



EIS11 PBBA Integrated 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 | | 18-10-2015 | 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. |
| 2.3 | 2.2 | 25-07-2016 | <ol style="list-style-type: none"> 1. Added information about cookieExpiryDays. 2. Added the GitHub location for the web Merchant button library. 3. Updated the external SharePoint location for the web Merchant button library. 4. Removed unwanted code as per my previous mail. 5. Updated the cookie management URL to: https://www.paybybankapp.co.uk/ |
| 2.4 | 2.3 | 11-10-2016 | <ol style="list-style-type: none"> 1. Cleaned up examples to remove unwanted fields. 2. Updated the PayConnect URL in the examples. 3. Added steps to continue polling upon receipt of a payment not confirmed status. |
| 2.5 | 2.4 | 21-10-2016 | <ol style="list-style-type: none"> 1. Added the setCookie() description to Appendix A. 2. Added support for Safari |
| 2.6 | 2.5 | 25-10-2016 | <ol style="list-style-type: none"> 1. Added section 5.5 to Appendix C on cookie management |
| 2.7 | 2.6 | 07-11-2016 | Updated the document after joint review with Participant |
| 2.8 | 2.7 | 22-11-2016 | <ol style="list-style-type: none"> 1. Added details on listeners to be registered. 2. Added details on functions to be invoked to stop the timers and close the pop-up. 3. Added details on getting the mobile cookie for M-COMM journey. 4. Updated example. 5. Updated sequence diagrams. |
| 3.0 | 2.8 | 11-01-2017 | Updated screenshots to show `Pingit` instead of `your banking app`. |

| Version | | Date | Summary of Changes |
|------------|------|------------|---|
| Was | Now | | |
| 3.1 | 2.9 | 24-01-2017 | Updated the document for custom Merchant button related changes. |
| 3.2 | 2.10 | 01-03-2017 | Rename `Custom` to `Integrated`. |
| 3.3 | 2.11 | 30-03-2017 | Update re polling for current status. |
| 3.0 | | 03-05-2017 | <ol style="list-style-type: none"> 1. Document format revised for R3 2. Added a section with steps to upgrade the library. 3. Updated pop-up close confirmation box screenshot and description. 4. Consumer edge cases added to Appendix. 5. Update to DDOS section regarding use of Cookie token |
| 3.1 | | 10-07-2017 | A.1.2 Pay by Bank app Cookie Management Component - Additional text and table added |
| 3.2 | | 14-07-2017 | <p>Changes to this version:</p> <ul style="list-style-type: none"> ● Document History revised to ensure versions align to Releases ● Figure 2 App Picker – image updated ● Table 3 – Reference to 'iPhone 5' changed to 'iPhone SE' ● Section 3.4 – Note amended to explain that although the version number mentions R2, the code itself will work with Release 3 of PBBA ● Section 3.6 Integrated Web Merchant Button with PBBA pop-up – The text '.' in the first paragraph / second sentence revised ● Figure 5 – Updated screenshot showing version number ● Section 3.6.3 / 7b – The text for the lower figure amended to read 'INPROGRESS status keeps the Polling continue until the Retrieval Timeout Period is reached, the below screen will continue to show during this period'. ● Appendix A.6.3 – Consumer retrieves payment in Mobile Banking app but pays with another payment method- Added ● Appendix A7 – Hybrid Merchant Apps – Added |
| 3.3 | | 23.10-2017 | <p>Note added clarifying the polling mechanism on:</p> <ul style="list-style-type: none"> ● Section 2.2 – M-COMM journey ● Section 2.3 – E-COMM journey - Pay by Bank app code journey Section 2.4 - E-COMM – PayConnect journey |
| 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 |

| Version | | Date | Summary of Changes |
|---------|-----|------------|---|
| Was | Now | | |
| 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"> • Section 1.1 - Introduction - Note on Level AA standard - added • Section 3.2 - Certified Browsers and Devices - table updated • Section 3.6 - Integrated Web Merchant Button with PBBA pop-up - revised for clarity <ul style="list-style-type: none"> ○ description revised for clarity ○ screenshot removed • Section 3.6.1 - PBBA Code Popup - added • Section 3.6.2 - More-About Popup - added • Section 3.6.4 - Integrated Web Merchant Button Setup: <ul style="list-style-type: none"> ○ Procedure Steps 2 - revised for clarity ○ Procedure Steps 7b - screenshot removed ○ Procedure Steps 9 - screenshot added • Section 3.6.5 - Integrated Web Merchant Button Layouts - added • Appendices A.2.2 - Changes to the custom configuration file - pbbacustomconfig.js - revised for clarity • Table 5 - Web Merchant Button Library download locations - revised for clarity • Figure 5 - Pay by Bank app Web Merchant Button Library structure - revised for clarity • Figure 6 - Integrated Web Merchant Button with PBBA Pop-up becomes PBBA Code Popup • Figure 7 - More-About Popup - added • Figure 10 - Mobile App Cookie - revised for clarity • Figure 11 - M-COMM Journey on a mobile browser - revised for clarity • Figure 12 - M-COMM Journey with another device - revised for clarity |
| 3.12.3 | | 11-09-2019 | <p>Changes to this version:</p> <ul style="list-style-type: none"> • Section 1 - Introduction - Important and Note - revised for clarity • Section 3.2 - Certified Browsers and Devices - iPad Mini 2 and iPad Mini 4 - OS Version added • Section 3.5.1 - General requirements: <ul style="list-style-type: none"> ○ Table1 - JQuery version - amended ○ <script - amended |

| Version | | Date | Summary of Changes |
|---------|-----|------------|--|
| Was | Now | | |
| 3.12.3 | | 11-09-2019 | <ul style="list-style-type: none"> ● Section 3.6.1 - PBBA Code Popup: <ul style="list-style-type: none"> ○ Revised for clarity ○ Note - added ● Section 3.6.2 - More-About Popup - Note - added ● Section 3.6.4 - Integrated Web Merchant Button Setup: <ul style="list-style-type: none"> ○ Procedure step 2 - revised for clarity ○ Procedure step 9 - Note - added ● Section 3.6.5 - Integrated Web Merchant Button Layouts: <ul style="list-style-type: none"> ○ Later 3 - Integrated Web Merchant Button with bank logos and more about link with informative text - removed and revised for clarity ○ Layer 4 - revised for clarity ● Table 5 - Web Merchant Button Library download locations – removed. ● Table 7 - Cookie management component - revised for clarity ● Figure 5 - Pay by Bank app Web Merchant Button Library structure - revised for clarity ● Appendices A.1.2 - Pay by Bank app Cookie Management Component - jquery version- amended ● Appendices A.2.1 - Merchant’s HTML file - jquery version- amended ● Appendices A.2.2 - Changes to the custom configuration file - pbbacustomconfig.js - revised for clarity ● Appendices A.8 - Configuring Content Security Policy (CSP) - added |
| 3.12.4 | | 14-10-2022 | External GitHub URL change to internal Mastercard GitHub URL |
| 3.12.5 | | 14-03-2023 | Updated font library |

Contents

| | | |
|----------|---|-----------|
| 1 | About this document | 10 |
| 1.1 | Introduction | 10 |
| 1.2 | Audience..... | 10 |
| 1.3 | Scope..... | 10 |
| 1.4 | Document conventions..... | 10 |
| 1.5 | Associated documents..... | 11 |
| 2 | Functional overview | 12 |
| 2.1 | Introduction | 12 |
| 2.2 | M-COMM journey | 12 |
| 2.2.1 | App Picker..... | 15 |
| 2.3 | E-COMM – Pay by Bank app code journey..... | 16 |
| 2.4 | E-COMM – PayConnect journey..... | 18 |
| 3 | Technical overview | 21 |
| 3.1 | Introduction | 21 |
| 3.2 | Certified Browsers and Devices | 21 |
| 3.3 | Hosting options..... | 22 |
| 3.4 | Pay by Bank app Web Merchant Button Library structure..... | 22 |
| 3.5 | Technical requirements..... | 24 |
| 3.5.1 | General requirements..... | 24 |
| 3.5.2 | Library hosting requirements..... | 24 |
| 3.6 | Integrated Web Merchant Button with PBBA pop-up..... | 25 |
| 3.6.1 | PBBA Code Popup | 25 |
| 3.6.2 | More-About Popup..... | 26 |
| 3.6.3 | Cookie management component..... | 27 |
| 3.6.4 | Integrated Web Merchant Button Setup..... | 29 |
| 3.6.5 | Integrated Web Merchant Button Layouts | 40 |
| 3.6.6 | Integrated Web Merchant Button Sample Code..... | 41 |
| 4 | Additional considerations for M-COMM | 42 |
| 4.1 | Prerequisites for M-COMM..... | 42 |
| 4.2 | Mobile App Cookie – retaining PBBA enabled Bank App selection | 42 |
| 4.2.1 | Setting hasApp cookie | 42 |
| 4.2.2 | Getting hasApp cookie | 46 |
| A | Appendices | 47 |

| | | |
|-------|--|----|
| A.1 | Merchant Configurable Properties..... | 47 |
| A.1.1 | Merchant Poll Intervals..... | 47 |
| A.1.2 | Pay by Bank app Cookie Management Component..... | 47 |
| A.2 | Integrated Web Merchant Button implementation sample code..... | 49 |
| A.2.1 | Merchant's HTML file..... | 49 |
| A.2.2 | Changes to the custom configuration file - pbbacustomconfig.js..... | 50 |
| A.3 | Additional Cookie Management Information..... | 55 |
| A.3.1 | Remove all connections to the Mobile Banking Application Consumer function..... | 55 |
| A.3.2 | DDoS protection..... | 55 |
| A.4 | Polling for Payment Status..... | 55 |
| A.5 | Upgrading the library..... | 56 |
| A.6 | Handling Consumer edge cases..... | 56 |
| A.6.1 | Consumer selects Pay by Bank app more than once per order..... | 56 |
| A.6.2 | Consumer retrieves payment in Mobile Banking app but pays with another payment method..... | 56 |
| A.6.3 | Consumer starts the payment journey in a non-default browser but is returned to the Merchant website with the device default browser post payment..... | 57 |
| A.7 | Hybrid Merchant Apps..... | 57 |
| A.7.1 | Hybrid iOS Apps..... | 57 |
| A.7.2 | Hybrid Android Apps..... | 58 |
| A.8 | Configuring Content Security Policy (CSP)..... | 59 |

Tables

| | | |
|----------|--|----|
| Table 1: | Web Merchant Button Library – Setup requirements..... | 24 |
| Table 2: | Cookie management component..... | 27 |
| Table 3: | PayConnectID Mapping to Distributor API..... | 32 |
| Table 4: | Pay Method - Successful response to RTP..... | 34 |
| Table 5: | PayConnect cookie setting for Pay Status Authorised..... | 39 |
| Table 6: | PayConnectID Set Cookie function..... | 48 |

Figures

| | | |
|-----------|---|----|
| Figure 1: | Interaction between the components of the M-COMM journey..... | 13 |
| Figure 2: | App Picker – sample screens..... | 15 |

| | | |
|------------|---|----|
| Figure 3: | Interaction between the components of the E-COMM PBBA Code journey..... | 17 |
| Figure 4: | Interaction between the components of the E-COMM PayConnect journey | 19 |
| Figure 5: | Pay by Bank app Web Merchant Button Library structure | 23 |
| Figure 6: | PBBA Code Popup..... | 25 |
| Figure 7: | More about Popup | 26 |
| Figure 8: | PayConnect Cookie (pcid)..... | 28 |
| Figure 9: | hasApp Cookie | 28 |
| Figure 10: | Mobile App Cookie..... | 42 |
| Figure 11: | M-COMM Journey on a mobile browser | 44 |
| Figure 12: | M-COMM Journey with another device..... | 45 |

1 About this document

1.1 Introduction

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

Important This version of EIS11 PBBA Integrated 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 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 Integrated 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 |
|------------------|--|
| Important | Highlights important text in the document. |

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

1.5 Associated documents

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

- *Brand Guidelines*
- *PBBA Branded 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 (PBBA) 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 integrated pop-up. This is covered in this document.

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

Merchants and Distributors can integrate their custom payment button with the Pay by Bank app Integrated Web Merchant Button. The additional considerations are covered in this document.

Merchants should contact their Distributor to get Distributor API definitions and also the implementation changes.

2.2 M-COMM journey

The Merchant website 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 a PBBA button which starts the payment. This document covers the PBBA Integrated Web Merchant button only.
 - If this is the first time PBBA has been used on the device and there is at least one PBBA enabled CFI App installed on the device then the PBBA pop-up will appear asking the Consumer to either continue his payment on the same device by pressing 'Open mobile banking app' or get the PBBA code to pay on another device.
 - If this is not a first payment on the device and the Consumer has selected 'Open mobile banking app' 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 website or App displays the payment confirmation page and also stores the Consumer's choice of using PBBA 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.

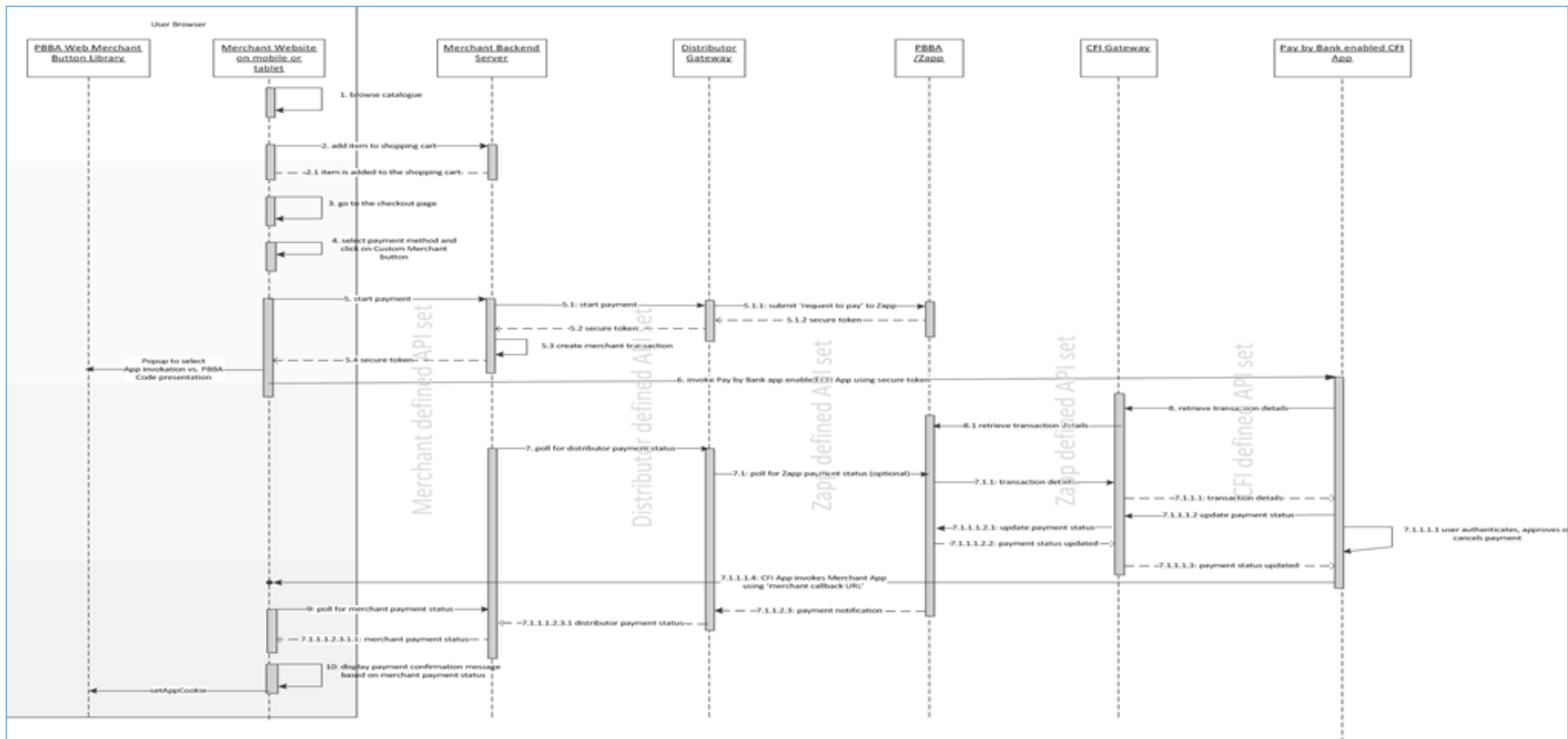


Figure 1: Interaction between the components of the M-COMM journey

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. 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 website or App, an App Picker is displayed (see Figure 2: App Picker – sample screens below) where the Consumer can select which CFI App they would like to use to complete the PBBA payment.

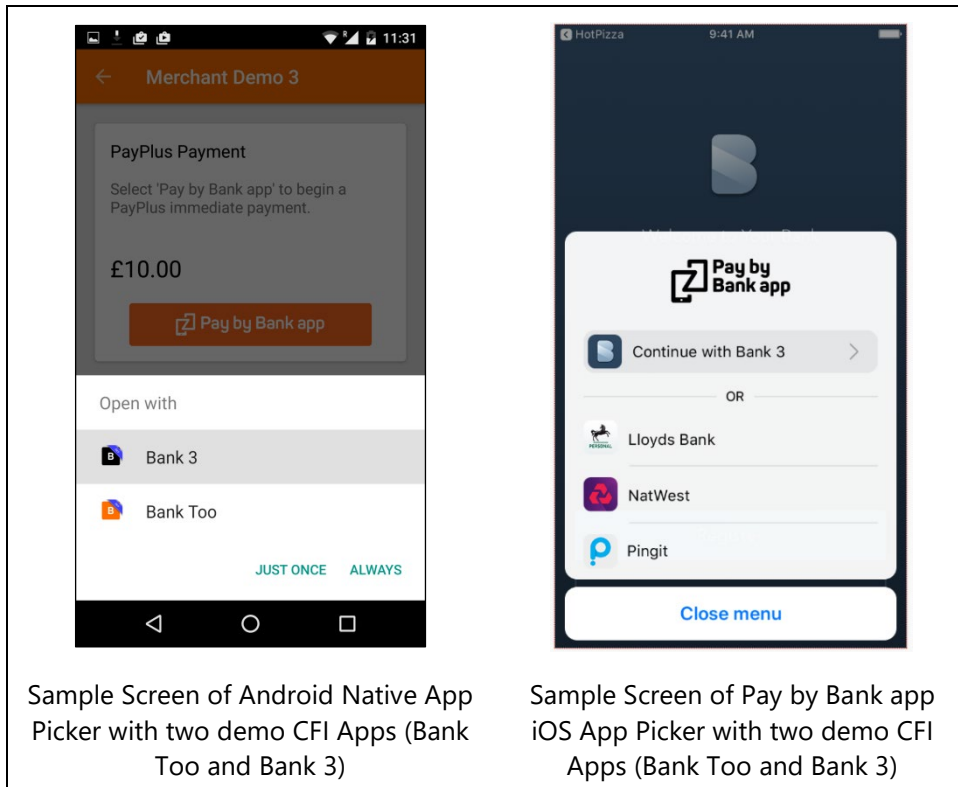


Figure 2: App Picker – sample screens

Note Merchants should contact their Distributor to get Distributor API definitions and also the implementation changes.

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
- The Merchant Website displays a six letter code to the Consumer
- The Consumer opens a Pay by Bank app enabled CFI App on another device and enters the 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 opt for 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 the Merchant along with the Payment Status. This is then set on the Consumer's Browser as a Cookie for future PBBA Payment from this specific browser as a E-COMM PayConnect Journey
- If the Consumer cancels the PayConnect option in the Pay by Bank app enabled CFI App, future Journey will require a Pay by Bank app Code

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.

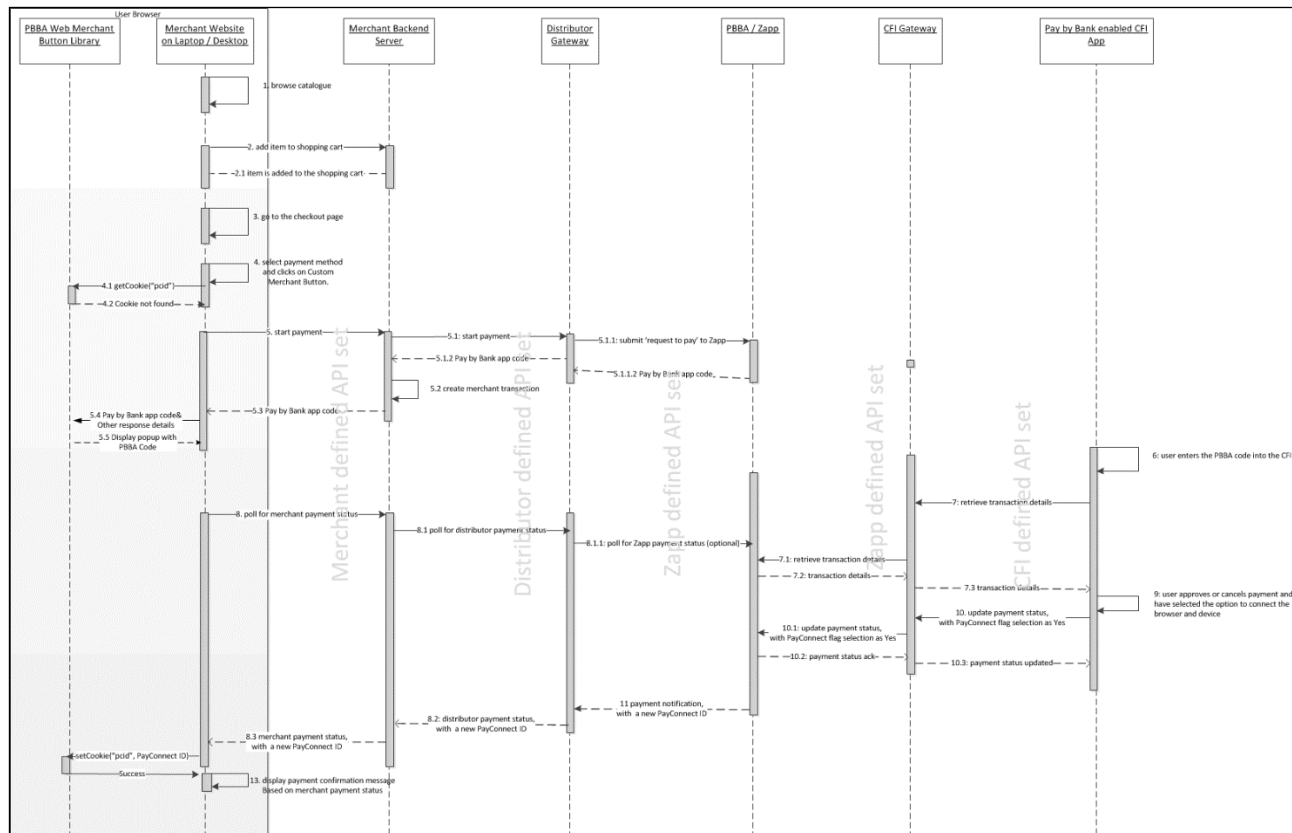


Figure 3: Interaction between the components of the E-COMM PBBA Code journey

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. 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 Pay by Bank app code Journey with a 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 taps 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 Consumer website displays a notification sent pop-up
- 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 taps 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.
- When the payment is completed, a new PayConnect ID and Expiry Days data is sent by the Distributor to the 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.

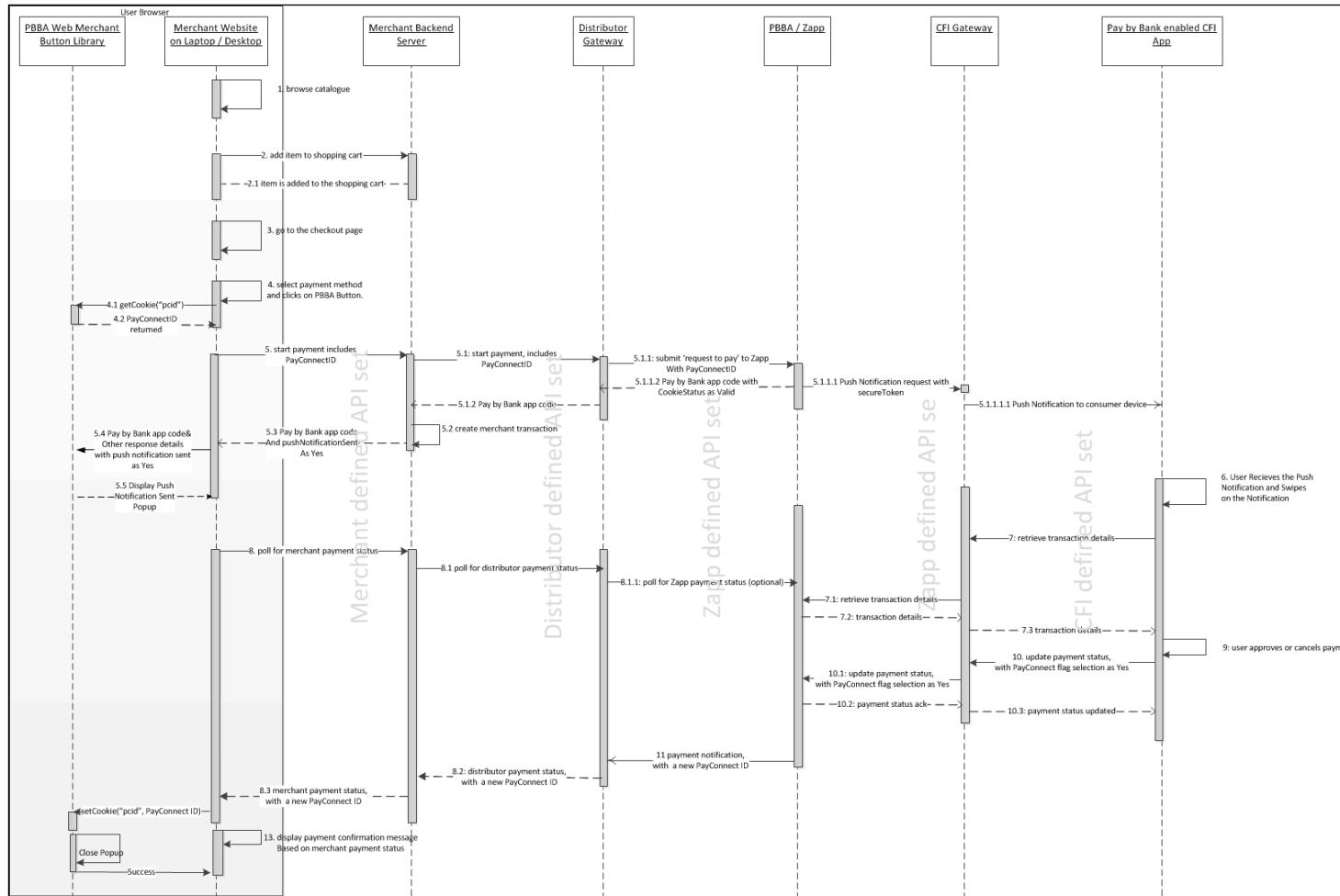


Figure 4: Interaction between the components of the E-COMM PayConnect journey

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. Merchant should contact their Distributor for any Distributor specific implementation updates, API definitions or amendments.

3 Technical overview

3.1 Introduction

This chapter 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 in the attached spreadsheet:

| | 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 | 1440 x 900,1680 X 1050,1600 X 1200,1920 X 1200 |
| | Windows/Firefox | 10 | 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 | |

| | Operating System/ Browser | OS Version | Comments |
|------------------------|------------------------------|------------|----------|
| | 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 | |
| | 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 have 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 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. It is in [Web Merchant Button Implementation Guide Code](#) folder as a compressed ZIP file. Alternatively 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: 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 in to the project.

Note This version of JQuery is the current certified version for Pay by Bank app pop-up 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 3.2 Certified Browsers and Devices.

3.6 Integrated Web Merchant Button with PBBA pop-up

The Merchant can integrate the Pay by Bank app pop-up component with their own button. In this way the colours, themes, fonts and other styling of the payment Button can conform to the Merchant's UX guidelines. If the Merchant has multiple payment options available for the Consumer, Pay by Bank app can also be offered as one of the options. When the Consumer clicks on the Merchant's Pay Button it will display the Pay by Bank app pop-up.

There are only two components to be implemented for the Integrated Web Merchant button:

1. Pop-up component - This component core focus is the pop-up styling and its functions including device specific UI responsiveness.
2. Cookie management component – This component covers the setting and retrieving of multiple cookies like HasApp ('hasApp') and PayConnect ('pcid') cookies.

3.6.1 PBBA Code Popup

The PBBA popup will be displayed upon clicking the branded we merchant button. This popup will display the PBBA code.

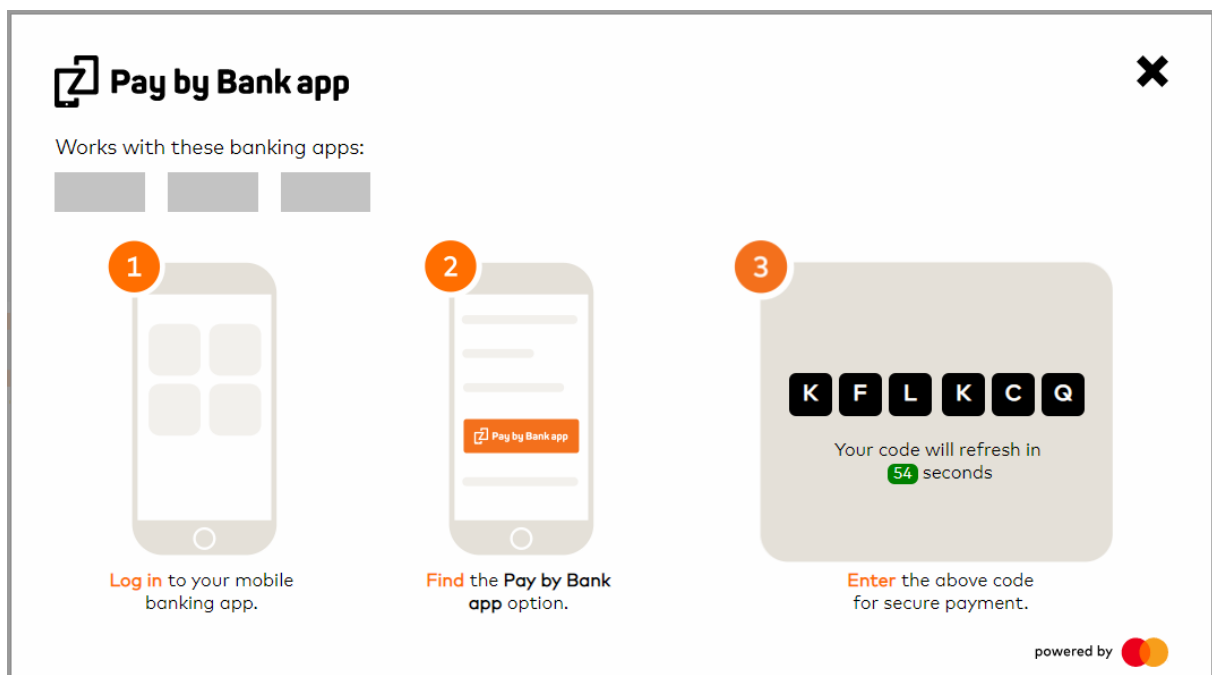


Figure 6: PBBA Code Popup

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). 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.

3.6.2 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.

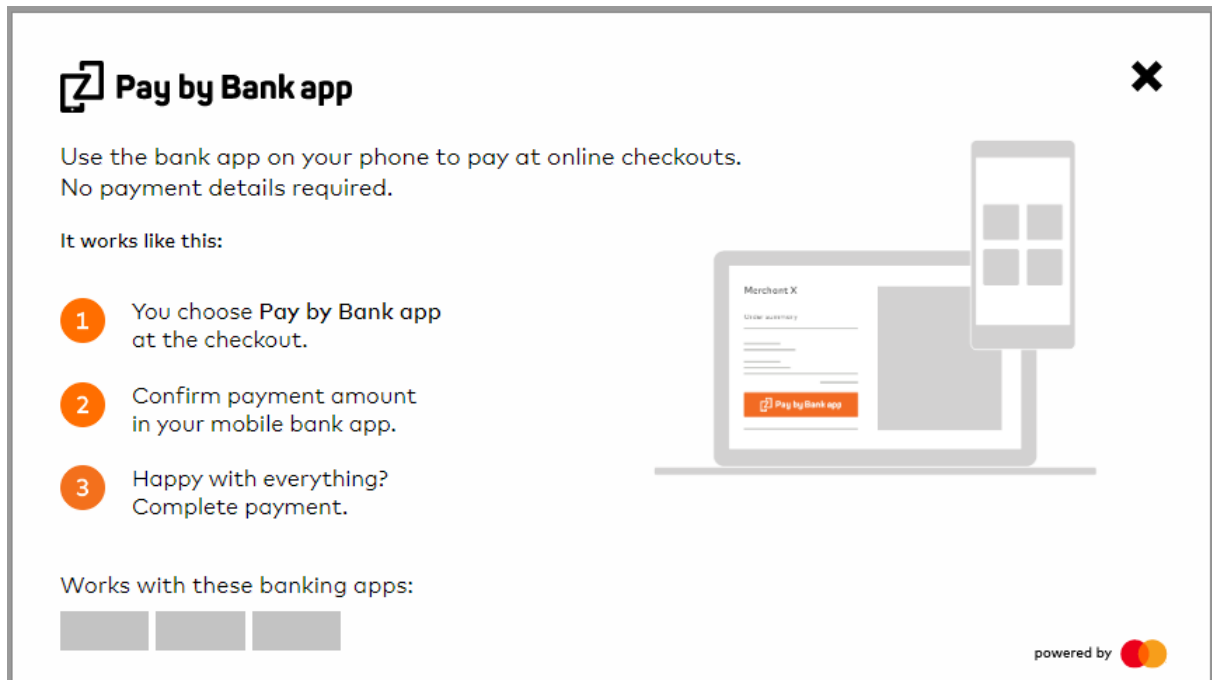


Figure 7: More about Popup

Note: As outlined above, the words “Works with these banking apps” and the grey boxes beneath 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 [Table 2](#) 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 cookie can be set |
| 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 – `pcid` and `hasApp`.

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

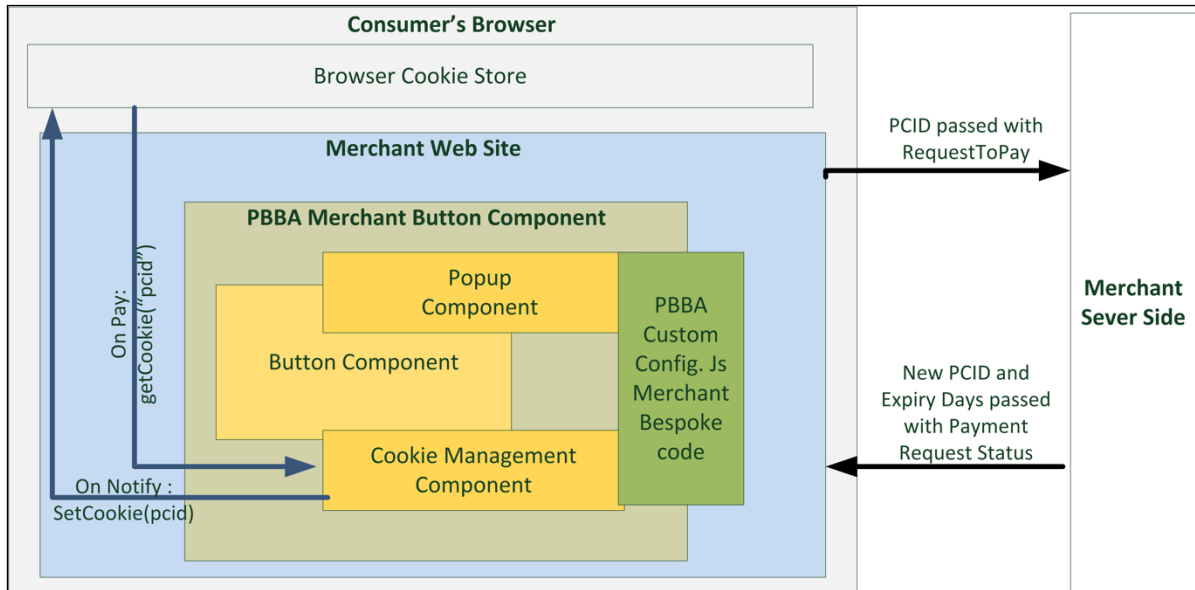


Figure 8: PayConnect Cookie (pcid)

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 explained below:

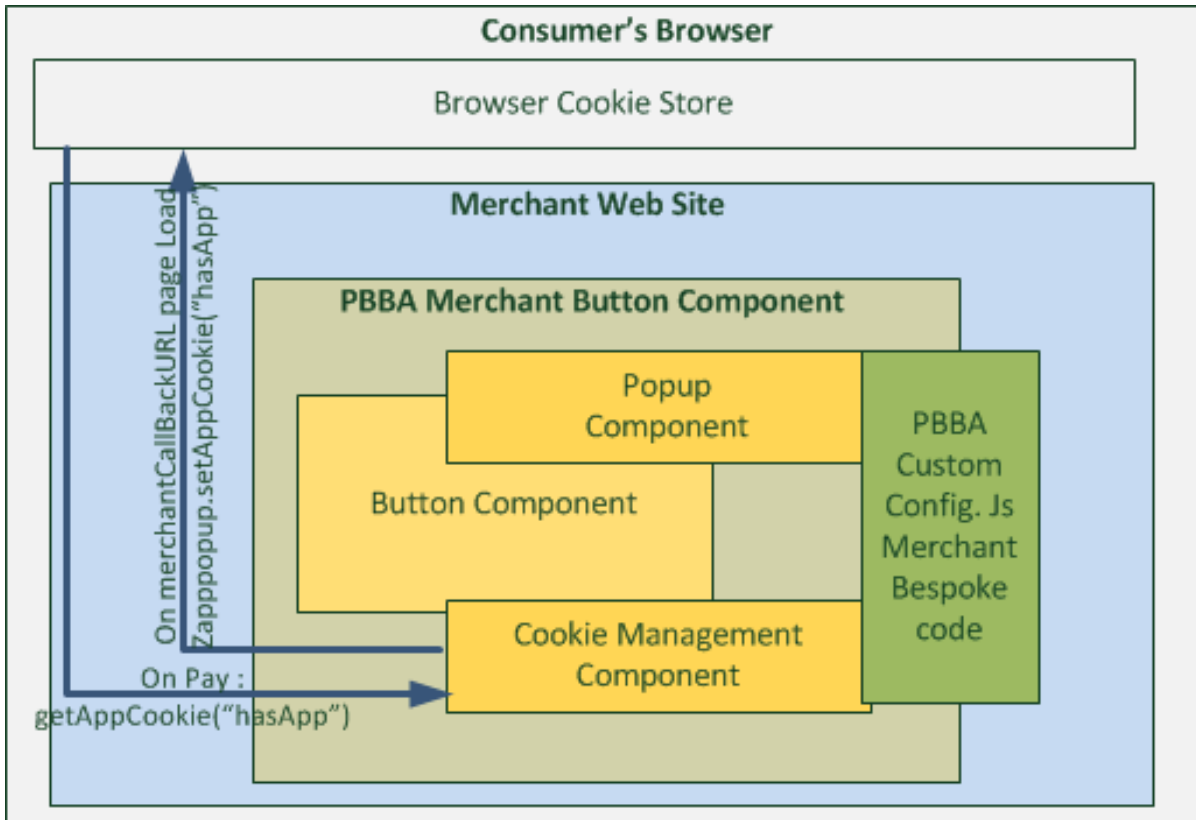


Figure 9: hasApp Cookie

The Setup section of this document (see section 3.6.4 Integrated Web Merchant Button Setup) provides integration details of the pop-up component with the Integrated Web Merchant Button.

3.6.4 Integrated Web 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 integrate the PBBA pop-up and PBBA Cookie Management component with the Integrated Merchant Button.

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/zpopup.js"></script>
  </head>
</html>
```

2. 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 this document.

The file pbbacustomconfig.js file contains the Branded Button implementation by default. In order to implement the Integrated Button, copy the contents of the file pbbacustomconfig_custom.template to the file pbbacustomconfig.js.

Import the file `pbbacustomfoncig.js` to the parent html page after importing the `zpop-up.js` file.

Example:

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

  </head>
</html>
```

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

- `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 zappVersion = "3.1.2";
```

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

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

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

3. Initialise the pop-up engine:

```
zapppopup.load(zappVersion, {
  cookieManagementUrl: cookieManagementUrl
} );
```

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

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

5. Register the following listeners.

These listeners will help capture various events like timeout, pop-up close etc. and you can write your custom logic in case of each event. Please ensure that the event names (example: pbba.transaction.timeout) are used exactly the way they are mentioned below.

```
function listener(event) {

// The first step is to parse the event data. The event object is constructed within the button engine.
// An event indicates a Consumer operation like popup close or a popup operation like transaction timeout.
  try {
    var data = JSON.parse(event.data);
  } catch (exception) {
    return;
  }

  if (data.eventType == "pbba.transaction.timeout") {
    // This event indicates that the Transaction has timed out and polling must be stopped.
    // The logic to stop polling must be implemented by the Merchant.
  }

  if (data.eventType == "pbba.button.regen.click") {
    // This event indicates that the Consumer clicked on the PBBA button on the popup.
    // The previous polling must be stopped and a new payment request should be sent to the Merchant Server.
// The logic to stop the previous polling must be implemented by the Merchant.
  }

  if (data.eventType == "pbba.popup.close") {
    // This event indicates that the Consumer decided to close the popup. In this case, the polling must be stopped.
    // The following two functions must be called to clear the internal PBBA timers and remove the popup.
    zappop-up._stopTimers();
    zappop-up._removePopup(true);
  }

}
```

6. The Merchant posts request with pay data to the Merchant Server and gets a response to request to pay. It is left to the Merchant to decide how this is done, with the only exception of the PayConnectID from the Zapp Specific Data Object.

The PayConnectID can be obtained by using the following function:

```
zapppopup.getCookie('pcid')
```

The Merchant should include this PayConnectID in the request to pay call to the Merchant Server.

The table below provides the mapping of the PayConnectID 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: PayConnectID Mapping to Distributor API

- 7a. Show the Pay by Bank app Code on the pop-up.

To do this the Merchant must create a response object as shown below. Once a successful response is received after the request to pay is posted to the Merchant Server, create a response object using the following syntax should be created to call the pop-up

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

```
var response = new zapppopup.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 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 Consumer, the pop-up component of the button shows the appropriate pop-up 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 pop-up when displays that a push notification is sent out, it also displays the CFI name. | < Consult Distributor Documentation > |

Table 4: Pay Method - Successful response to RTP

7b. The response object is constructed.

Once the response object is constructed, add the following line of code to render a Notification Sent message on the pop-up:

```
if (merchantRequestToPayResponseObject.cookieSentStatus) {
    response.pcid = zapppopup.getCookie('pcid');}
```

7c. Check the hasApp cookie exists.

After the PayConnect check is implemented, check if the hasApp cookie exists. The hasApp cookie will be present if an M-COMM journey has been performed before. The following code will invoke the app instead of showing the pop-up in the second M-COMM journey.

```
if (zapppopup.getCookie("hasApp")) {
    zapppopup._invokeAppWithResponse(<clicked Merchant Consumer Button
    object>, <response object constructed in step 7a above>);
    // Start the polling process to poll for a payment response.
    return;
```

7d. If the hasApp cookie does not exist.

If hasApp cookie is not present then show the pop-up using this syntax:

```
zapppopup._addPopup(<clicked Merchant Consumer Button
object>).sendMessage(<clicked Merchant Consumer Button object>,
"com.zapp.popup.data",<response object constructed in step 7a above>);
```

Example:

```
var clickedButton = this;
zapppopup._addPopup(clickedButton).sendMessage(clickedButton,
"com.zapp.popup.data", response);
```

Note 'this' is the reference to the Button that was clicked. In JavaScript realm, the keyword "this" holds the reference.

8. Error response for request to pay.

In case of error response for request to pay, use the following syntax to show the error on the pop-up:

```
zapppopup._addPopup(<clicked Merchant Consumer Button
reference>).sendMessage(<clicked Merchant Consumer Button reference >,
"com.zapp.popup.data","requestFailure");
```

Example:


```
var clickedButton = this;
error : function(response) {
    zapppopup._addPopup(clickedButton).sendMessage(clickedButton,
"com.zapp.popup.state", "requestFailure");
}
```


9: Poll for Payment Status.

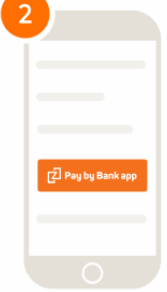
When the Pay by Bank app code is generated, the Merchant Server is polled for the payment status. The Merchant can decide how to implement polling with the exception of Zapp pop-up specific functions and variables.

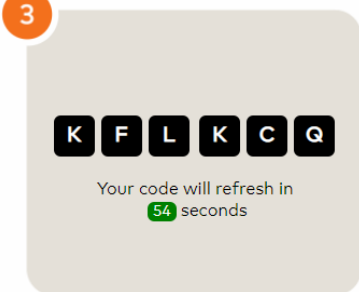
Pay by Bank app ✕

Works with these banking apps:




- 

1 Log in to your mobile banking app.
- 

2 Find the Pay by Bank app option.
- 


3 Enter the above code for secure payment.


powered by 


OR

Pay by Bank app ✕

Works with these banking apps:




- 


1 Open the notification from the banking app
- 

2 Log in and complete your purchase

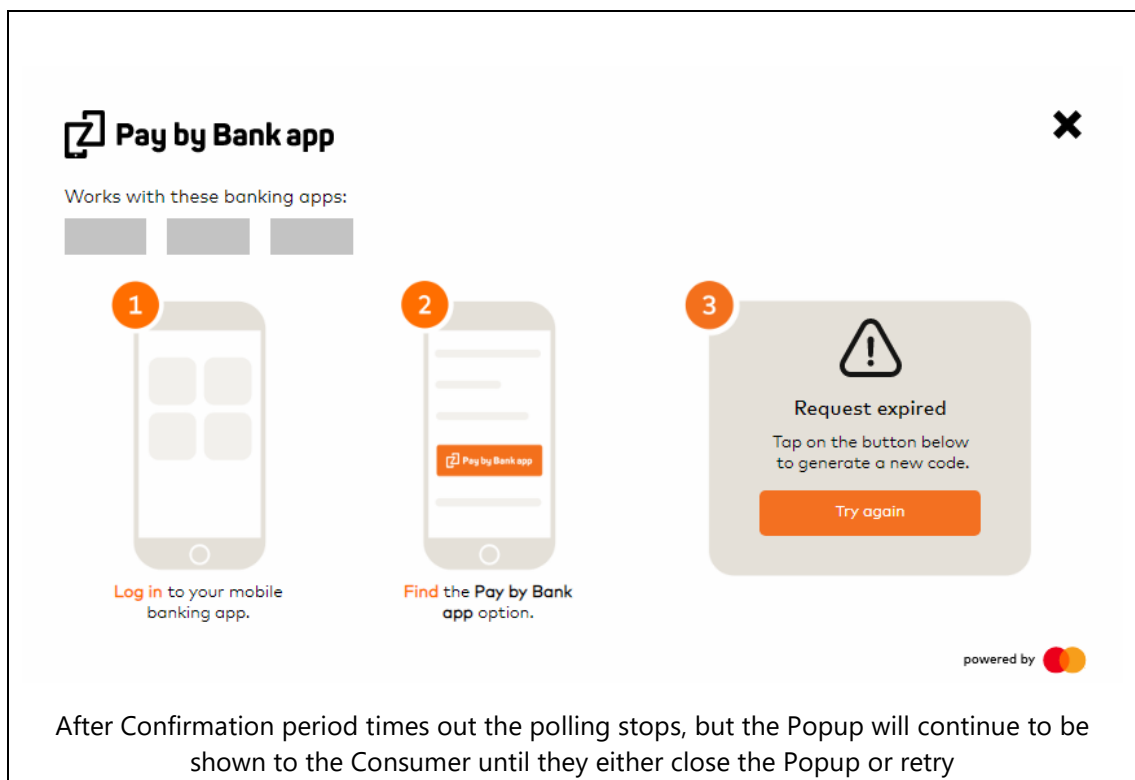
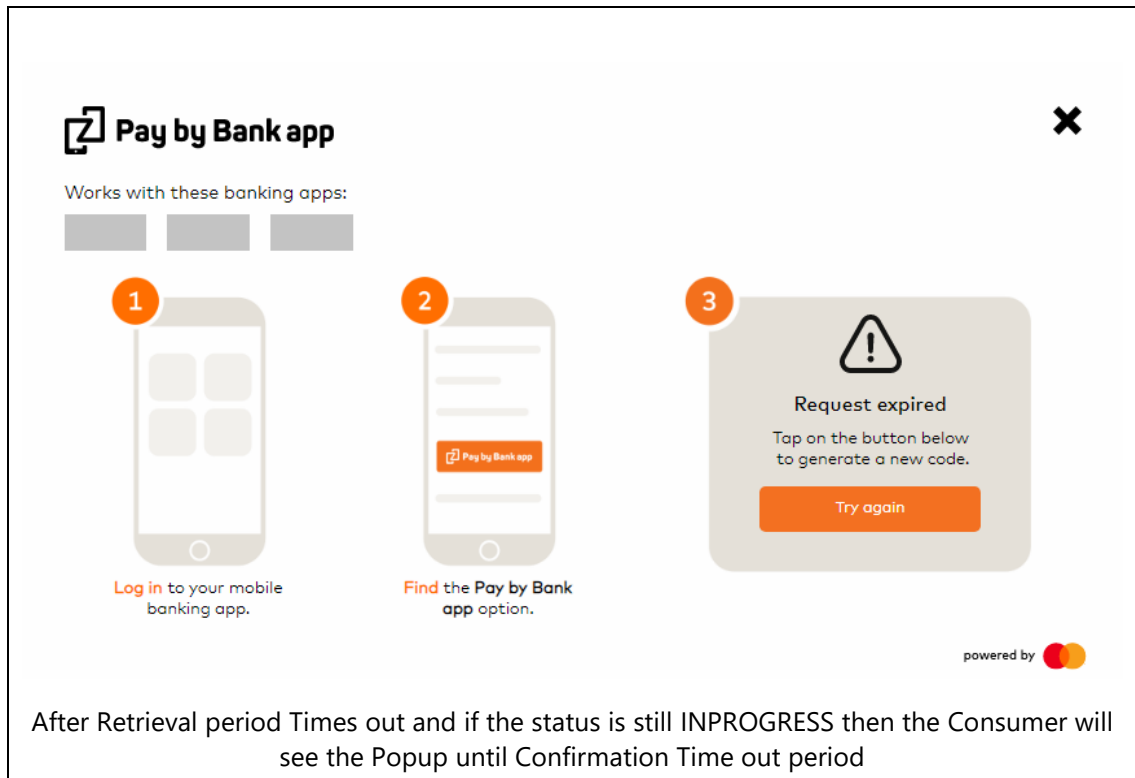
No notification?

Didn't receive a notification or want to pay on another device?

Get  code

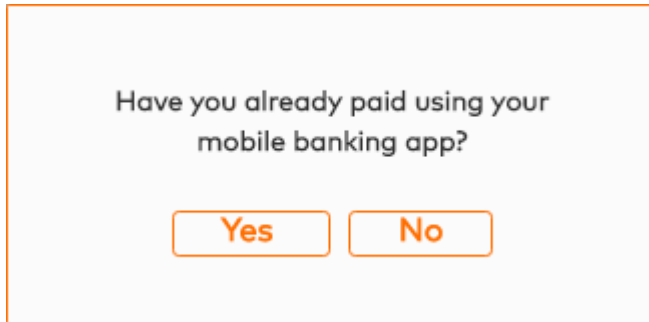
powered by 

INPROGRESS status leads to the continuation of polling until the Retrieval Timeout Period is reached, the above screen will be displayed during this period



- If the Consumer tries to close the pop-up 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.



Have you already paid using your mobile banking app?

Yes No

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

9a. After a successful response.

Once a successful response is received, check for the existence of the PayConnect ID in the response data. If present then call the following function , note that the 1st argument, "pcid", should remain untouched as this is the cookie key

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

| Merchant Request To Pay Response Element Name | Element Description | Distributor API Name / Element Name |
|---|---|---------------------------------------|
| merchantGetPaymentStatusObject.payConnectID | 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 > |
| merchantGetPaymentStatusObject.cookieExpiryDays | 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

9b. After the successful response is received.

Remember to clear the timers and close the pop-up by calling the following functions:

```
zappopup._stopTimers(); // This will clear the pop-up timers
zappopup._removePopup(true); // This will close the popup. The flag true
indicates force close.
```

9c: On receipt of an in progress status

Continue to poll for a payment response using the following syntax:

```
setTimeout(function() {
  // Invoke the function to poll the merchant server
}, merchantPollInterval);
```

9d. If an error is received

On receipt of an error, the Merchant can implement custom error handling mechanism but must remember to invoke the following functions:

```
zappopup._stopTimers(); // This will clear the popup timers
zappopup._removePopup(true); // This will close the popup. The flag true
indicates force close.
```

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

3.6.5 Integrated Web Merchant Button Layouts

Integrated button can be displayed in four different layouts. The assets for these layouts can be obtained by calling four different APIs that are available in the merchant button library. These APIs are present in `zpopup-extras.js`. This file is imported when the merchant library is initialized.

Following are the four APIs that the merchants can invoke to display any of the four layouts mentioned below.

1. `getIntegratedBtnLogo()` – This API will return the URL to the PBBA logo
2. `getCfiLogos()` – This API will return the CFI Logos. Before using this API, the following API must be invoked:
 - `readJSONFile(zappopup.options.cfiLogosURL)` – After invoking this API, it is advised to wait for at least 500 milliseconds before invoking the `getCfiLogos()` method. For example:

```
readJSONFile(zappopup.options.cfiLogosURL);
setTimeout(function() {
  var cfiLogos = getCfiLogos();
}, 500);
```

3. `getMoreAboutLinkWithPaySecureText()` – This API will return the more about link along with an informative text. Add an on-click event to invoke `showMoreAboutPopup` function after invoking the API.

Example:

```
document.getElementById("moreAboutPopupWithPaySecureTextLink").addEventListener("click", showMoreAboutPopup);
```

4. `getMoreAboutLinkWithoutPaySecureText()` – This API will return the more about link without any informative text. Add an on-click event to invoke `showMoreAboutPopup` function after invoking the API.

Example:

```
document.getElementById("moreAboutPopupWithoutPaySecureTextLink").addEventListener("click", showMoreAboutPopup);
```

Note The radio button in the layout examples given below is for demonstration purpose only. It is not returned as a part of any API calls.

1. Integrated Web Merchant Button with the PBBA logo



Invoke the following API to get the merchant button logo:

```
getIntegratedBtnLogo()
```

2. Integrated Web Merchant Button with bank logos and more about link



Invoke the following APIs to get the assets required to display this layout:

```
getIntegratedBtnLogo()
```

```
getMoreAboutLinkWithoutPaySecureText()
```

3.6.6 Integrated Web Merchant Button Sample Code

See Appendix [A.2](#) Integrated Web Merchant 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 illustrates 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 pop-up with the option to open the app.

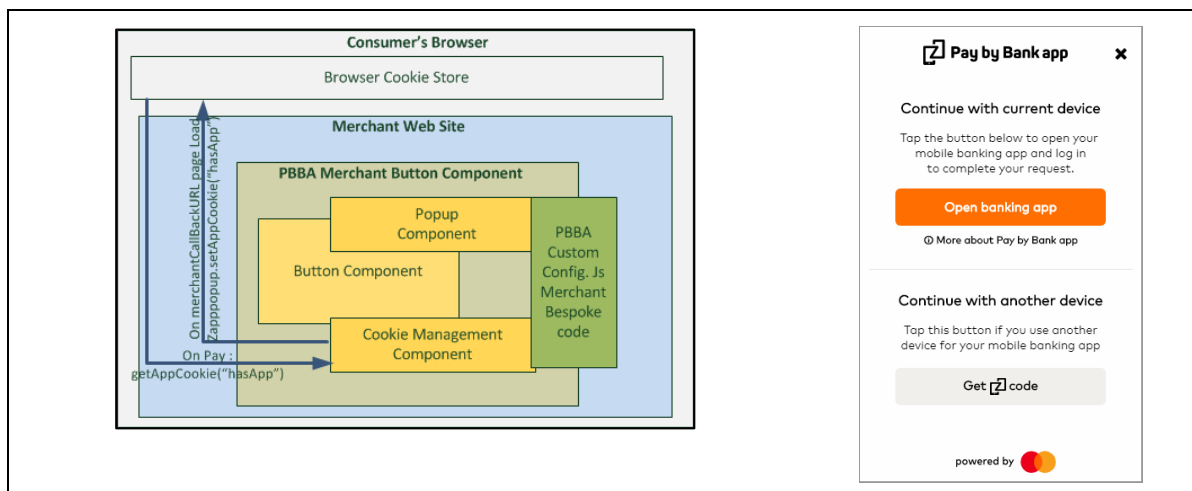


Figure 10: Mobile App Cookie

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 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:

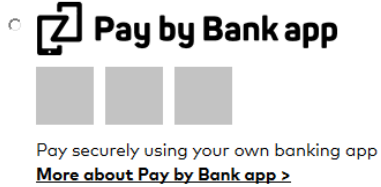
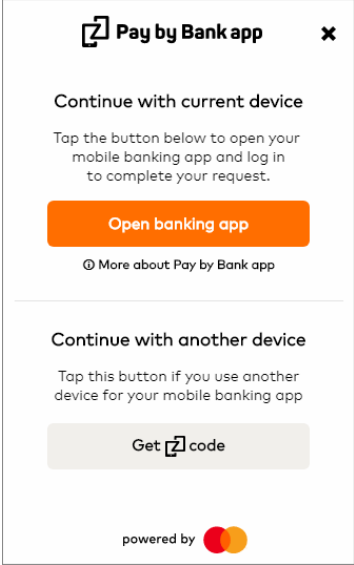
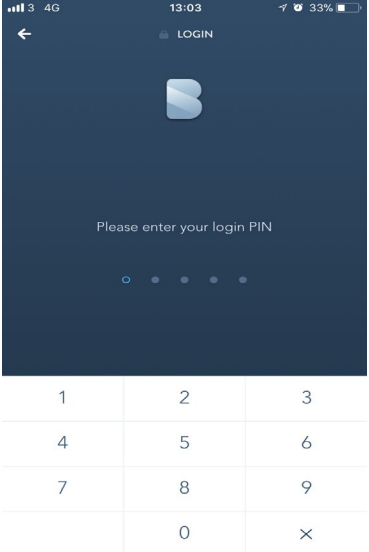
| | | |
|--|--|---|
| <p>1. Consumer starts a Pay by Bank app Journey for the first time on a Browser</p> | <p>2. Consumer is provided with a selection, as shown below</p> | <p>3. The PBBA enabled CFI bank App on the same device is invoked automatically</p> |
|  |  |  |
| <p>Consumer selects the Pay by Bank app option and clicks the Integrated Web Merchant Button (Pay) 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:

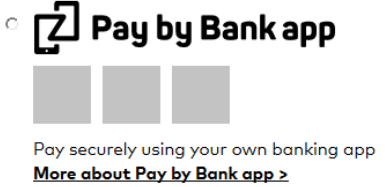
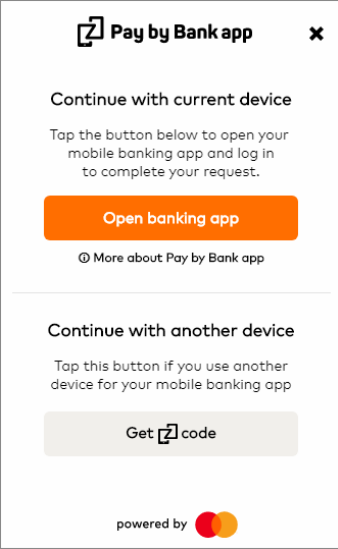
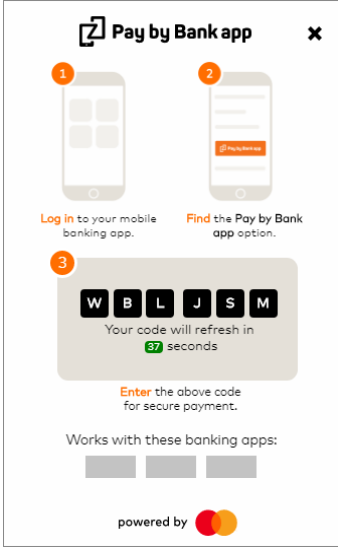
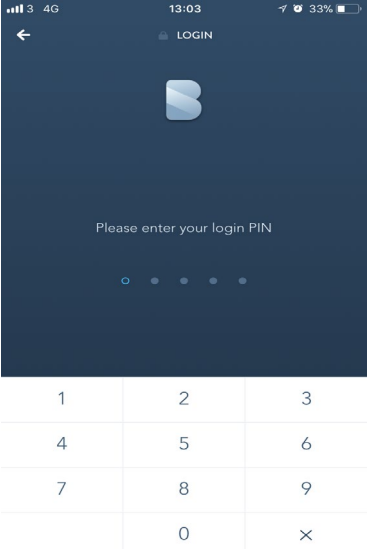
| | | | |
|--|---|--|--|
| <p>1. Consumer starts a PBBA Journey for the first time on a Browser</p> | <p>2. Consumer is provided with a selection, as shown below</p> | <p>3. Consumer selects 'Get PBBA Code' in the 'Continue with another device' section of the pop-up</p> | <p>4. Consumer completes the Journey on another device with a PBBA Enabled CFI App</p> |
|  |  |  |  |
| <p>Consumer selects the Pay by Bank app option and clicks the Integrated Web Merchant Button (Pay) 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

4.2.2 Getting hasApp cookie

In case of the Branded PBBA Web Merchant Button, the hasApp cookie is identified by the button itself. However, in case of the Integrated Web Merchant Button, the following code needs to be added after step 7b above to pbbacustomconfig.js.

`clickedButton` is the reference to the button object that was clicked.

`response` is the response object that was constructed after the response to requests to pay was received from the Merchant Server.

```
if (zappop-up.getCookie("hasApp")) {
    zappop-up._invokeAppWithResponse(clickedButton, response);
    // Start polling for payment notification
return;
```

A Appendices

A.1 Merchant Configurable Properties

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 polling 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 `zappopup.load()` function as mentioned below:

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

A.1.2 Pay by Bank app Cookie Management Component

The PBBA Pop-up 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 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 display of 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 function to the page:

```

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

Where:

`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 |
|--|---|---------------------------------------|
| <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 6: PayConnectID Set Cookie function

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 Integrated Web Merchant Button implementation sample code

The example below is a sample pbbacustomconfig.js file depicting the implementation of the custom methods to post payment request to the Merchant Server and polling for payment notification from the Merchant Server:

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

This is an example of a pbbacustomconfig.js file showing a sample implementation of the Integrated Web Merchant Button integrated with the PBBA pop-up. Zapp Distributor gateway has been used for this sample.

Assumption This is the HTML file for the Merchant's website and contains the following HTML code for rendering the Button.

A.2.1 Merchant's HTML file

```
<HTML>
<HEAD>
<script type="text/javascript" src="<Merchant or Distributor web server URL>/zapp_default/zpopup.js"></script>
<script type="text/javascript" src="<Merchant or Distributor web server URL>/jquery-3.4.1.min.js"></script>
<script type="text/javascript" src="<Merchant or Distributor web server URL>/zapp_default/pbbacustomconfig.js"></script>
</HEAD>
<BODY>
...
<Button type="Button" title="Pay" class="customButton" onclick=" postPaymentRequestToMechantServer (this) ">
<span>Pay</span>
</Button>
...
</BODY>
</HTML>
```

A.2.2 Changes to the custom configuration file - pbbacustomconfig.js

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

var zappVersion = "3.1.2";
var cookieManagementUrl = "https://paybybankappcookie.mastercard.co.uk/static/cookie-management/pbba-3550ce7763041531b9214e9e23986b37/";
var merchantPollInterval = 5000;
var cfiLogosURL = "https://paybybankappcdn.mastercard.co.uk/static/ml/pbba-3550ce7763041531b9214e9e23986b37/merchant-lib/banks.json";// CDN location for CFI logos

var clickedButton = null;

zapppopup.load(zappVersion, {
  cookieManagementUrl: cookieManagementUrl,
  cfiLogosURL : cfiLogosURL
} );

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

// Define the listeners.
function listener(event) {

  try {
    var data = JSON.parse(event.data);
  } catch (exception) {
    return;
  }
}
```

```
if (data.eventType == "pbba.transaction.timeout") {
    // Abort the current polling process
}

if (data.eventType == "pbba.button.regen.click") {
    // Abort the current polling process
    // Start a new payment process.
    postPaymentRequestToMechantServer(clickedButton);
}

if (data.eventType == "pbba.popup.close") {
    // Abort the current polling process
    // Stop the timers and remove the popup.
    zapppopup._stopTimers();
    zapppopup._removePopup(true);
}

}

// Register the listeners
if (window.addEventListener){
    addEventListener("message", listener, false)
} else {
    attachEvent("onmessage", listener)
}

function postPaymentRequestToMechantServer(clickedBtn) {

    clickedButton = clickedBtn; // This is the clicked button reference

    var merchantRequestToPayPostData = {
        "pcid" : zapppopup.getCookie('pcid')
    };

    jQuery.ajax({
```

```

url : "MerchantRequestToPayPostOrder", //The merchant URL to post the Request To Pay
dataType : "json",
type : "POST",
crossDomain : true,
data : merchantRequestToPayPostData,
success : function(merchantRequestToPayResponseObject) {

    var response = new zapppopup.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
    });

    if (merchantRequestToPayResponseObject.cookieSentStatus) {
        response.pcid = zapppopup.getCookie('pcid');
    }

// If an M-Comm journey has been performed before, hasApp cookie will be set on the mobile browser. Check for the hasApp
// cookie and invoke the app if present.
    if (zapppopup.getCookie("hasApp")) {
        zapppopup._invokeAppWithResponse(clickedButton, response); // Invoke the app
        pollMerchantServerForPaymentNotification(response.secureToken); // Start the polling process
        return;
    }

zapppopup._addPop-up(clickedButton).sendMessage(clickedButton, "com.zapp.popup.data", response); // Display the popup.

    zapppopup._startTimers(clickedButton); // Start the notification timers.

    pollMerchantServerForPaymentNotification (response.secureToken); // Invoke the polling method.
},

```

```
        error : function(merchantRequestToPayResponseObject) {
            zapppopup._addPopup(clickedButton).sendMessage(clickedButton, "com.zapp.popup.state",
                "requestFailure"); // Display the error message on the popup.
        }
    });
}

function pollMerchantServerForPaymentNotification(secureToken) {
    var _confirmOrder = function(merchantResponse) {
        // Merchant specific order processing logic to show success or cancel page goes here.
    };
    jQuery.ajax({
        url : "/getstatus/merchantgetstatuscall.<something>", //this is merchant backend server call
        dataType : "json",
        crossDomain : true,
        cache: false,
        type : "GET",
        success : function(merchantGetPaymentStatusObject) {

            if (<payment not confirmed>) {
                // Continue polling using the merchant poll interval.
                setTimeout(function() {
                    pollMerchantServerForPaymentNotification (secureToken);
                }, merchantPollInterval);
            } else {
                merchantResponse = JSON.parse(merchantGetPaymentStatusObject);

                zapppopup._stopTimers();
                zapppopup._removePopup(true);

                if (typeof merchantResponse.payConnectID !== "undefined") {
                    setCookie("pcid", merchantResponse.payConnectID, merchantResponse.cookieExpiryDays, cookieManagementUrl);
                }

                _confirmOrder(merchantResponse);
            }
        }
    });
}
```

```
    }  
    },  
    error : function() {  
        zappopup._stopTimers(); // Stop the timers  
        zappopup._removePopup(true); // Remove the popup  
    }  
});  
}
```

-----FINISH-----

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 have 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 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:

- Zapp suggests disabling the Integrated Web Merchant Button for approximately 10 seconds after the button being clicked to prevent multiple clicks using automated scripts/robots.
- An individual cookie token can only be submitted ten (10) 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 PBBA 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

A.4 Polling for Payment Status

Whilst the Pay by Bank app Integrated 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.5 Upgrading the library

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

- Back up the existing library (the zapp_default folder)
- 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.
- Replace the existing library with the latest library. This means replace the contents of the existing zapp_default folder with the contents of the zapp_default folder from the newly downloaded library
- 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.6 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.6.1 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.4 Polling for Payment Status for more information on polling for payment status.

A.6.2 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.6.1 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.6.3 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 into the Merchant website.
- 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.7 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.7.1 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.7.1.1 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.7.1.2 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.7.2 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.7.2.1 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.7.2.2 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.8 Configuring Content Security Policy (CSP)

Note 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 step for the library to work properly.

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
```