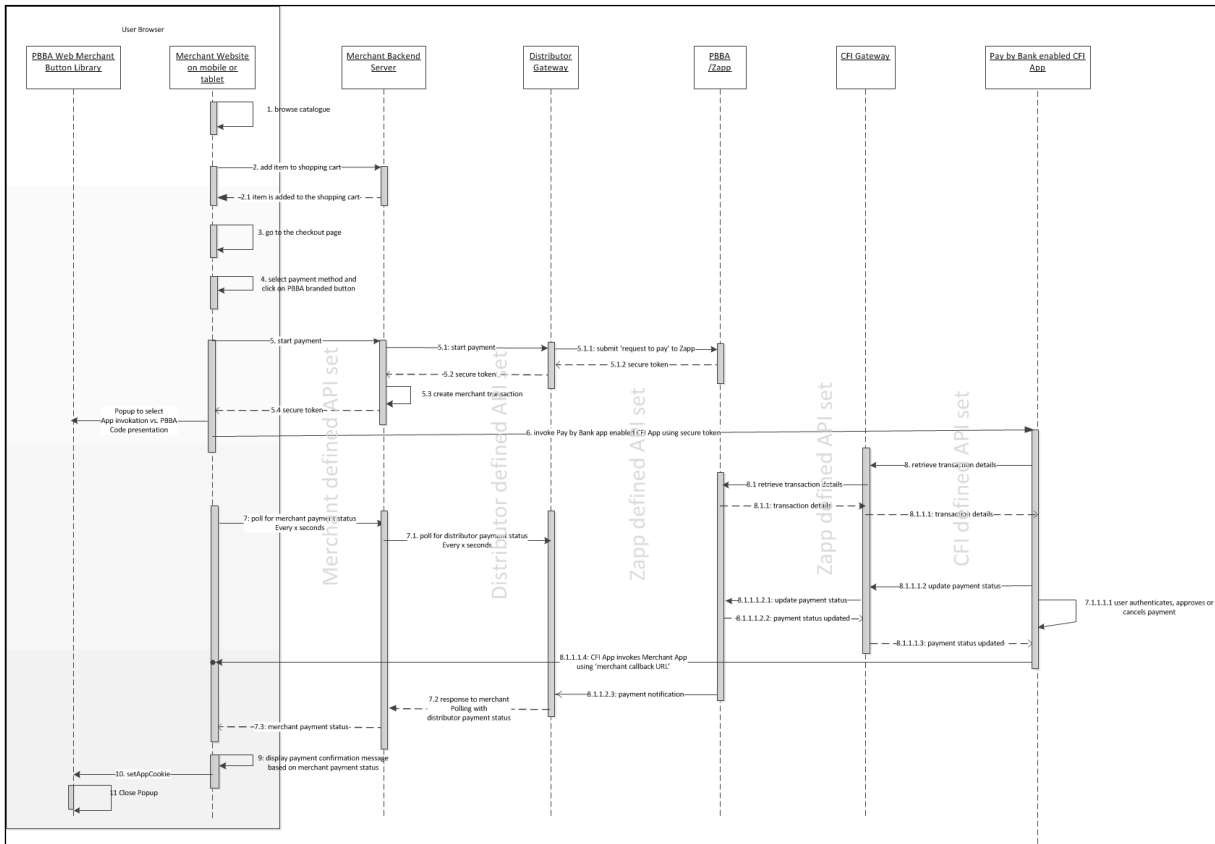


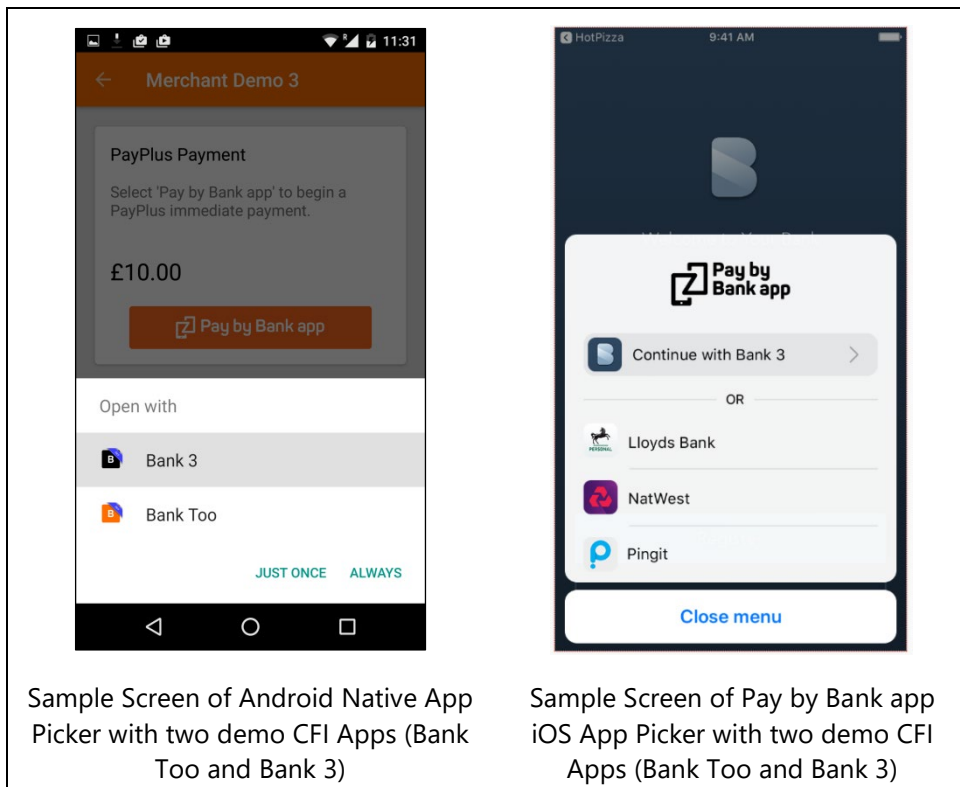
Figure 1: Interaction between the components of the M-COMM journey

**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 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 being enabled, CFI App is installed on the same device that the Merchant App is running on and the App Picker is displayed (see Figure 1:

Interaction between the components of the M-COMM journey Figure 2: App Picker – sample screens 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 Android Native App Picker with two demo CFI Apps (Bank Too and Bank 3)

Sample Screen of Pay by Bank app iOS App Picker with two demo CFI Apps (Bank Too and Bank 3)

Figure 2: App Picker – sample screens



---

## 2.3 E-COMM – Pay by Bank app code journey

The Merchant Website 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 clicks the button which starts the payment. This document covers the Pay by Bank app Branded Web Merchant button only
- The Merchant Website displays a branded Popup with a six letter code (known as the Pay by Bank app code or Pay by Bank app code to the Consumer
- The Consumer starts a Pay by Bank app enabled CFI App on another device and enter Pay by Bank app code to retrieve the Transaction
- The Consumer can approve or cancel the Transaction
- If the Consumer approves the Transaction and if the PayConnect ID was not presented to the Zapp server, then they are presented with an option to enable PayConnect. The Consumer either selects or cancels the PayConnect option.
- When the payment has been completed, the Merchant Website displays the payment confirmation page
- If the Consumer did select the PayConnect option in the Pay by Bank app enabled CFI App, then a PayConnect ID and Expiry Days data is send by Distributor to Merchant along with the Payment Status. This will then be set on the Consumer's Browser as a Cookie for future PBBA Payment from this specific browser as an E-COMM PayConnect Journey.
- If the Consumer cancels the PayConnect option in the Pay by Bank app enabled CFI App, future Journey will be PBBA code journey

**Note** The PayConnect feature connects a consumer's browser to the consumer's Pay by Bank app enabled CFI App on another device

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

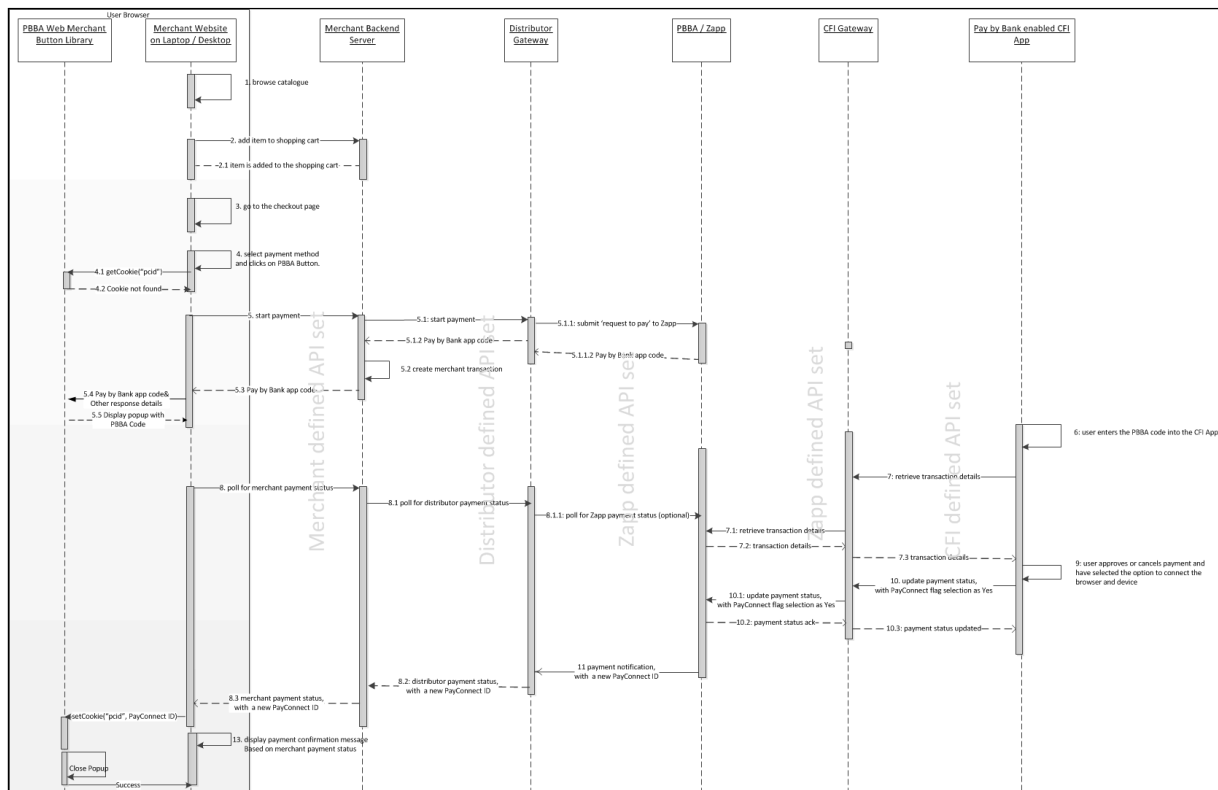


Figure 3: Interaction between the components of the E-COMM PBBA code journey

**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 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.

---

## 2.4 E-COMM – PayConnect journey

The Merchant Website and the Pay by Bank app CFI App are on different devices. This journey assumes that a PayConnect cookie is set on the consumer browser from previously completed E-COMM PBBA code journey with the Consumer selecting the PayConnect option in the Pay by Bank app enabled CFI App.

A sample Consumer journey includes the following steps:

- The Consumer selects a Pay by Bank app method and clicks on the button which starts the payment, the payment request will now include the PayConnect ID retrieved by PBBA Button from the PayConnect cookie on the browser
- The Merchant Website displays a Pay by Bank app branded 'notification sent' Popup
- The Consumer gets a push notification on the Pay by Bank app enabled CFI App device, this device was originally linked with the PayConnect Cookie and will be used to establish the PayConnect journey
- The Consumer clicks on the push notification which starts the Pay by Bank app enabled CFI App on the device and retrieves the Transaction
- The Consumer can approve or cancel the Transaction
  - Note** The Consumer will not be prompted to link the browser and device again as they had done it before
- When the payment has been completed, a new PayConnect ID and Expiry Days data is sent by Distributor to Merchant along with the Payment Status, this new cookie will replace the previous cookie on the browser
- The Merchant Website displays the payment confirmation or cancellation page

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

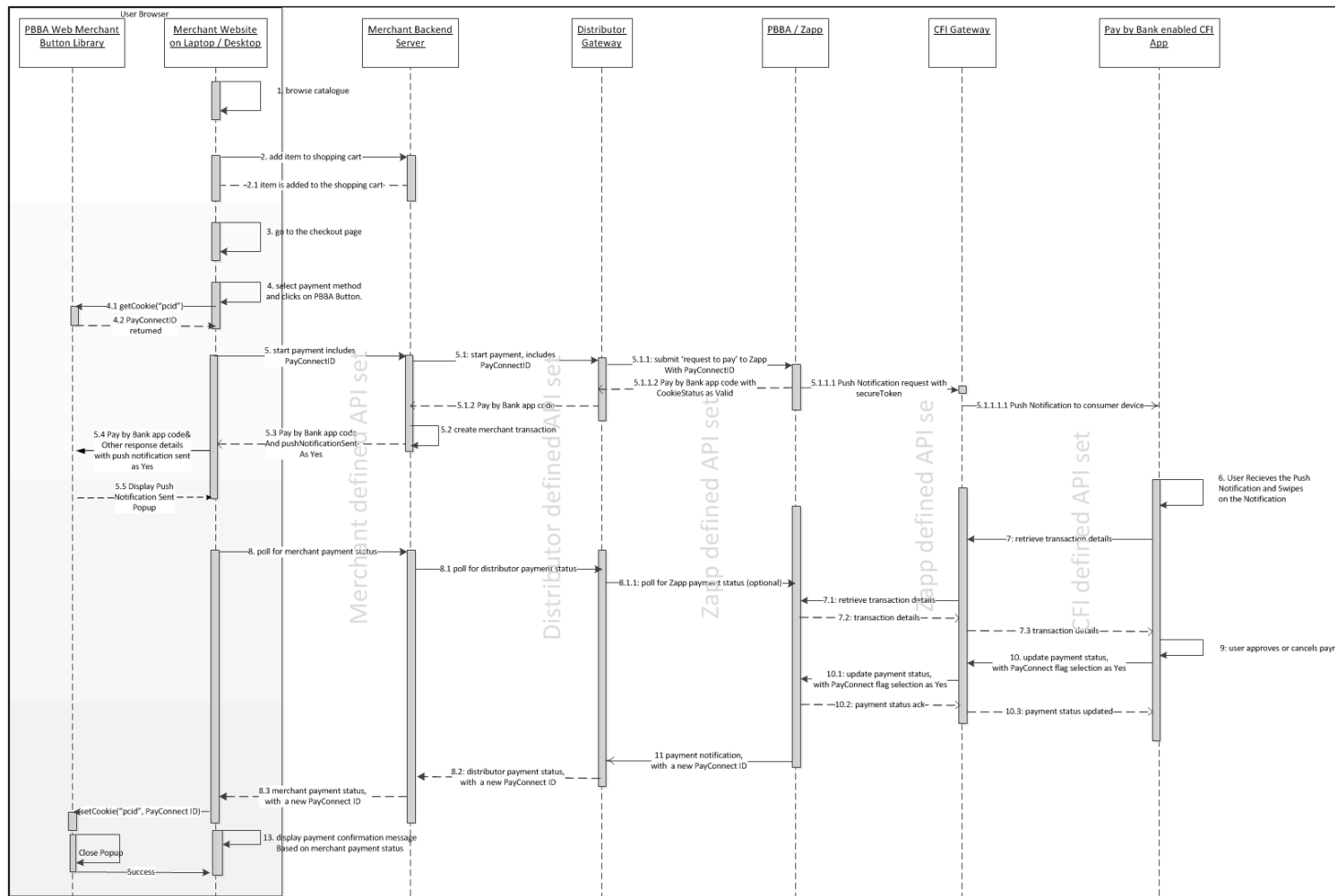


Figure 4: Interaction between the components of the E-COMM PayConnect journey

**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 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 Introduction

This section provides instructions on the implementation of the Branded Web Merchant Button.

Each section contains:

- Setup instructions
- An explanation of each Web Merchant Button Library component
- Examples of the use of the library

### 3.2 Certified Browsers and Devices

Zapp has certified the Web Merchant Button library to work with the browsers, mobile devices and operating systems mentioned below:

	Operating System/ Browser	OS Version	Comments
<b>Web</b>			
	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	
<b>Mobile (iOS)</b>			
	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	
<b>Mobile (Android)</b>			
	Samsung Galaxy S8	8.0.0	

	Operating System/ Browser	OS Version	Comments
	Samsung Galaxy S7 edge	6.0.1	
	Google Pixel	9	
	Samsung Galaxy S9	8.0.0	
<b>iPAD and Tablet</b>			
	Samsung Tab2	4.1.2	
	Galaxy Tab S4	5.1.1	
	iPad Pro 12.0	12.1.1	
	iPad Pro 10.5	12.1.1	
	iPad Mini 2	12.3.0	
	iPad Mini 4	12.2.0	

**Note** Landscape orientation is not supported for Web on mobile browsers.

Download the latest version from:

<https://github.com/Mastercard/pbba-merchant-button-library-web>

### 3.3 Hosting options

The Pay by Bank app Web Merchant Button Library can be hosted on the Merchant server or on the Hosted Payment Pages provider's server.








- Merchant hosted model – The Web Merchant Button library is hosted on the Merchant's server. This is the usual case for Merchant hosted Websites
- Hosted Payment Page model – The Web Merchant Button Library is hosted on the Hosted Payment Pages provider's server.

**Important** Previously Zapp has offered an option to host the libraries on Zapp servers. Following a careful review, this is now considered to be inappropriate as most browsers are moving to restrict such third party access, and we cannot recommend a solution which may be broken without warning.

### 3.4 Branded Pay by Bank app Web Merchant Button Library structure

The Pay by Bank app (PBBA) Web Merchant Button library is a JavaScript based product. It consists of HTML and JavaScript files, images and CSS files in a folder for the current version of the library. You can also clone the project from [GitHub](#). The overall folder structure is represented in [Figure 5](#) below:

---

 3.1.2	08/03/2023 16:14	File folder	
 jQuery.XDomainRequest.js	07/03/2023 12:13	JS File	4 KB
 pbbacustomconfig.js	08/03/2023 16:14	JS File	11 KB
 pbbacustomconfig_branded.template	08/03/2023 16:14	TEMPLATE File	10 KB
 pbbacustomconfig_custom.template	08/03/2023 16:14	TEMPLATE File	12 KB
 zapp.js	08/03/2023 16:14	JS File	9 KB
 zpopup.js	08/03/2023 16:14	JS File	8 KB

**Figure 5: Branded Pay by Bank app Web Merchant Button Library structure**



## 3.5 Technical requirements

### 3.5.1 General requirements

The following table shows the general requirements to setup the Web Merchant Button library.

Component	Version
JQuery	3.4.1

**Table 1:** Web Merchant Button library – Setup requirements

JQuery is used by the Web Merchant Button library to perform various operations e.g. cookie management, selecting DOM elements, etc. JQuery should be the first script to be imported in the project. JQuery can be included by printing the following HTML script tag in the parent HTML page in the header section:

```
<head>
...
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
...
</head>
```

This will import the JQuery plugins into the project.

**Note** This version of JQuery is the current certified version for Pay by Bank app Popup functioning on the supported browsers and devices, support for the latest version of JQuery is on the Web Merchant Button Product Roadmap and will be considered for future releases of this button.

### 3.5.2 Library hosting requirements

Merchants or Distributors must have a Web server (Apache, IIS or similar) to host the Web Merchant Button library. Download instructions can be found in section [3.6.4 PBBA Branded Merchant Button Setup](#).

## 3.6 Branded Web Merchant Button

The Branded PBBA Button is provided by Zapp and is ready to implement. The Branded button library comes with a button and a Popup. The colours, fonts and styles conform to PBBA standards (refer to the *Brand Guidelines* document for more information) and the button is integrated with the PBBA Popup and cookie management component.

There are three components associated with the Branded Web Merchant Button:

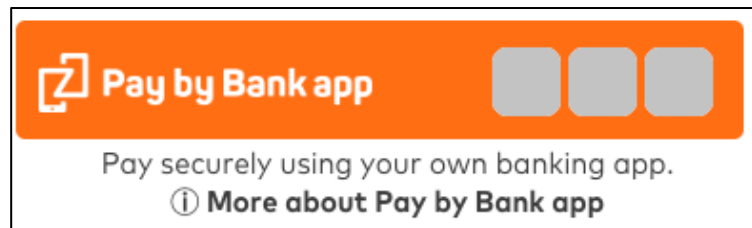
1. Button component This component is the code that represents the button itself covering the functions like on-click, hover events, the look and feel including logic for multi styled buttons.
2. Popup component This component core focus is the Popup styling and its functions including device specific UI responsiveness.

3. Cookie management component This component covers the setting and retrieving of multiple cookies like Has App ('hasApp') and PayConnect ('pcid') cookies.

### 3.6.1 Button component

This component is the code that represents the button itself covering the functions like on-click, hover events, the look and feel (see below) including logic for multi styled buttons.

#### Pay by Bank app Branded:



See section [3.6.4 Pay by Bank app Branded Merchant Button Setup](#) for detailed implementation instructions.

**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 diagram above). 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.

### 3.6.2 Popup component

This component core focus is the Popup styling and its functions including device specific UI responsiveness. The Popup is an out of the box function and relies on data feed like status and other data elements to show appropriate Popups.

#### 3.6.2.a. PBBA Code Popup.

The PBBA popup will be displayed upon clicking the Branded Web Merchant button. This popup will display the PBBA code.

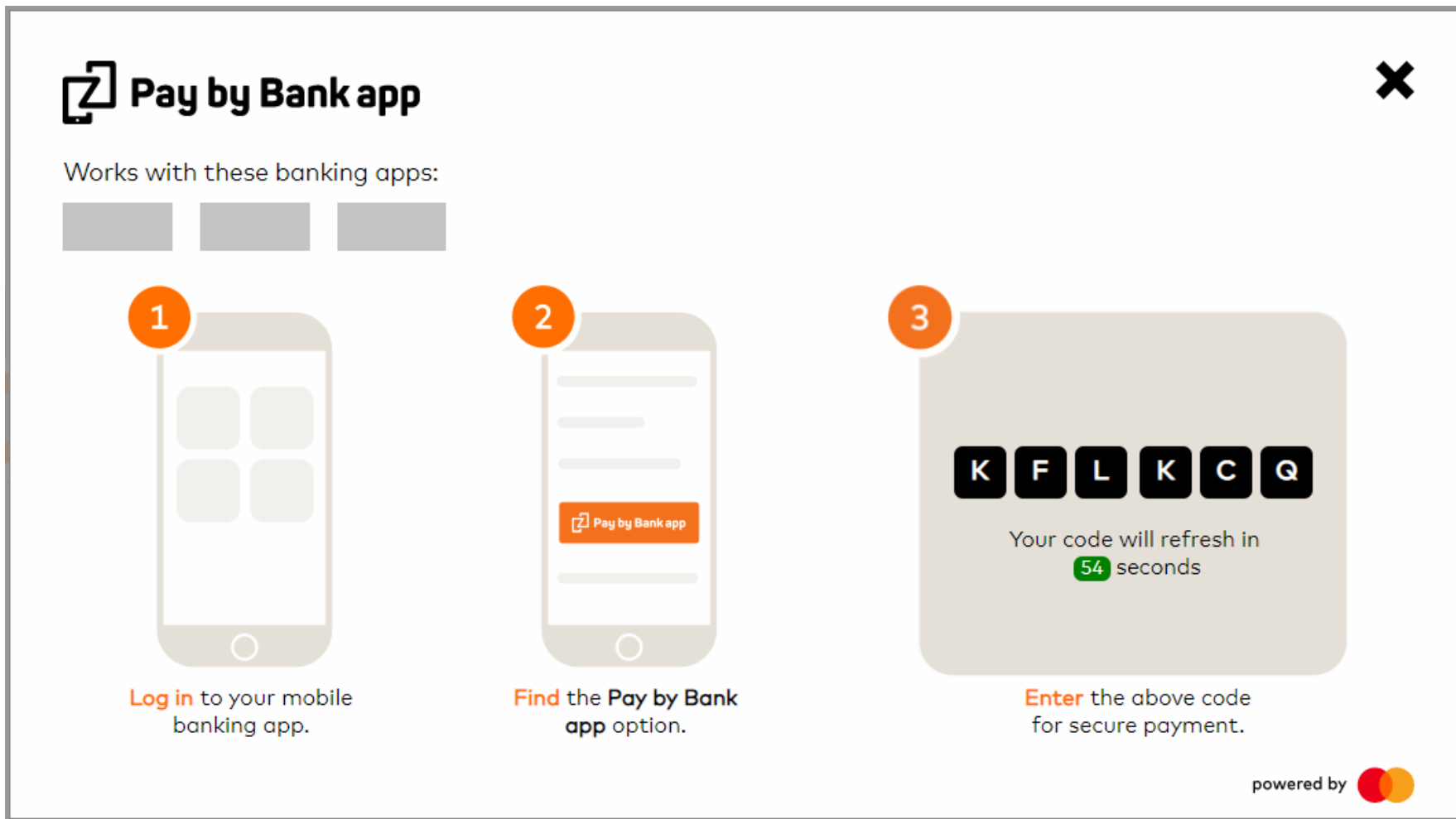


Figure 6: PBBA Pop-up component

### 3.6.2.b. More-About Popup.

The more about popup is displayed when the user clicks on the “More about Pay by Bank app” link. This is an informative popup which contains instructions on how to use the merchant button.

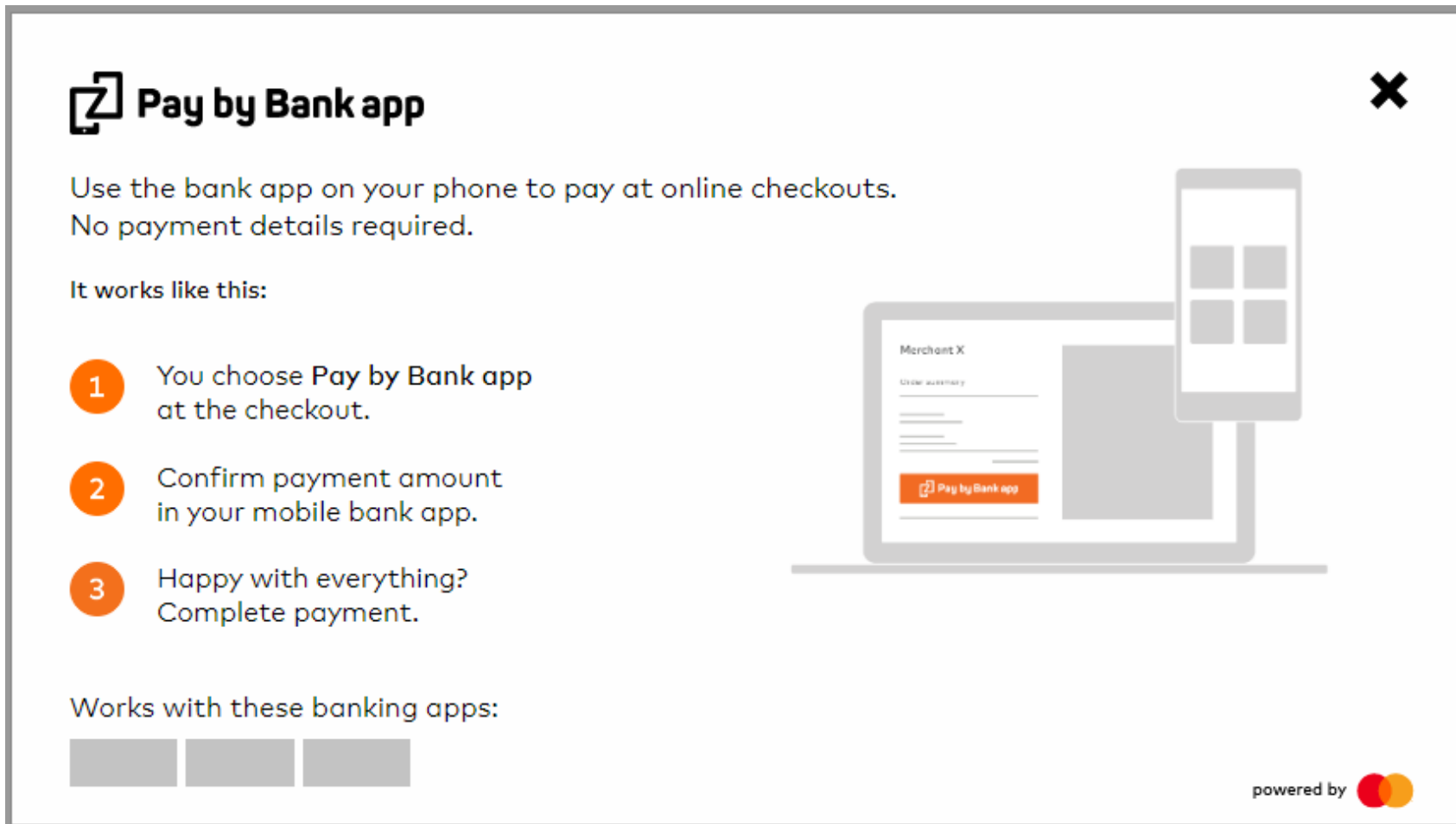


Figure 7: More about popup.

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

### 3.6.3 Cookie management component

This component covers the setting and retrieving of multiple cookies as listed in the table below. The server side code for this component is hosted by `paybybankappcookie.mastercard.co.uk`, but Merchants or Distributors are required to initialise the call as detailed in this document.

Cookie	Type	Party	Set against	Set during	Consumer consent responsibility	Set after consumer consent	Purpose
hasApp	Persistent	Third	paybybankappcookie.mastercard.co.uk	On response to notify	Not available – Basket enhancement		To detect whether installed on mobile
pcid	Persistent	Third	paybybankappcookie.mastercard.co.uk	On response to notify	Zapp	In CFI App	PayConnect Feature (individual tracking)
pcid	Session	First	Merchant Domain	On response to notify	Merchant	In Merchant Website	PayConnect Feature (individual tracking)
testcookie	Session	First	Merchant Domain	Page Load	Merchant	In Merchant Website	Check whether the cookie can be set

Cookie	Type	Party	Set against	Set during	Consumer consent responsibility	Set after consumer consent	Purpose
TPCookieDisabled	Session	First	Merchant Domain	Page Load	Merchant	In Merchant Website	Check whether Third Party cookie enabled

**Table 2: Cookie management component**

The two most important cookies to be noted here are the two persistent cookies – hasApp and pcid

PayConnect Cookie (`pcid`) is used for the PayConnect Journey shown below:

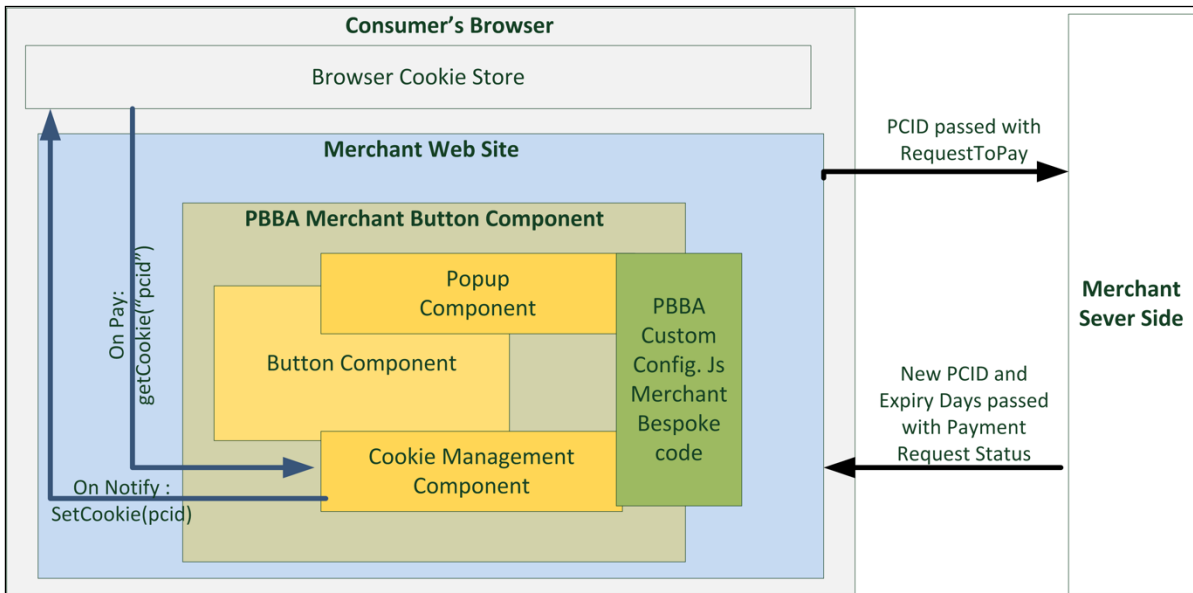


Figure 8: PayConnect Cookie (pcid)

The `hasApp` cookie is used to check if there is a PBBA enabled CFI App within the same device as the browser used for the Merchant Website and is shown below:

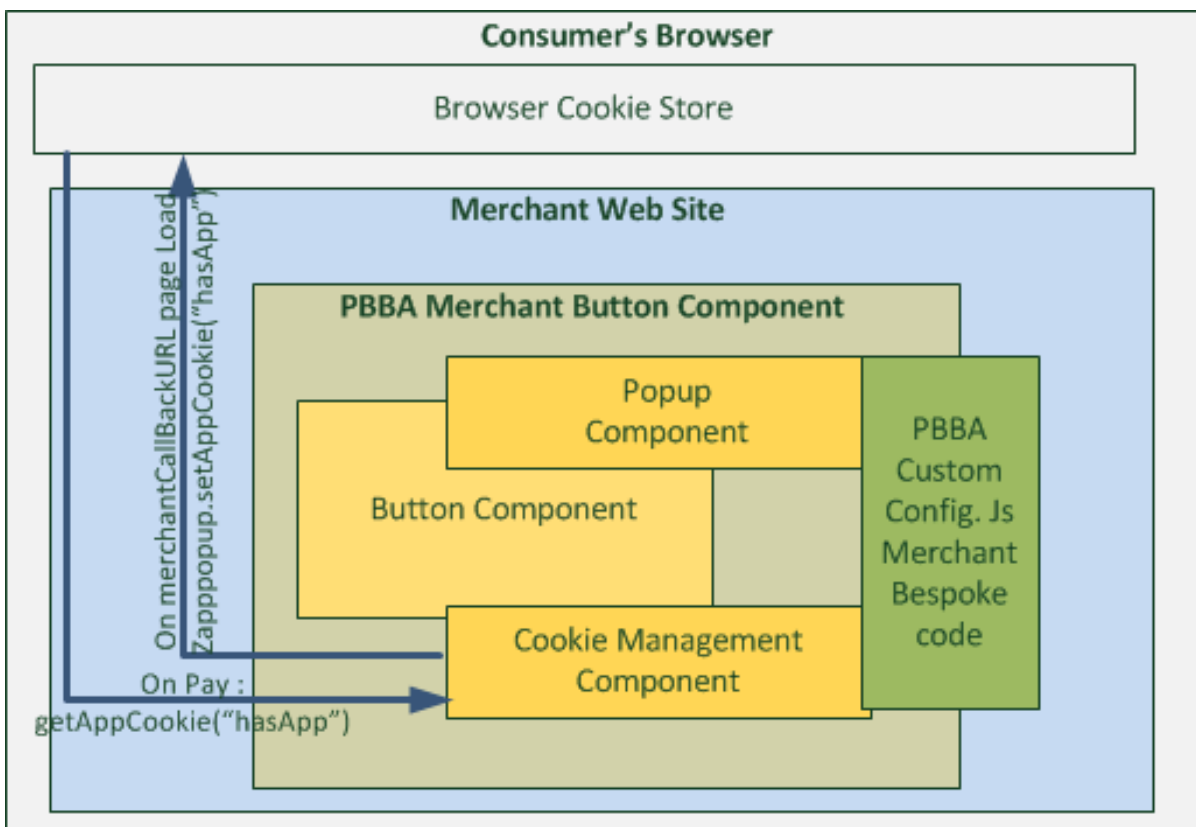


Figure 9: hasApp Cookie

### 3.6.4 Pay by Bank app Branded Merchant Button setup

Download the Merchant Button library from the Web Merchant Button Implementation Guide code folder. The file name is: Web Merchant Button\_x.y.z.zip (see section [3.2 Certified Browsers and Devices](#) for downloading the version compatible with this document). Alternatively, you can also clone the project from GitHub.

Once the library is downloaded, extract the contents of the zip file to a location on your webserver. This location must be accessible via HTTP/HTTPS.

Use the following procedure steps to include the branded Pay by Bank app Button into the Merchant checkout.

**Note** All illustration of Merchant technical component/requirement in all the code snippets and examples below are represented in *italic* text.

#### Procedure steps

1. Import the JavaScript library zapp.js (hosted on the Merchant or Distributor's server) in the parent page where the Button needs to be displayed.

Example:

```
<html>
  <head>
    <script src="http://<Merchant or Distributor web server
URL>/zapp_default/zapp.js"></script>
  </head>
</html>
```

2. The Button could be added to the Website by including the following script in the HTML body.

Example:

```
<script>
  new zapp.button({});
</script>
```

**Note** This will not show the Button immediately.

3. A JavaScript file is needed to initialise several variables used by the Web Merchant Button library. This file is called pbbacustomconfig.js and it resides in the zapp\_default folder along with the zapp.js file.

There are two template files present in the zapp\_default folder:

- pbbacustomconfig\_branded.template

Contains implementation for the Branded Web Merchant Button. You can copy the contents of this file to pbbacustomconfig.js file for Branded Web Merchant Button implementation and then modify it as given below.

- pbbacustomconfig\_custom.template

Contains implementation for the Integrated Web Merchant Button. You can copy the contents of this file to pbbacustomconfig.js file for the Integrated Web Merchant Button implementation and then modify it as detailed in the *PBBA Integrated Web Merchant Button Implementation* document.

The file pbbacustomconfig.js file contains the Branded Button implementation by default.



Open the file named `pbbacustomconfig.js` in the same location as the `zapp.js` file on the Web server and make changes to this file for Merchant specific behaviour. This will make it easy to access and keeps all the Merchant Button files together.

Open the file `pbbacustomconfig.js` in an editor and define the following variables:

```
var zappVersion = "3.1.2";
```

`zappVersion` – This is the Zapp library version. Update this value to point to the library version you have chosen. This variable helps Merchants or Distributors to upgrade/downgrade to different Merchant Button library versions.

```
var cookieManagementUrl =  
"https://paybybankappcookie.mastercard.co.uk/static/cookie-management/pbba-  
3550ce7763041531b9214e9e23986b37/";
```

- `cookieManagementUrl` – This is needed for PayConnect. Normally, the value for this URL will not change. If it does, Zapp will notify any changes to Merchants or Distributors.

```
var cfiLogosURL = "https://paybybankappcdn.mastercard.co.uk/static/ml/pbba-  
3550ce7763041531b9214e9e23986b37/merchant-lib/banks.json"; //cdn location to  
fetch the cfi logos
```

- `cfiLogosUrl` – This is the CDN location for the bank logos that are displayed on the merchant button and the more about popup.

4. To set the PayConnect functionality in the JavaScript file `pbbacustomconfig.js`, type the following

```
window.onload = function() {  
    setupPayConnect(cookieManagementUrl, document);  
}
```

5. Load the Pay by Bank app Web Merchant Button Library.

The Pay by Bank app Web Merchant Button Library must be loaded along with the Consumer defined implementation of the Merchant Button attributes. The Merchant Button has the following attributes which must be implemented in the `pbbacustomconfig.js` file:

- `Pay` – a function which Merchants should overload to send a request to pay and receive a response
- `Notify` – a function which Merchants should overload in order to poll their servers after the request to pay is sent and a response is received
- `Error` – a function which Merchants should overload to handle errors
- `cookieManagementUrl` – set this to the `cookieManagementUrl` variable declared above
- `cfiLogosURL` – set this to the `cfiLogosURL` variable declared above.

The Zapp library is loaded by calling the `zapp.load()` function as given below. This will initialise the Button engine.

```
zapp.load(zappVersion, {
  pay : function(data, callback) {
    // The "data" object above will contain the PayConnectID (pcid) if
    it exists in Consumer's browser
    // callback parameter above is to call the predefined callback function in
    the PBBA button
    // component with response data set
    // See step 5a for mandatory fields to be sent to the Merchant Server
    // See steps 5b to 5d for creating the response data set object,
    // invoking the notify method and error handling
    // respectively.

    },
  notify : function(secureToken, callback) {
    // SecureToken data is part of the request to pay response,
    identifies a request to pay uniquely
    // callback parameter above is to call the predefined callback function in
    the
    // PBBA button component with response data set
    // User defined implementation of polling the Merchant Server
    // See step 5e and 5f for information on this function
    // and setting the PayConnect cookie respectively.

    },
  error : function(errors) {
    // See step 5h for error handling.
  },
  cookieManagementUrl: cookieManagementUrl,
  cfiLogosURL: cfiLogosURL
});
```

### 5a. The **Pay** method

```
pay : function(data, callback)
```

**Note** `data` and `callback` are provided by the Pay by Bank app Web Merchant Button library.

In the Pay method, post the request to pay Merchant data object to the Merchant Server. The only additional data required to be posted from the Consumer's browser is the PayConnectID which can be obtained from the `data` object passed into the function by the Merchant Buttons Cookie Management component.

In the example below, it is assumed that the Merchant's Website is using a `merchantRequestToPayObject` which has a data element by the name `payConnectID` defined by the Merchant.

Example:

```
if (typeof data.pcid !== "undefined")
  merchantRequestToPayObject.payConnectID = data.pcid;
```

The following table provides the mapping of the merchantRequestToPayObject elements mapping to the Distributor's API Element Mapping.

Merchant Request To Pay Element Name	Distributor API Name / Element Name
merchantRequestToPayObject.payConnectID	< Consult Distributor Documentation >

**Table 3: PayConnect ID mapping to Request to Pay**

#### 5b. The Pay method – for Successful Request To Pay Response

Once a successful response is received after posting Merchant request to pay data to the Merchant server, create a response object using the following syntax.

Follow the syntax carefully. If the value for a specific attribute is null then leave it as null:

```
var response = new zappopup.response.payment({
  success : true, // Leave it As is
  secureToken : merchantRequestToPayResponseObject.secureToken,
  brn : merchantRequestToPayResponseObject.pbbaCode,
  retrievalExpiryInterval :
  merchantRequestToPayResponseObject.retrievalTimeOutPeriod,
  confirmationExpiryInterval :
  merchantRequestToPayResponseObject.confirmationTimeoutPeriod,
  notificationSent: merchantRequestToPayResponseObject.cookieSentStatus,
  pcid: null, // Leave it As is
  cfiShortName: merchantRequestToPayResponseObject.bankName
});
```

The following table below provides the mapping of the `merchantRequestToPayResponseObject` elements to your Distributor API Elements.

Merchant Request To Pay Response Element Name	Element Description	Distributor API Name / Element Name
<code>merchantRequestToPayResponseObject.secureToken</code>	Unique token that identified a Request to Pay	< Consult Distributor Documentation >
<code>merchantRequestToPayResponseObject.pbbaCode</code>	A six character code, that identifies a Request to Pay for the duration of retrieval timeout period	< Consult Distributor Documentation >
<code>merchantRequestToPayResponseObject.retrievalTimeOutPeriod</code>	This value specifies the time window from generation of Pay by Bank app code /secure token to the expiry of PBBA code/secureToken, this is used by the get status (Notify method) polling engine	< Consult Distributor Documentation >
<code>merchantRequestToPayResponseObject.confirmationTimeoutPeriod</code>	This is the allowed period of time after the retrieval is complete and before a Payment status is received, the polling continues for the total sum of retrieval and confirmation timeout period	< Consult Distributor Documentation >
<code>merchantRequestToPayResponseObject.cookieSentStatus</code>	This field is used in the PayConnect journey only, the field confirms if a payment notification was sent out to the consumer, the Popup component of the button shows the appropriate Popup based on this flag	< Consult Distributor Documentation >

Merchant Request To Pay Response Element Name	Element Description	Distributor API Name / Element Name
merchantRequestToPayResponseObject.bankName	This field is used in the PayConnect Journey only, the Popup when displays that a push notification is sent out, it also displays the CFI name.	< Consult Distributor Documentation >

**Table 4: Response to Request to Pay Mapping**

After the response object is implemented, make a call-back to start the notification process:

```
callback(response);
```

### 5c. The Pay method – for failed Request To Pay response

If the request to pay response comes back as failed, use the following call-back to notify the Pay by Bank app Web Merchant Button Library of an error: By doing so the Notify method will show a standard error message and no polling for payment status is triggered too.

**Note** The `data` object expects a JSON object from Merchant to be mapped, the sole purpose of this `data` object is to show error messages/related information in the browser console.

```
callback(new zappopup.response.payment({
  success : false, // Leave As is
  data : MerchantErrorJSONObject
}));
```

### 5d. The **Notify** method

```
notify : function(secureToken, callback) {}
```

**Note** `secureToken` and `callback` are provided by the PBBA Web Merchant Button library.

The Merchant library will invoke the notify method every X seconds where X is a configurable property called `merchantPollInterval` in the PBBACustomConfig.js.

The property `merchantPollInterval` allows the Merchant to set the poll interval for the Notify method. The default value of this property is 5000 milliseconds.

For example, to change the property to 10000 milliseconds, go to PBBACustomConfig.js file and change the value of `merchantPollInterval`.

```
var merchantPollInterval = 10000; // 10 seconds
```

The Popup continues to display as long as the polling is on. The polling is controlled by a success flag set based on Payment Status.

Setting this flag to true will stop the polling and remove the Popup. Setting the success flag to false continues the polling.

The following steps explains this in more detail.

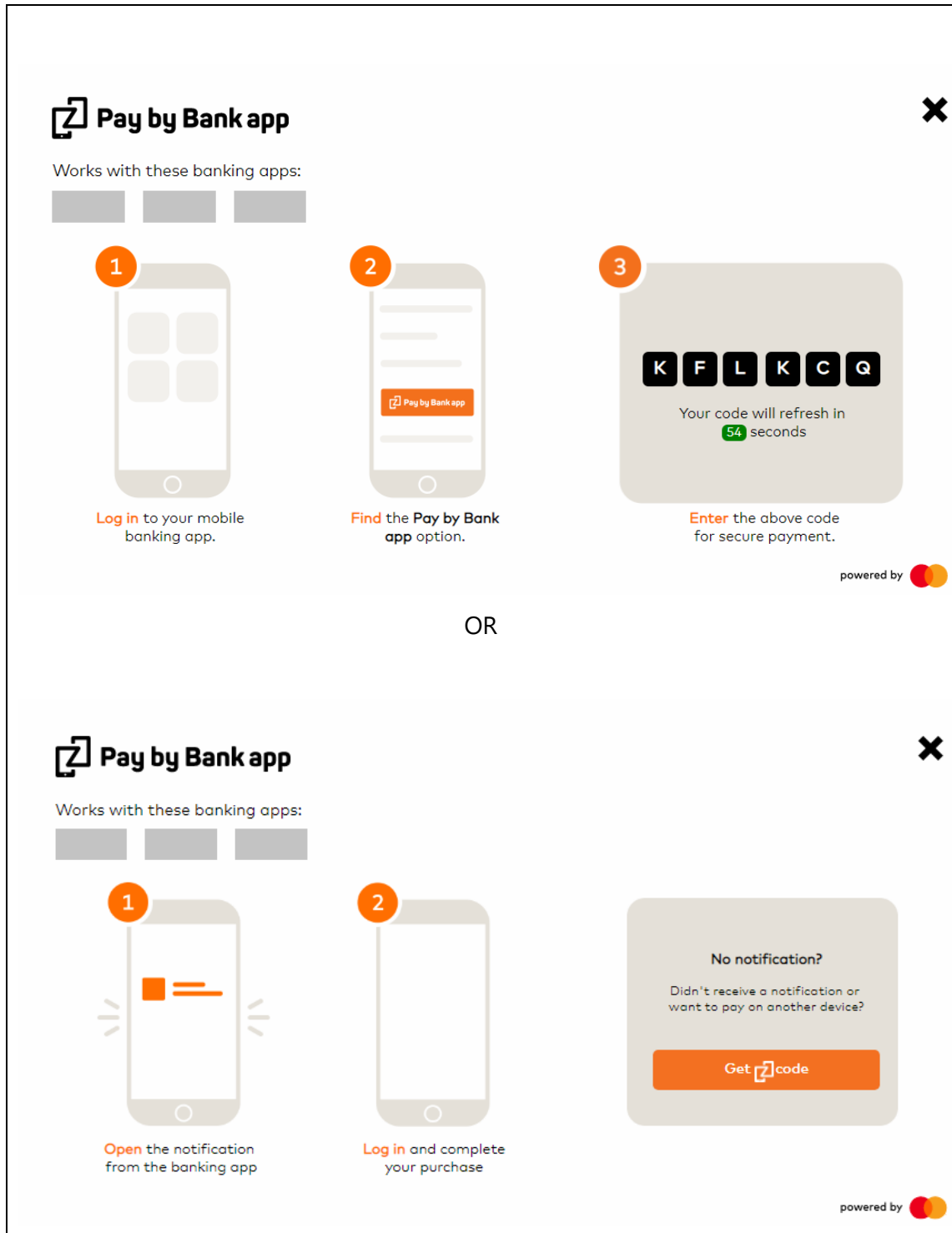
See Appendix [A.1.1 Merchant Poll Intervals](#) for more information on polling.

## 5e. The Notify method – Payment Status INPROGRESS

If the payment status response indicates that the Transaction is INPROGRESS, then create a response object with a success flag set to false and invoke the `callback` function:

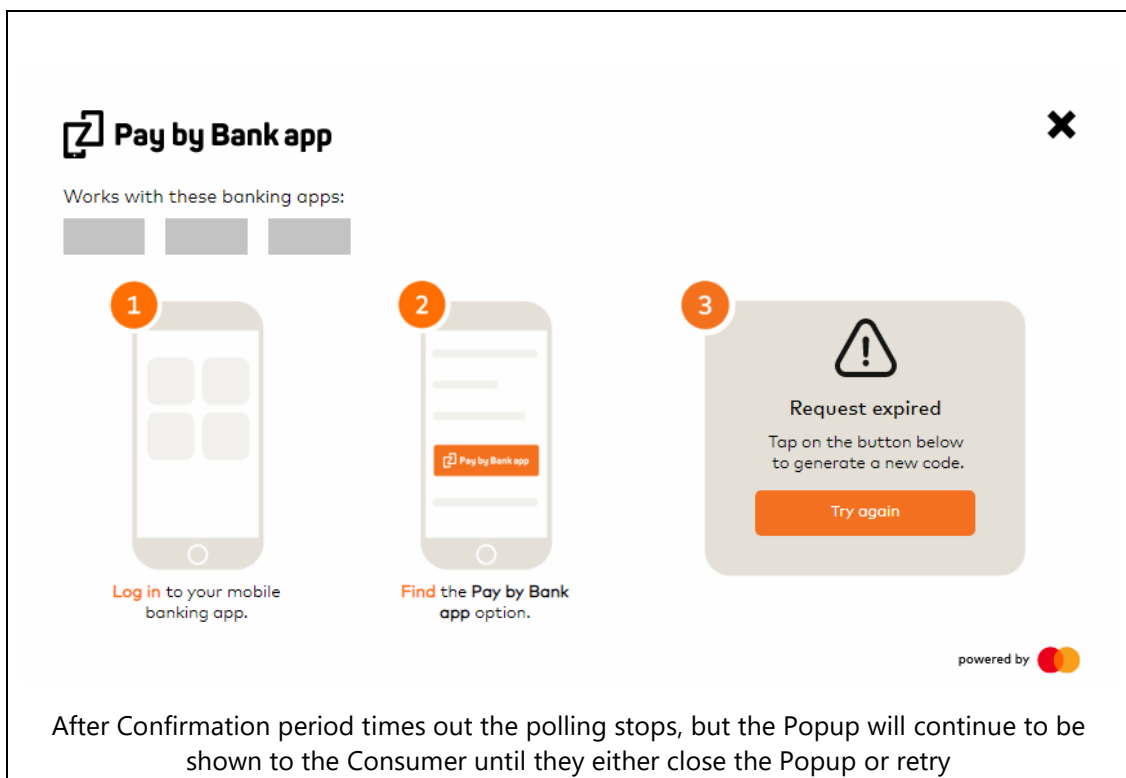
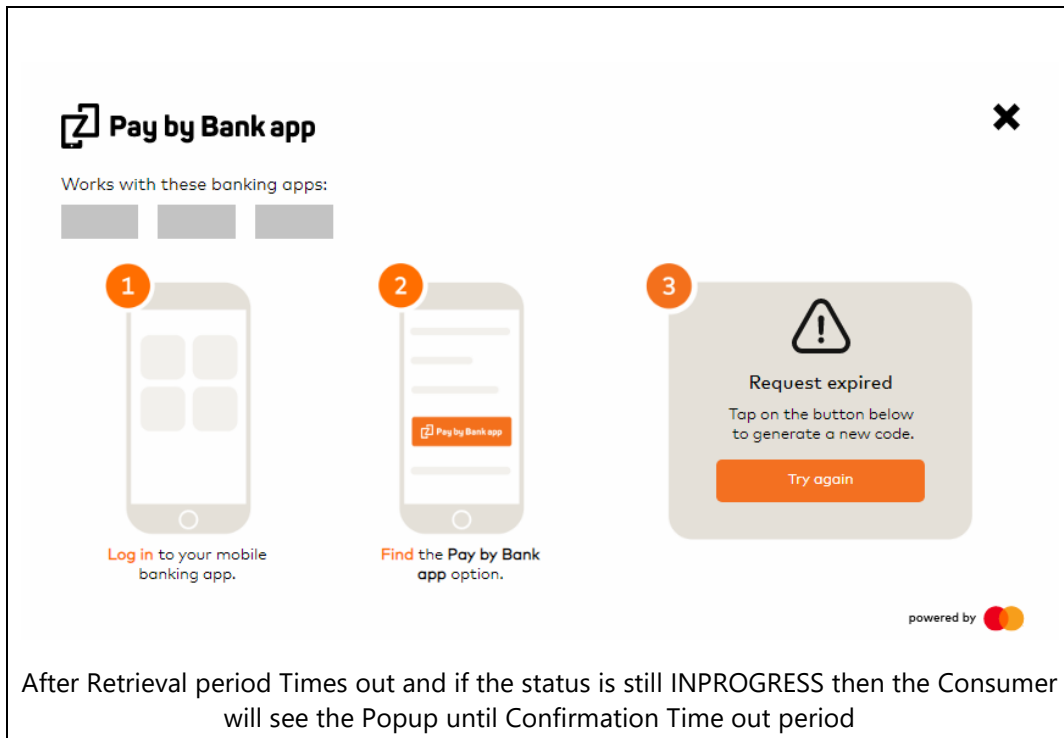
```
var response = new zappopup.response.notify({success : false});
```

```
callback(response);
```



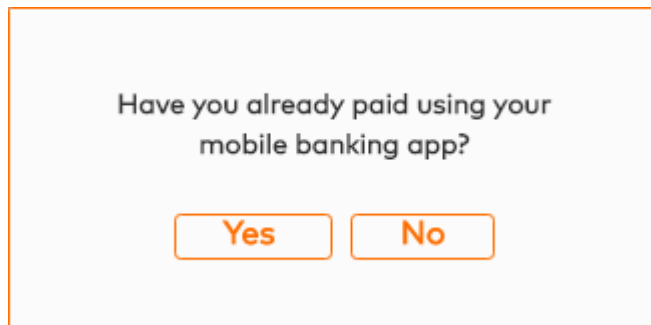
INPROGRESS status keeps the Polling continue until the Retrieval Timeout Period is reached, the below screen will continue to show during this period





- If the Consumer tries to close the Popup while polling is in progress then the following confirmation box will be displayed.

**Note** As outlined above in relation to the M-COMM journey, 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.



- Selecting **No** will stop the polling process and close the Popup, thereby ending the Transaction.
- Selecting **Yes** will close the confirmation box and continue polling.

#### 5f. The Notify method – Payment Status AUTHORISED

If the payment status response indicates that the Transaction is AUTHORISED by the Consumer, then create a response object with a success flag set to true and invoke the callback function.

- Create a response object with success flag set to true and invoke the callback function:

```
var response = new zapppopup.response.notify({success : true});  
callback(response);
```

- Check for the PayConnect ID in the response data. If present, call the following function, ensuring the first argument key `pcid` remains unaltered:

```
setCookie("pcid", merchantGetPaymentStatusObject.payConnectID,  
          merchantGetPaymentStatusObject.cookieExpiryDays,  
          cookieManagementUrl);
```

Merchant Request To Pay Response Element Name	Element Description	Distributor API Name / Element Name
<code>merchantGetPaymentStatusObject.payConnectID</code>	This element is for PayConnect Journey, if the Consumer has opted for PayConnect, then this ID will be passed back in the payment status response and should be set into the browser	< Consult Distributor Documentation >
<code>merchantGetPaymentStatusObject.cookieExpiryDays</code>	This element defines the number of days the above PayConnectID based cookies is valid for	< Consult Distributor Documentation >

**Table 5:** PayConnect cookie setting for Pay Status Authorised

- Merchant's custom success page handling should follow after setting the PayConnect cookie.

See Appendix [A.3 Additional Cookie management Information](#) for more information on how the cookie management function works.

### 5g. The Notify method – Payment Status DECLINED

If the payment status response indicates that the Transaction is DECLINED by the Consumer or error by the system, then create a response object with a success flag set to true and invoke the callback function.

- Create a response object with a success flag set to true and invoke the callback function:

```
var response = new zapppopup.response.notify({success : true});
callback(response);
```

- Merchant's custom failure page handling should follow the above function call.

### 5h. Implement your custom error handling mechanism

```
error : function(errors) {}
```

Implement the error handling mechanism here.

After implementing the above steps, import the pbbacustomconfig.js file to the parent html page after the zapp.js file, reload the page and the Pay by Bank app Button is ready to be used.

Example:

```
<html>
  <head>
    <script src="http://<Merchant or Distributor web server
URL>/zapp_default/zapp.js"></script>
    <script src="http://<Merchant or Distributor web server
URL>/zapp_default/pbbacustomconfig.js"></script>
  </head>
</html>
```

**Note** This Button has a height set to 100% which means that if the script above is the only script residing in the HTML body, then it will take up the entire height. Zapp recommends that you wrap this Button within a <DIV> tag to control the height and width of this Button.

### 3.6.5 Pay by Bank app Branded Merchant Button sample code

See Appendix [A.2 Pay by Bank app Button implementation sample code](#) for a full sample code set for reference.

**Important** All Merchant data objects in the sample code are assumed to be JSON objects.

## 4 Additional considerations for M-COMM

The Web Merchant Button Library is optimised also to work on the mobile devices listed in the section Certified Browsers and Devices.

The following sections illustrate the minor adjustments to the Pay by Bank app Merchant Button to get a full M-COMM experience.

### 4.1 Prerequisites for M-COMM

The parent Website/page must be optimised for mobile devices. This means that they should have a responsive UI. This can be completed easily by including the following meta tag:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

### 4.2 Mobile App Cookie – retaining PBBA enabled Bank App selection

A mobile App cookie by the name `hasApp` is set on the mobile browser once the Consumer clicks on the `Open mobile banking app` button and completes the payment journey. This cookie helps the Pay by Bank app Web Merchant Button to remember the decision and open the Pay by Bank app enabled CFI App the next time a Consumer chooses Pay by Bank app as the payment method, instead of providing a Popup with the option to open the app.

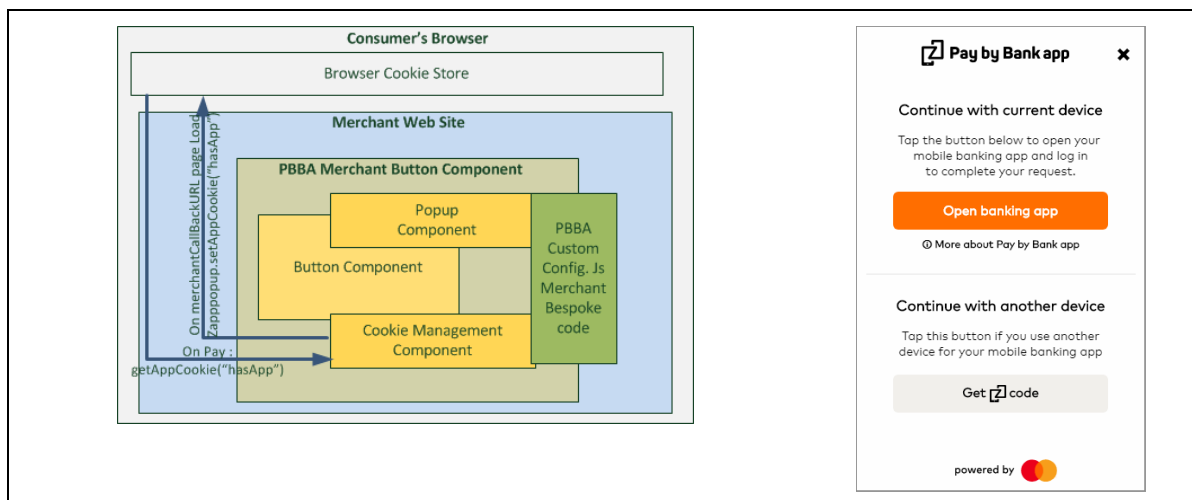


Figure 10: Mobile App Cookie

#### 4.2.1 Setting hasApp cookie

In the case of a Branded Web Merchant Button, follow these steps on the Web page that was provided by the Merchant as a MerchantCallbackURL, in the Request To Pay API. The MerchantCallbackURL is used to transfer the control from a Pay by Bank app enabled CFI App back to the Merchant Website in M-COMM Journey (Single Device) after the Consumer has either Confirmed or Declined the payment.

1. Import zpopup-extra.js file to the Merchant call back URL Page.

2. Call the function `zapppopup.setAppCookie(cookieManagementUrl)`, where the value of `cookieManagementUrl` is the same one that was set in the `pbbacustomfconfig.js` file stated in the above sections.

**Important** Wait for a minimum of 500 milliseconds after calling this function as it requires setting a cookie using an IFRAME.

The Consumer experience of the M-COMM Journey, when clicked for the first time with Pay by Bank app Web Merchant Button on a mobile browser, is illustrated in Figure 11 and Figure 12 following.

Consumer selects Pay by Bank app on the same device:

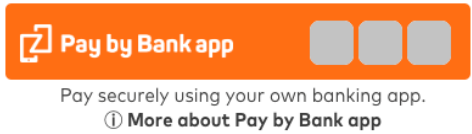
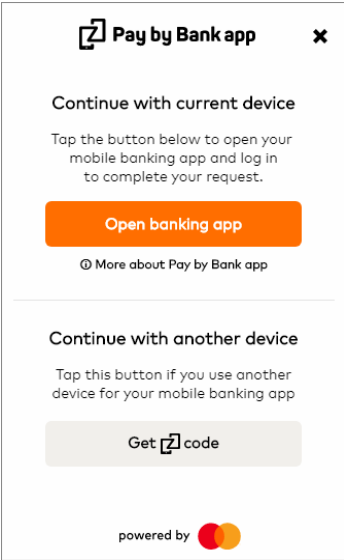
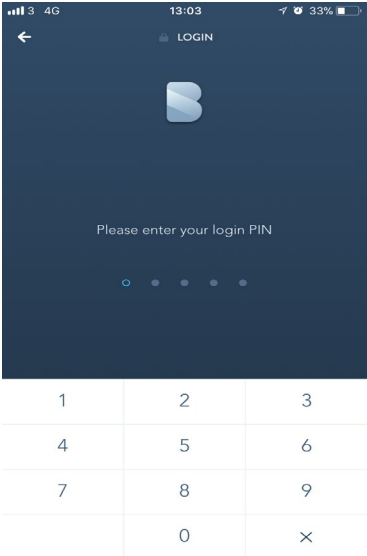
<p>1. Consumer starts a Pay by Bank app Journey for the first time on a Browser</p>	<p>2. Consumer is provided with a selection, as shown below</p>	<p>3. The PBBA enabled CFI bank App on the same device is invoked automatically</p>
		
<p>Consumer clicks on the PBBA Branded Web Merchant Button to make a payment</p>	<p>Consumer selects Pay by Bank app and clicks 'Open banking app'</p>	<p>On Consumer completion of the payment confirmation decision and when the control from CFI App gets returned to the Merchant Webpage, the 'hasApp' cookie is set on this browser for future PBBA Journey from the browser</p>

Figure 11: M-COMM Journey on a mobile browser

Consumer selects Pay by Bank app but uses another device to open the Pay by Bank app Enabled CFI App:

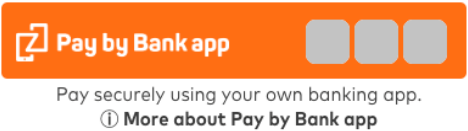
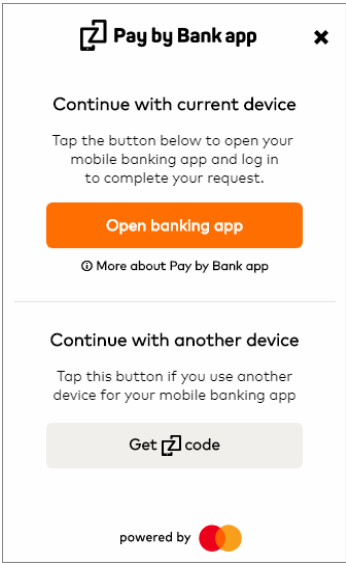
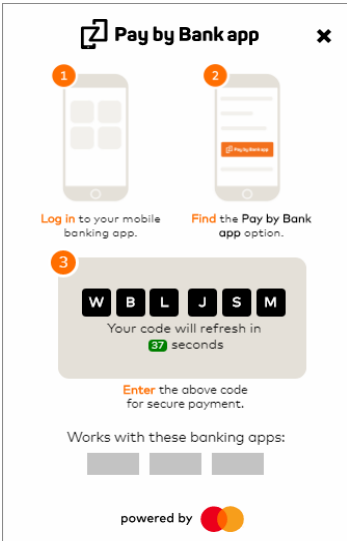
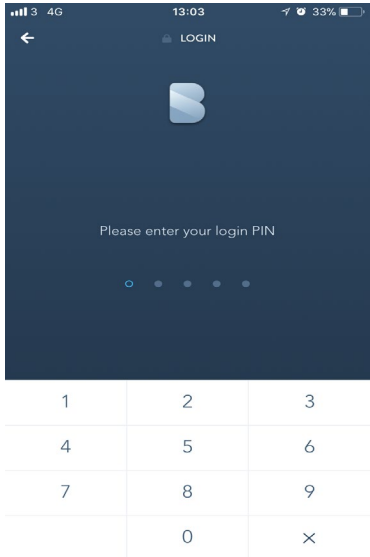
<p>1. Consumer starts a PBBA journey for the first time on a Browser</p>	<p>2. Consumer is provided with a selection, as shown below</p>	<p>3. Consumer selects 'Get PBBA code' in the 'Continue with another device' section of the pop up</p>	<p>4. Consumer completes the journey on another device with a PBBA Enabled CFI App</p>
			
<p>Consumer clicks on the PBBA Branded Web Merchant Button to make a payment</p>	<p>Consumer selects 'Get PBBA code' in the 'Continue with another device' section of the pop up</p>	<p>Consumer enters the PBBA code into the PBBA enabled CFI App on another device</p>	<p>The 'hasApp' cookie is not set</p>

Figure 12: M-COMM journey with another device



### 4.3 Integrated Web Merchant Button

An Integrated Web Merchant Button has been developed for Merchants who require the use of radio buttons or drop down menus at the checkout.

This document covers the implementation of the Pay by Bank app Branded Web Merchant Button only. For details on how to implement the Pay by Bank app Integrated Web Merchant Button with PBBA Popup please refer the *PBBA Integrated Web Merchant Button Implementation* document.

## A Appendices

---

### A.1 Merchant Configurable Properties for the PBBA Branded Web Merchant Button

This section describes the available configurable properties and how to initialise these properties for the Merchant Button in `pbbacustomconfig.js` file.

#### A.1.1 Merchant Poll Intervals

The property `merchantPollInterval` allows the Merchant to set the poll interval for the Notify method.

This default value of this property is 5000 milliseconds. In order to override this property, declare a variable named `merchantPollInterval` in `pbbacustomconfig.js` file and set the interval in milliseconds. Pass this variable to the `zapp.load` function as mentioned below:

```
var merchantPollInterval = 10000; // 10 seconds
```

#### A.1.2 Pay by Bank app Cookie Management Component

The PBBA Popup sets multiple cookies to provide a rich user experience. Some are Merchant domain specific cookies and others are `paybybankapp.co.uk` domain specific third party cookies. There are persistent cookies and session cookies which are detailed in Table 2 [Cookie management component](#).

Some browsers (like Safari) do not support third party cookies. In order to set third party cookies on such browsers, one must visit the third party website on the browser first.

The PayConnect feature relies on setting the PayConnect cookie (called `pcid`) on the browser. The `setCookie()` function, when invoked, first detects if the third party cookies are enabled on the browser. If the third party cookies are not enabled then the `setCookie()` function redirects to the cookie management URL and lands back on the merchant page from where the `setCookie()` function was invoked. This means that the merchants will have to hold the data in the session to be displayed when the page is reloaded after the redirect.

If the third party cookies are enabled then there will be no redirect

There are two ways to set the PayConnect cookie post receipt of a successful response:

1. From within `pbbacustomconfig.js` after the payment notification comes through; or
2. After the display of the successful page to the Consumer. If the `setCookie()` function is invoked after the order success page, then the following steps need to be carried out:
  - Import the following files in the page where `setCookie()` needs to be invoked from in the order they appear:
    - `jquery-3.4.1.min.js`
    - `zapp.js`
    - `cookie-management.js`
  - Add the following javascript to the page::

```
window.onload = function() {  
  setupPayConnect(cookieManagementUrl, document);  
}
```

`cookieManagementUrl`: This is the value of the cookie management URL which was set in `pbbacustomconfig`.

`js document`: This is the document property of the window

- Invoke the `setCookie()` function with the following syntax:

```
setCookie("pcid", merchantGetPaymentStatusObject.payConnectID,  
         merchantGetPaymentStatusObject.cookieExpiryDays, cookieManagementUrl);
```

Merchant Request To Pay Response Element Name	Element Description	Distributor API Name / Element Name
<i>merchantGetPaymentStatusObject.payConnectID</i>	This element is for PayConnect Journey, if the Consumer has opted to PayConnect , then this ID will be passed back in the payment status response and should be set into the browser	< Consult Distributor Documentation >
<i>merchantGetPaymentStatusObject.cookieExpiryDays</i>	This element defines the number of days the above PayConnectID based cookies is valid for	< Consult Distributor Documentation >

As stated previously, since there is an element of page redirect in case of browsers like Safari, the Merchant has to choose one of the above options based on the ease of returning to the same page/final state.

## A.2 Pay by Bank app Button implementation sample code

The example below is a sample pbbacustomconfig.js file depicting implementation of the pay and notify methods.

**Note** Any data elements and comments in *Italic* are a Merchant specific data element which must be provided by the Merchant.

```
-----START-----
jQuery.support.cors = true;
if (!window.console) console = {log: function() {}};

var zappVersion = "3.1.2"; // This is the current web merchant button library version. Packaged with button releases
var cookieManagementUrl = "https://paybybankappcookie.mastercard.co.uk/static/cookie-management/pbba-3550ce7763041531b9214e9e23986b37/"; // Cookie Management, packaged with Button releases.
var merchantPollInterval = 5000; // Default 5 Seconds. Could be overridden by merchants Seconds
var cfiLogosURL = "https://paybybankappcdn.mastercard.co.uk/static/ml/pbba-3550ce7763041531b9214e9e23986b37/merchant-lib/banks.json";// CDN location for CFI logos

// Initialize Payconnect.
window.onload = function() {
    setupPayConnect(cookieManagementUrl, document);
}

zapp.load(zappVersion, {
    pay : function(data, callback) {

        var _data = data;

        var MerchantRequestToPayPostData = {
            payConnectID: null
        };

        if (typeof data.pcid !== "undefined")
            MerchantRequestToPayPostData.payConnectID = data.pcid;
    }
});
```

```

jQuery.ajax({
url : "MerchantRequestToPayPostOrder", //The Merchant URL to post the Request To Pay
  type : "POST", // Merchant specific setting for HttpGet or HttpPost
  crossDomain : true,
  data : MerchantRequestToPayPostData, // The MerchantRequestToPay JSON Object
  headers : {
    "accept" : "application/json; charset=UTF-8"
  },
  success : function(merchantRequestToPayResponseObject) {
    var response = new zappopup.response.payment
    (
      success : true,
secureToken : merchantRequestToPayResponseObject.secureToken,
brn : merchantRequestToPayResponseObject.pbbaCode,
retrievalExpiryInterval : merchantRequestToPayResponseObject.retrievalTimeOutPeriod,
confirmationExpiryInterval : merchantRequestToPayResponseObject.confirmationTimeoutPeriod,
notificationSent: merchantRequestToPayResponseObject.cookieSentStatus,
pcid: null, // Leave it As is
cfiShortName: merchantRequestToPayResponseObject.bankName
    );
    callback(response);
  },
  error : function(merchantRequestToPayResponseObject) {
    callback(new zappopup.response.payment({
      success : false,
      data : merchantRequestToPayResponseObject
    }));
  }
});
},
notify : function(secureToken, callback) {

  var _callback = callback;

  var _confirmOrder = function(data) {

```

```
var _orderData = data;

jQuery.ajax({

    url: "/merchantsuccessOrDeclinepage.<something>", //this is a Merchant backend server call
    type: "POST", //Merchant specific http method get or post
    crossDomain: true,
    contentType : "application/json; charset=UTF-8",

    data : {
        jsonArray : JSON.stringify(_orderData)
    },
    success : function(merchantGetPaymentStatusObject) {
        merchantres = JSON.parse(merchantGetPaymentStatusObject);

        if (merchantres.status === 'success') {
            setTimeout(function() {
                window.location = merchantres.html;
            },1000);
            if (typeof merchantres.PayConnectID !== "undefined") {
setCookie("pcid", merchantres. PayConnectID, merchantres.cookieExpiryDays, cookieManagementUrl);
            }

            } else {
                window.location = merchantres.html;
            }
        }
    });
};

var nothing = null;

jQuery.ajax({
url : "/getstatus/merchantgetstatuscall.<something>", //this is Merchant backend server call
    type: "get", //Merchant specific http method get or post
    crossDomain: true,
    cache: false,
```

```
        contentType : "application/json; charset=UTF-8",
        success : function(merchantGetPaymentStatusObject) {
            if (typeof merchantGetPaymentStatusObject.errorCode === "undefined") {
                merchantGetPaymentStatusObject.success = true;
            }
            var response = new zappopup.response.notify(merchantGetPaymentStatusObject);
            _callback(response);
            _confirmOrder(merchantGetPaymentStatusObject);
        },
        error : function(merchantGetPaymentStatusObject) {
            console.log('error');
            merchantGetPaymentStatusObject.success = false;
            var response = new zappopup.response.notify(merchantGetPaymentStatusObject);
            _callback(response);
        }
    });
},
error : function(errors) {
    console.log(errors);
    alert(errors);
},
cookieManagementUrl: cookieManagementUrl,
merchantPollInterval: merchantPollInterval,
cfiLogosURL : cfiLogosURL
});
```

-----FINISH-----

Sample Div tag for button sizing

Div Tag



```
<div class="zapp-button-div">  
<script>  
  new zapp.button({"productId" : "1" });  
</script>  
</div>
```

### CSS file entry

```
.zapp-button-div {width: 390px;}
```

**Note** Wrap this Button within a <DIV> tag to control the size of the Button. The width of this <DIV> is to be defined in px and the library will render the button of the same width and of proportionate height. Zapp recommends setting the width of the wrapping <DIV> tag to 390px.

## A.3 Additional Cookie management Information

In addition to the cookie features controlled by the Merchant button and the data passed via the gateway interface, the Pay by Bank app cookies has two other property controls.

### A.3.1 Remove all connections to the Mobile Banking Application Consumer function

Consumers have an option within the Mobile Banking application to cancel all connections to allow them to deactivate cookie tokens for Pay by Bank app. This service can be used when either the Consumer no longer wishes to receive push notifications or where one or more of the Consumer's devices may have been compromised.

The cookies will remain active on the Browser. However, when processed by Pay by Bank app as part of a submit RTP, the response will be returned as invalid. The Consumer will be re-offered the opportunity to connect as part of the payment confirmation journey.

### A.3.2 DDoS protection

To prevent a potential Pay by Bank app DDoS threat, an individual cookie token can only be submitted ten times without the Consumer authorising the payment request.

When the cookie token is submitted and successfully retrieved by the Consumer to authorise a payment it is refreshed with a new cookie token which is returned to the Merchant as part of the payment notification and used to update the Pay by Bank app third party browser cookie.

If the same cookie token is submitted ten times in each occasion either:

- the order is not retrieved by the Consumer, or
- the order is retrieved by the Consumer and the Transaction is declined by the Consumer

The cookie token is invalidated within Pay by Bank app system. On next submission of the cookie token, the response is invalid cookie token and Pay by Bank app code is displayed within the Web Merchant Button pop up as the only available retrieval method.

## A.4 Known browser specific requirements

This section lists any additional consideration that has to be given, to some browsers, from the list of supported browsers (see section [3.2 Certified Browsers and Devices](#)) to implement this Pay by Bank app Branded Web Merchant Button.

### A.4.1 Internet Explorer

Internet Explorer's default behaviour is to cache AJAX GET requests. If the Merchant uses jQuery AJAX for polling in the notify method using GET, then IE may cache the polling requests if the polling request parameter remains the same for every poll.

In order to circumvent this behaviour of IE, there are four options identified by the Zapp development community as detailed below:

1. jQuery Fix:

The pbbacustomconfig.js is shipped with this change, Merchant can benefit from this option if they have chosen to set a jQuery based Ajax call. The change is as below

```
$.ajax({  
  ...  
  cache: false,  
  ...  
});
```

## 2. URL Cache Buster

A parameter with a random number can be added to the polling request URL. This will invalidate the IE cache, because for every poll the value of the URL cache busting parameter will be different.

Example:

If the polling URL is:

```
`https://www.merchantdomain.com/responseForPaymentStatus.aspx?secureToken=1234  
&orderId=5678`
```

Then, after adding a cache buster parameter with date timestamp (called `datetime` as shown in the example below), the URL would look like:

```
`https://www.merchantdomain.com/responseForPaymentStatus.aspx?secureToken=1234  
&orderId=5678`+ `&datetime='+Date.now()
```

## 3. Change Http GET to Http POST:

Changing the AJAX method from GET to POST prevents IE from caching the AJAX requests.

## 4. Web server no-cache header

Make changes to webserver setting to send no-cache header in response to the polling request.

## Recommendations

- Option 1 is already implemented within the Web Merchant Button Library for JQuery based Ajax calls
- Zapp recommends that Merchants also implement Option 2 as this will ensure a foolproof solution to the immediate IE issue and any future browsers that could adopt a similar caching policy as IE.
- Option 3 is not a recommended option by Zapp as HttpPost is against Web design principles for GET status style requests and it also has performance implications
- Option 4 is a server side option which will require Merchant infrastructure design level consultation. Full testing is required to ensure that the changes just impact this URL/ URI and not any other area that requires caching.

## A.5 Polling for Payment Status

Whilst the Pay by Bank app Branded Web Merchant button has in-built polling capabilities to determine current status of the payment, the Merchant must also use their Distributor's query or status request APIs to determine payment status for scenarios that cannot be covered by the Pay by Bank app Branded Web Merchant button, for example, if the Consumer closes the browser.

## A.6 Upgrading the library

Follow the steps below to upgrade the Web Merchant Button Library.

1. Back up the existing library (the zapp\_default folder)
2. Download the latest version of the library from the URLs mentioned in **Error! Reference source not found.** of section 3.2 [Certified Browsers and Devices](#).
3. Replace the contents of the existing zapp\_default folder with the contents of the zapp\_default folder from the newly downloaded library
4. Merge (don't replace) your changes from the backed up pbbacustomconfig.js file in Step 1 above into the pbbacustomconfig.js file of the new library

## A.7 Handling Consumer edge cases

The following represent edge cases with respect to the Consumer's use of the Pay by Bank app Web Merchant Button.

### A.7.1 Consumer selects Pay by Bank app more than once per order

The Consumer selects Pay by Bank app but then does not go to their Mobile Banking App to complete the payment, for example they close the Pay by Bank app pop up. The Consumer could do this multiple times resulting in multiple Request to Pays per Merchant order. In this scenario, the Request to Pay remains active until the Retrieval timeout period passes.

The implication is that the Merchant needs to understand the status of each Request to Pay to ensure the Order can be fulfilled on receipt of payment. The Merchant can do this in one of the following ways:

- Via the polling within the Merchant Button Library.
- Upon receipt of a payment notification from the Merchant's Distributor. The method of notification may differ for each Distributor.
- By using the Distributor's query API – see Appendix [A.5 Polling for Payment Status](#) for more information on polling for payment status.

### A.7.2 Consumer retrieves payment in Mobile Banking app but pays with another payment method.

The Consumer selects Pay by Bank app, retrieves the payment within the Mobile Banking App but then returns to the Merchant website and pays with a different payment method.

The implication is that the Merchant needs to understand the payment status of the Consumer's order across all available payment methods to ensure the Consumer does not

pay twice. In addition to the methods described in Appendix [A.7.1 Consumer selects Pay by Bank app more than once per order](#), the Merchant also has the Refund capability available as a fall back to recover double payments.

### **A.7.3 Consumer starts the payment journey in a non-default browser but is returned to the Merchant website with the device default browser post payment.**

The Consumer opens a non-default browser, shops on a merchant site using the browser, selects Pay by Bank app, retrieves the payment within the Mobile Banking App but then is returned to the default browser.

This happens due to the Merchant return URL taking the consumer from the Mobile Banking App to the default browser. The implication is that the Merchant needs to understand and handle this edge case. There are several options that can be taken –

- As the Merchant return URL contains the SecureToken, the Merchant can choose to display the status of the order in the default browser despite the Consumer not being logged in.
- If the Merchant can correctly detect the non-default browser and articulates invocation intent as part of the Merchant return URL which correctly maps to the non-default browser, then this edge can be avoided. Devising such a URL would need to consider all possible browser combinations across both Android and iOS.

## **A.8 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 or Android Merchant Button. Pay by Bank app does not mandate that the native libraries must be used for Hybrid apps.

### **A.8.1 Hybrid iOS Apps**

iOS web view (WKWebView) doesn't handle the custom URL schemes automatically.

There are two ways to enable custom URL scheme handling from web content inside an iOS app:

1. Manually handle custom URL scheme invocations for WKWebView using the delegate interface.
2. Switch to Safari View Controller (SFSafariViewController) instead of WKWebView.

#### **A.8.1.1 Manually handle the custom URL scheme invocations for WKWebView**

One delegate method should be implemented

`webView:decidePolicyForNavigationAction:decisionHandler:` which will intercept the custom URL scheme and will invoke it manually.

Objective-C sample code which handles "zapp" custom scheme:

```
#import <WebKit/WebKit.h>
#import "WebViewDelegate.h"
```

```

@interface WebViewDelegate () <WKNavigationDelegate>

@end

@implementation WebViewDelegate

//
// Objective-C example of how to handle custom scheme invocatins
//
- (void)webView:(WKWebView *)webView
decidePolicyForNavigationAction:(WKNavigationAction *)navigationAction
decisionHandler:(void (^)(WKNavigationActionPolicy))decisionHandler
{
    NSURL *invocationURL = navigationAction.request.URL;
    if ([invocationURL.scheme isEqualToString:@"zapp"]) {
        [[UIApplication sharedApplication] openURL:invocationURL];
        decisionHandler(WKNavigationActionPolicyCancel);
        return;
    }

    decisionHandler(WKNavigationActionPolicyAllow);
}

@end

```

### A.8.1.2 Switch to Safari View Controller (SFSafariViewController)

This requires the minimum OS target to be iOS 9. If the app developer switches to Safari View Controller instead of WKWebView then the Safari View Controller will invoke the custom URL schemes automatically.

## A.8.2 Hybrid Android Apps

Android web view (WebViewClient) doesn't handle the custom URL schemes automatically.

There are two ways to enable custom URL scheme handling from web content inside an iOS app:

1. Override the shouldOverrideURLLoading method for WebViewClient
2. Switch to Chrome View Client (WebChromeClient) instead of WebViewClient.

### A.8.2.1 Override the shouldOverrideUrlLoading method for WebViewClient

Overriding shouldOverrideUrlLoading for WebViewClient tells the client to only handle http or https scheme URLs and to pass every other scheme back to the Android OS to handle.

Sample code which overrides shouldOverrideUrlLoading:

```

@Override
public boolean shouldOverrideUrlLoading(WebView view, WebResourceRequest request) {
    String url = request.getUrl().toString();
    if( url.startsWith("http:") || url.startsWith("https:") )

    { return false; }
    // Otherwise allow the OS to handle it
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
    view.getContext().startActivity( intent );
    return true;
}

```

### A.8.2.2 Switch to Chrome View Client (WebChromeClient)

If the app developer switches to Chrome View Client instead of WebViewClient then the Chrome View Client will invoke the custom URL schemes automatically.

## A.9 Configuring Content Security Policy (CSP)

**Important** This section only applicable if the Content Security Policy (CSP) is enabled.

If the Content Security Policy has been enabled on the server, Zapp recommends adding the Content-Security-Policy HTTP header to the web page by following the below mentioned steps for the library to work properly.

1. Update the Content Security Policy to be able to add the cookie management onto the CDN domains list of **connect-src** white listed domains as shown below

```
connect-src 'self' https://paybybankappcdn.mastercard.co.uk  
https://paybybankappcookie.mastercard.co.uk
```

2. The button must be added to the website by including the following inline script in the HTML body. Zapp recommends generating a **nonce** (a random value) and adding the **nonce** attribute to the script element.

Example:

```
<script nonce="cb9ebecc-0667-44f4-ae41-929dad35a775">  
    new zapp.button({});  
</script>
```

**Note** Ensure to update the CSP header by appending the same **nonce** (generated in step 2 above) to the list of items in **script-src**.