**PBBA Branded Web Merchant Button**

Implementation Guide

April 2017

Version 4.0

**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.

© VocaLink Limited 2017. All rights reserved

Document History

|  |  |  |
| --- | --- | --- |
| Version | Date | Summary of Changes |
| 1.3 | 18/11/2015 | Final Version 1 Draft |
| 2.0 | 19/01/2015 | 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 | 29/01/2015 | Removed hosted option as deprecated following browser standards and security reviews. |
| 2.0 Final | 20/05/2016 | Released as effective from 20 May to support Pay by Bank app live service. |
| 2.1 Draft | 23/06/2016 | Updated document to add appendix and sequence diagrams. |
| 2.2 Draft | 06/07/2016 | Updated with the Theme information and peer review comments |
| 2.3 Final Draft | 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 <https://www.paybybankapp.co.uk/> |
| 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. |
| 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 | 11/01/2017 | Updated screenshots to show “Pingit” instead of “your banking app”. |
| 3.1 | 24/01/2017 | Updated the document for custom merchant button related changes. |
| 3.2 | 01/03/2017 | Update re polling for current status. |
| 3.3 | 30-03-2017 | Updated Links to the latest Web Merchant Button Library |
| 4.0 | 00-00-2017 | Changes to this version |

Contents

[1 About this document 8](#_Toc479672293)

[1.1 Introduction 8](#_Toc479672294)

[1.2 Audience 8](#_Toc479672295)

[1.3 Scope 8](#_Toc479672296)

[1.4 Document conventions 8](#_Toc479672297)

[1.5 Associated documents 8](#_Toc479672298)

[2 Functional overview 9](#_Toc479672299)

[2.1 Introduction 9](#_Toc479672300)

[2.2 M-COMM Journey 9](#_Toc479672301)

[2.2.1 App Picker 11](#_Toc479672302)

[2.3 E-COMM – Pay by Bank app Code Journey 12](#_Toc479672303)

[2.4 E-COMM – PayConnect Journey 14](#_Toc479672304)

[3 Technical Overview 16](#_Toc479672305)

[3.1 Introduction 16](#_Toc479672306)

[3.2 Certified Browsers and Devices 16](#_Toc479672307)

[3.3 Hosting Options 17](#_Toc479672308)

[3.4 Branded Pay by Bank app Web Merchant Button library structure 18](#_Toc479672309)

[3.5 Technical Requirements 19](#_Toc479672310)

[3.5.1 General requirements 19](#_Toc479672311)

[3.5.2 Library hosting requirements 19](#_Toc479672312)

[3.6 Branded Web Merchant Button (Standard Offering) 20](#_Toc479672313)

[3.6.1 Button component 20](#_Toc479672314)

[3.6.2 Popup component 20](#_Toc479672315)

[3.6.3 Cookie management component 21](#_Toc479672316)

[3.6.4 Pay by Bank app Branded Merchant Button Setup 23](#_Toc479672317)

[3.6.5 Pay by Bank app Branded Merchant Button Sample Code 34](#_Toc479672318)

[4 Additional considerations for M-COMM 35](#_Toc479672319)

[4.1 Prerequisites for M-COMM 35](#_Toc479672320)

[4.2 Mobile App Cookie – retaining PBBA enabled Bank App selection 35](#_Toc479672321)

[4.2.1 Setting hasApp cookie 35](#_Toc479672322)

[4.3 Integrated Web Merchant Button 39](#_Toc479672323)

[A Appendices 40](#_Toc479672324)

[A.1 Merchant Configurable Properties 40](#_Toc479672325)

[A.1.1 Merchant Poll Intervals 40](#_Toc479672326)

[A.1.2 Co-branded Pay by Bank app Button Images 40](#_Toc479672327)

[A.2 Pay by Bank app Button implementation sample code 41](#_Toc479672328)

[A.3 Changing the look and feel of the Button 46](#_Toc479672329)

[A.4 Additional Cookie management Information 46](#_Toc479672330)

[A.4.1 Remove all connections Mobile Banking Application (Pingit) Consumer function 46](#_Toc479672331)

[A.4.2 DDoS protection 46](#_Toc479672332)

[A.5 Known browser specific requirements 47](#_Toc479672333)

[A.5.1 Internet Explorer 47](#_Toc479672334)

[A.6 Polling for Payment Status 48](#_Toc479672335)

Tables

[Table 1: Certified Browsers and Devices 16](#_Toc478732719)

[Table 2: Compatibility with mobile devices and operating systems (Android) 16](#_Toc478732720)

[Table 3: Compatibility with mobile devices and operating systems (Apple) 17](#_Toc478732721)

[Table 4: Third party component 17](#_Toc478732722)

[Table 5: Web Merchant Button Library download locations 17](#_Toc478732723)

[Table 6: Web Merchant Button library – Setup requirements 19](#_Toc478732724)

[Table 7: Cookie management component 21](#_Toc478732725)

[Table 8: PayConnect ID mapping to Request to Pay 26](#_Toc478732726)

[Table 9: Response to Request to Pay Mapping 28](#_Toc478732727)

[Table 10: PayConnect cookie setting for Pay Status Authorised 33](#_Toc478732728)

Figures

[Figure 1: Interaction between the components of the M-COMM journey 10](#_Toc478732729)

[Figure 2: App Picker – sample screens 11](#_Toc478732730)

[Figure 3: Interaction between the components of the E-COMM PBBA Code journey 13](#_Toc478732731)

[Figure 4: Interaction between the components of the E-COMM PayConnect journey 15](#_Toc478732732)

[Figure 5: Branded PBBA Web Merchant button library structure 18](#_Toc478732733)

[Figure 6: Popup component 20](#_Toc478732734)

[Figure 7: PayConnect Cookie (pcid) 22](#_Toc478732735)

[Figure 8: hasApp Cookie 22](#_Toc478732736)

[Figure 9: Mobile App Cookie 35](#_Toc478732737)

[Figure 10: M-COMM Journey on a mobile browser 37](#_Toc478732738)

[Figure 11: M-COMM Journey with another device 38](#_Toc478732739)

# About this document

## 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.

1. The Merchant Button Library is a mandated aspect of Pay by Bank app and must be used for any implementation of Pay by Bank app’s products or services.

Implementation support is available on request.

## Audience

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

## 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.

## Document conventions

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

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

## Associated documents

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

* *Brand Guidelines*
* *PBBA Integrated Web Merchant Button Implementation Guide*
* *Zapp Glossary*

# Functional overview

## 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.

1. 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 the model described in this document.

* 1. Pay by Bank app Integrated Web Merchant Button with Pay by Bank app Popup

Merchants and Distributors can integrate their integrated 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.

## M-COMM Journey

The Merchant Website or App is opened on the same device as the Pay by Bank app CFI App (Pingit). 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 (Pingit) 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 Pingit’ or get the Pay by Bank app Code to pay on another device.
* If this is not a first payment on the device and the Consumer has selected `open Pingit’ from before, the Pay by Bank app enabled CFI App (Pingit) 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.

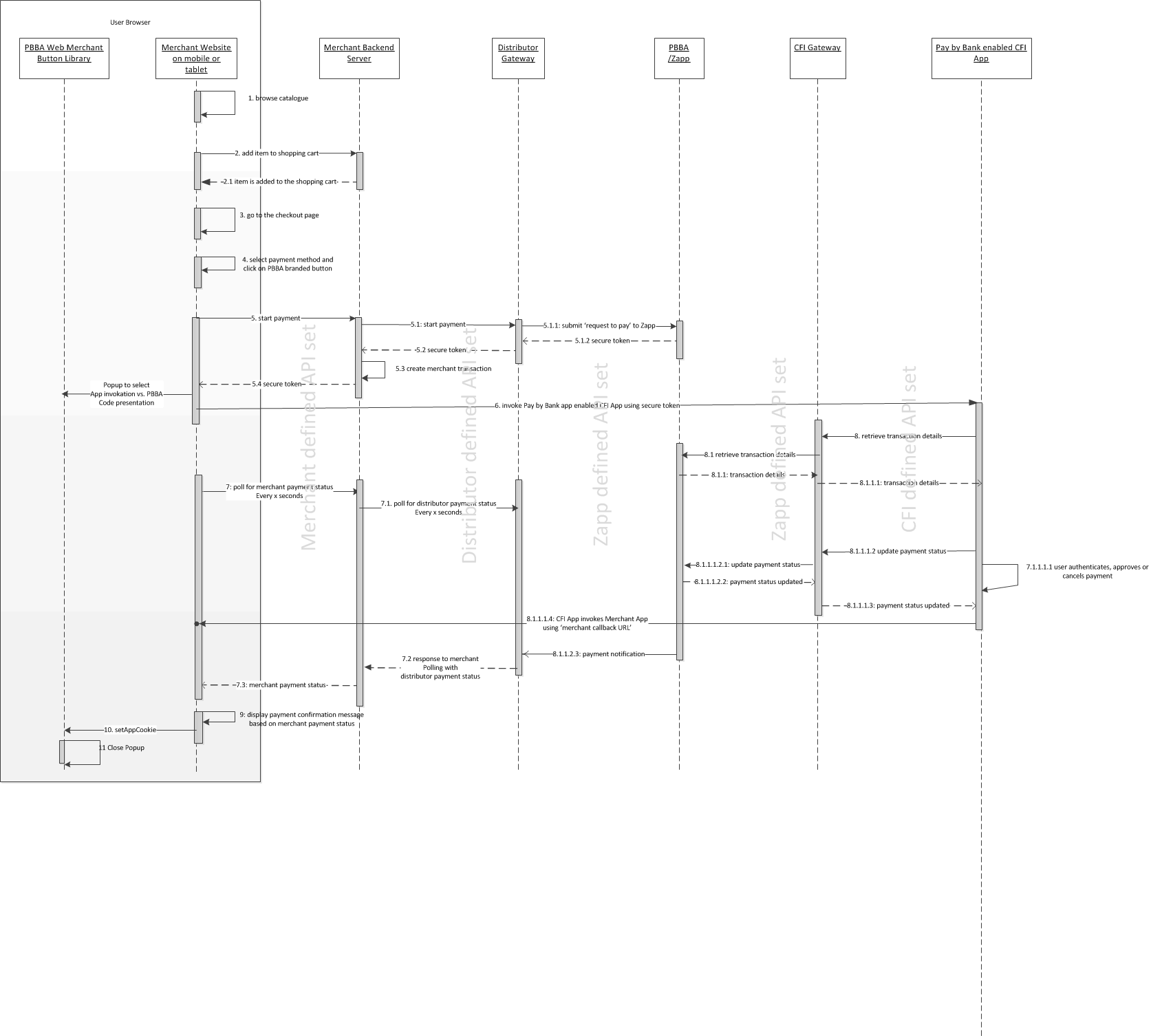


Figure : Interaction between the components of the M-COMM journey

Merchant’s should contact their Distributor to get Distributor API definitions and also any specific implementation changes.

### App Picker

In the case of 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 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.

|  |  |
| --- | --- |
| C:\Users\miklos.sagi\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\Screenshot_20161122-113137.png |  |
| 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 : App Picker – sample screens

## E-COMM – Pay by Bank app Code Journey

The Merchant Website and the Pay by Bank app CFI App (Pingit) 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 (Pingit) 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 (Pingit), 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 a E-COMM PayConnect Journey.
* If the Consumer cancels the PayConnect option in the Pay by Bank app enabled CFI App (Pingit), future Journey will be PBBA Code Journey

1. The PayConnect feature connects a consumer’s browser to the consumer’s Pay by Bank app enabled CFI App (Pingit) on another device

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

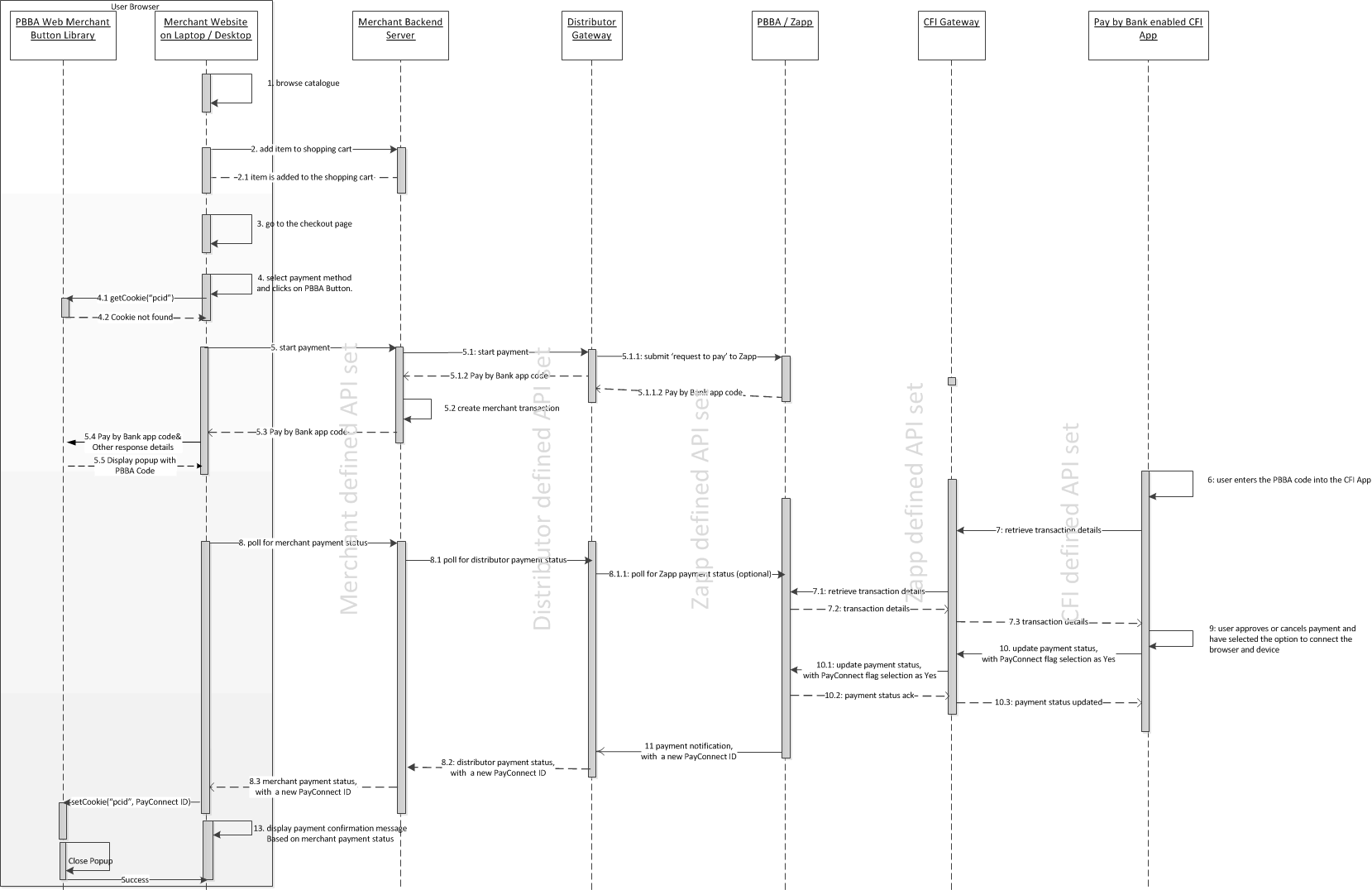


Figure : Interaction between the components of the E-COMM PBBA Code journey

Contact your Distributor for any Distributor specific implementation updates, API definitions or amendments.

## E-COMM – PayConnect Journey

The Merchant Website and the Pay by Bank app CFI App (Pingit) 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 (Pingit).

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 (Pingit) 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 (Pingit) on the device and retrieves the transaction
* The Consumer can approve or cancel the transaction
  1. 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.

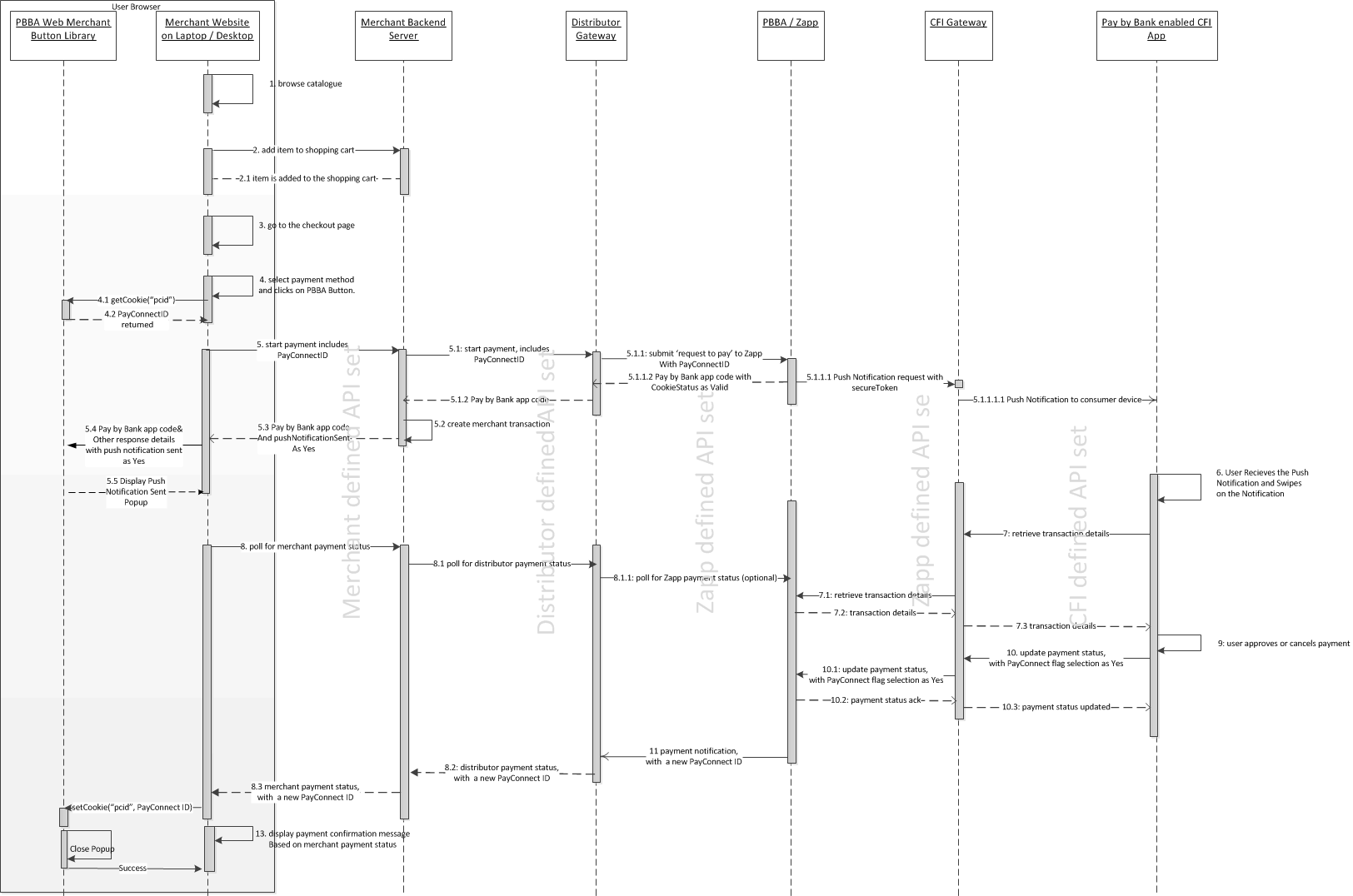


Figure : Interaction between the components of the E-COMM PayConnect journey

Contact your Distributor for any Distributor specific implementation updates, API definitions or amendments.

# Technical Overview

## 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

## Certified Browsers and Devices

Zapp has certified the Web Merchant Button library to work with the following browsers:

| Browser | Version |
| --- | --- |
| Chrome | 44.0+ |
| Firefox | 39.0.3+ |
| IE | 10+ |
| Safari | 10+ |

Table : Certified Browsers and Devices

Zapp has certified the Web Merchant Button library to work with the following mobile devices and operating systems:

1. Landscape orientation is not supported for Web on mobile browsers.

| Android Device Manufacturer | OS Version |
| --- | --- |
| LG Nexus 5 (D821) | Android v22 5.1 |
| Samsung Galaxy S6 or S6 edge (SM-G920F or SM-G925F) | Android v22 5.1 |
| Samsung Galaxy Tab 3 (GT-P5210) | Android V19 4.4 |
| LG Nexus 5 (D821) | Android v22 5.1 |

Table : Compatibility with mobile devices and operating systems (Android)

| Apple Devices | iOS Version |
| --- | --- |
| iPhone 5 | 8+ |
| iPhone 5s | 8+ |
| iPhone 6 | 8+ |
| iPhone 6 Plus | 8+ |
| iPad Air | 8+ |
| iPhone 7 | 8+ |
| iPhone 7 Plus | 8+ |

Table : Compatibility with mobile devices and operating systems (Apple)

| Third Party Component used in PBBA Button | Version |
| --- | --- |
| JQuery | 1.11.3 |

Table : Third party component

Download the latest version from:

| File Name / Version | Hosted On | Version | Download Location |
| --- | --- | --- | --- |
| Web Merchant Button\_2.0.4.zip | SharePoint | 2.0.5 | <https://vocalink.sharepoint.com/sites/zapp/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2fsites%2fzapp%2fShared%20Documents%2fParticipant%20Documentation%2fR%202%2e0%20Final%2fEIS11%20Web%20Merchant%20Button%20Implementation%20Guide%20Code&FolderCTID=0x012000F75770F3FD4DEF4283C5550202EE7AAC> |
| GitHub | GitHub | 2.0.5 | https://github.com/vocalinkzapp/ZappWebMerchantButton-R2/releases/tag/R2.0.5 |

Table : Web Merchant Button Library download locations

## 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.

1. 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.

## 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. It is in [Web Merchant Button Implementation Guide Code](https://vocalink.sharepoint.com/sites/zapp/_layouts/15/guestaccess.aspx?guestaccesstoken=1Ub102vpXHQECZq69NimdJG9YO9TSatgq2VzsY5DSK4%3d&folderid=2_1685d741880634224b60afdb81633ac48&rev=1) folder as a compressed ZIP file. Alternatively you can also clone the project from [GitHub](https://github.com/vocalinkzapp/ZappWebMerchantButton-R2.git). The overall folder structure is represented in Figure 5 below:

1. The text ‘Version Number’ in Figure 5 below represents the actual version number that is displayed, for example, 2.0.5.

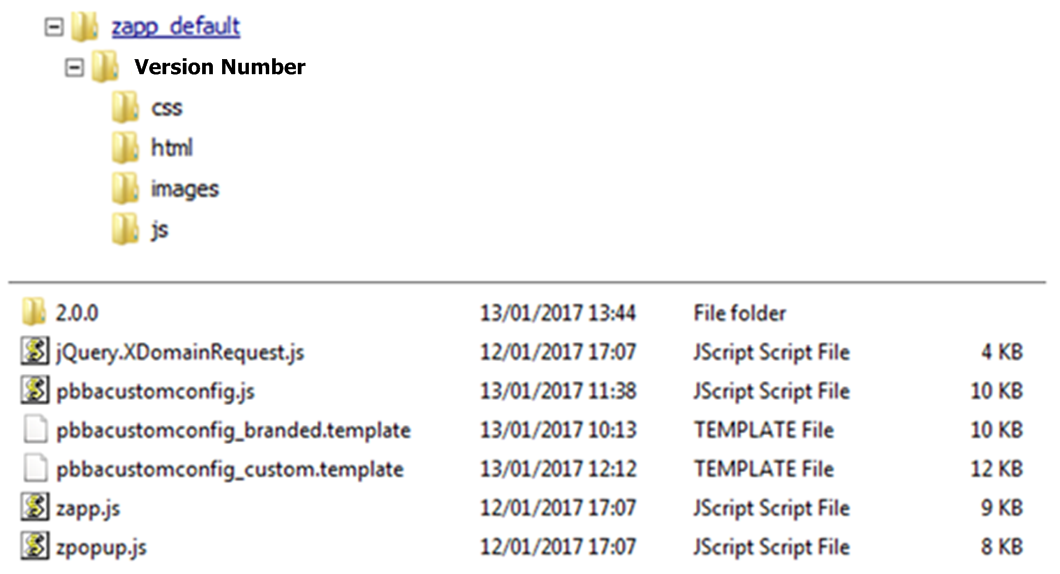


Figure : Branded Pay by Bank app Web Merchant button library structure

## Technical Requirements

### General requirements

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

| Component | Version |
| --- | --- |
| JQuery | 1.11.3 |

Table : 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="http://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

…

</head>

This will import the JQuery plugins in to the project.

1. 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.

### 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 3.7 PBBA Branded Merchant Button Setup.

## Branded Web Merchant Button (Standard Offering)

The Branded PBBA Button is provided by Zapp and is ready to implement. The standard offering 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. |
| * 1. Popup component | This component core focus is the Popup styling and its functions including device specific UI responsiveness. |
| * 1. Cookie management component | This component covers the setting and retrieving of multiple cookies like Has App (`hasApp’), PayConnect (`pcid’) cookies. |

### 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 3.7 Pay by Bank app Branded Merchant Button Setup for detailed implementation instructions.

### 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.

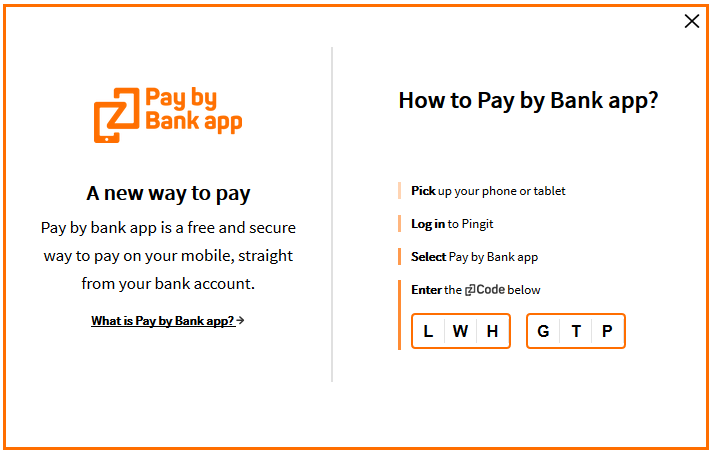


Figure : Popup component

### 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 paybybankapp.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 | [www.paybybank.co.uk](http://www.paybybank.co.uk) | On response to notify | Not available – Basket enhancement |  | To detect whether installed on mobile |
| pcid | Persistent | Third | [www.paybybank.co.uk](http://www.paybybank.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 : 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:

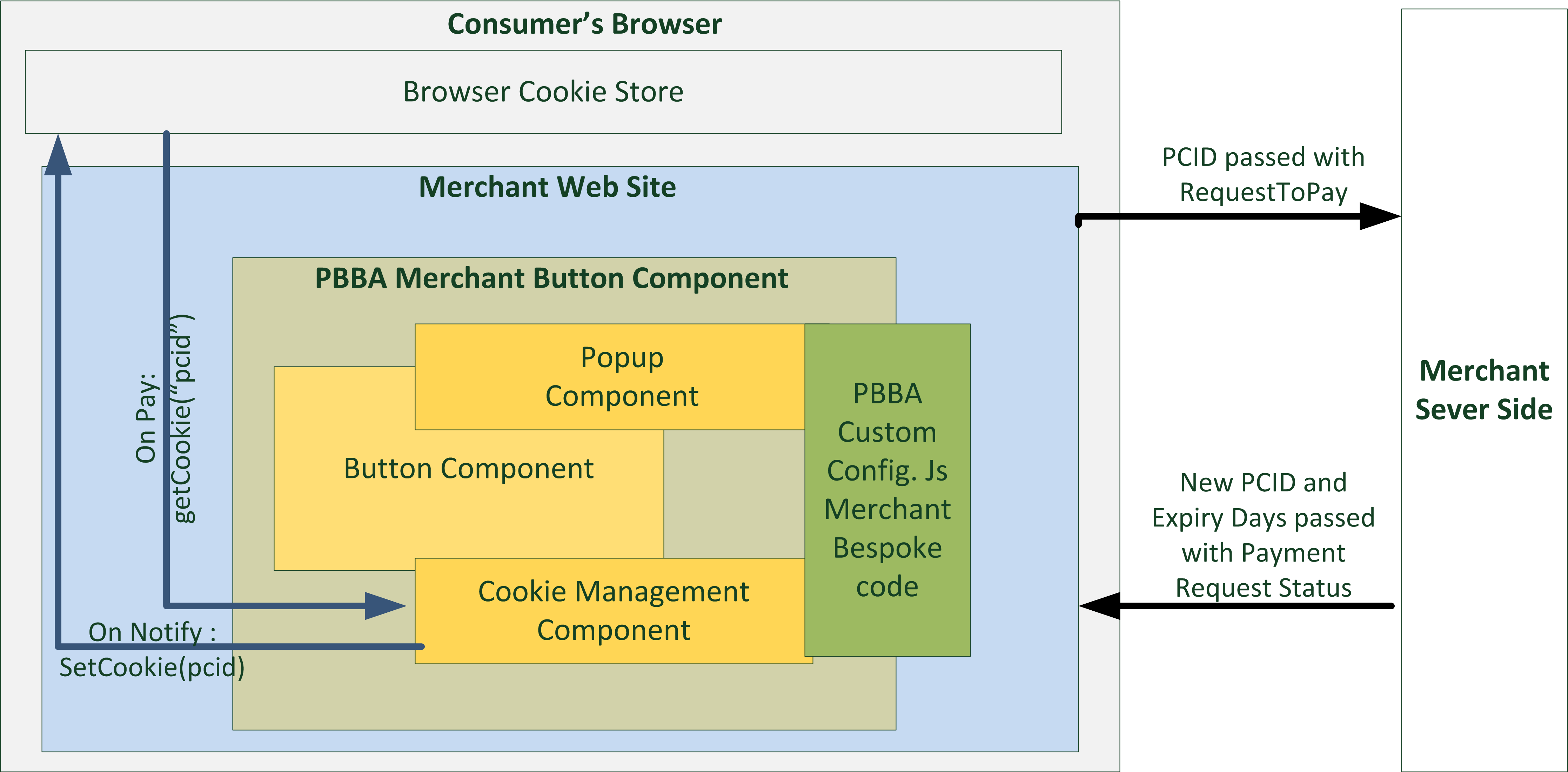


Figure : PayConnect Cookie (pcid)

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

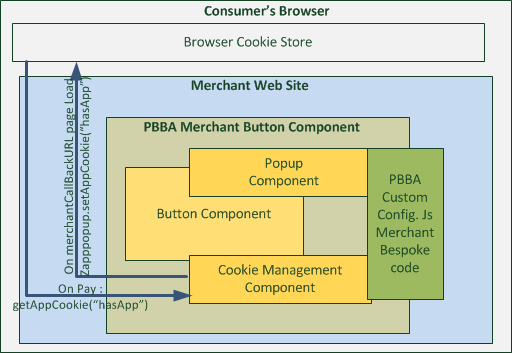


Figure : hasApp Cookie

### 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.

1. All illustration of Merchant technical component/requirement in all the code snippets and examples below are represented in *italic* text.
2. **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>

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

Example:

<script>

**new** zapp.button({});

</script>

* 1. This will not show the Button immediately.
  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 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 **=** "2.0.0";

* 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://www.paybybankapp.co.uk/";

* 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.
  1. To set the PayConnect functionality in the JavaScript file pbbacustomconfig.js, type the following

window.onload = **function**() {

setupPayConnect(cookieManagementUrl, document);

}

* 1. The variable cookieManagementUrl is used here from the Step 3 settings

The one time browser redirect (setupPayConnect()) is through an IFRAME and sets a PayConnect cookie on the domain as per the cookieManagementUrl. Once this cookie is set on the Consumer’s browser, they can use PayConnect.

* 1. 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 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

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

});

5a. The **Pay** method

pay : function(data, callback)

* 1. `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 : 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** 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 |
| --- | --- | --- |
| merchantRequestToPayResponseObject.secureToken | Unique token that identified a Request to Pay | < Consult Distributor Documentation > |
| merchantRequestToPayResponseObject.pbbaCode | A six character code, that identifies a Request to Pay for the duration of retrieval timeout period | < Consult Distributor Documentation > |
| merchantRequestToPayResponseObject.retrievalTimeOutPeriod | 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 > |
| merchantRequestToPayResponseObject.confirmationTimeoutPeriod | 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 > |
| merchantRequestToPayResponseObject.cookieSentStatus | This field is used in the PayConnect journey only, the field confirms if a  payment notification was sent out to consumer, the Popup component of the button shows the appropriate Popup based on this flag | < Consult Distributor Documentation > |
| 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 : 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.

* 1. 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 browser console.

callback(new zapppopup.response.payment({

success : false, // Leave As is

data : MerchantErrorJSONObject

});

5d. The **Notify** method

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

* 1. `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 success flag set to false and invoke the `callback’ function:

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

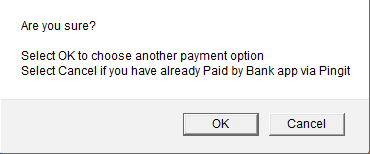
callback(response);

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

|  |
| --- |
| P2.0 S2 - PBBA Code Generated.png |
| After Retrieval period Times out and if the status is still INPROGRESS then the Consumer will see the Popup until Confirmation Time out period |

|  |
| --- |
| P2.0 S2 - PBBA Code Generated.png |
| After Confirmation period times out the polling stops, but the Popup will continue to be shown to the Consumer until they either close the Popup or retry |

* If the Consumer tries to close the Popup while polling is in progress then the following confirmation box will be displayed. The polling and timers (confirmation and retrieval timers) will be paused as long as this Popup is displayed.



* Selecting **OK** will stop the polling process and close the Popup, thereby ending the transaction.
* Selecting **Cancel** will resume the timers and the polling will continue.

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 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 |
| --- | --- | --- |
| 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 : PayConnect cookie setting for Pay Status Authorised

* Merchant’s custom success page handling should follow after setting the PayConnect cookie.

See Appendix A.1.3 PBBA Cookie Management components for more information on 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 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);

* 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>

1. 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.

### Pay by Bank app Branded Merchant Button Sample Code

See Appendix A.2 PBBA Button implementation sample code for a full sample code set for reference.

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

# 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.

## 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">

## 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 Pingit’ 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 (Pingit) 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.

|  |  |
| --- | --- |
| Cookie Management Block Diagram-hasApp |  |

Figure : Mobile App Cookie

### Setting hasApp cookie

1. 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 (Pingit) 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.
2. 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 10 and Figure 11 following.

Consumer selects Pay by Bank app on the same device:

|  |  |  |
| --- | --- | --- |
| * 1. Consumer starts a Pay by Bank app Journey for the first time on a Browser | * 1. Consumer is provided with a selection, as shown below | * 1. The PBBA enabled CFI bank App (Pingit) on the same device is invoked automatically |
|  |  | C:\Chinmay\Web Merchant Button\Documents\Images\Pingit.PNG |
| Consumer clicks on the PBBA Branded Web Merchant Button to make a payment | Consumer selects Pay by Bank app and clicks `Open Pingit’ | On Consumer completion of the payment confirmation decision and when the control from CFI App (Pingit) gets returned to the Merchant Webpage, the `hasApp’ cookie is set on this browser for future PBBA Journey from the browser |

Figure : M-COMM Journey on a mobile browser

1. Consumer selects Pay by Bank app but uses another device to open the Pay by Bank app Enabled CFI App (Pingit):

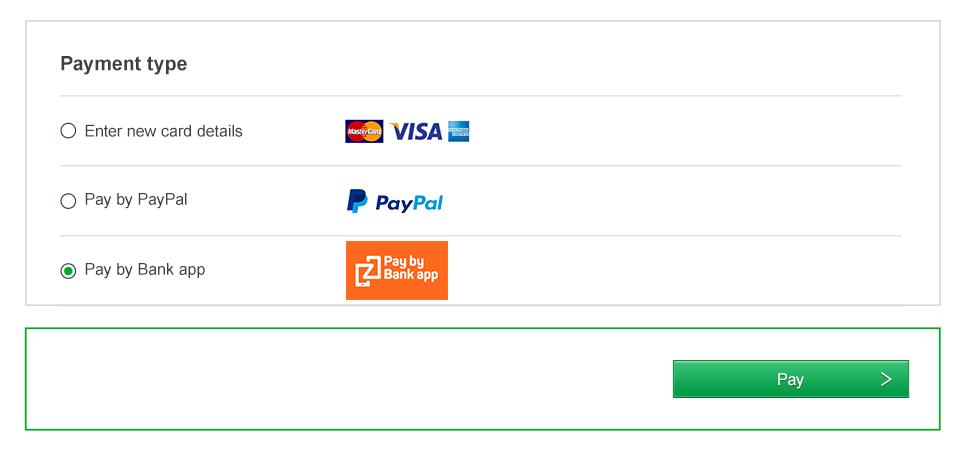
|  |  |  |  |
| --- | --- | --- | --- |
| * 1. Consumer starts a PBBA Journey for the first time on a Browser | * 1. Consumer is provided with a selection, as shown below | * 1. Consumer selects `Get PBBA Code’ in the `Pay with another device’ section of the pop up | * 1. Consumer completes the Journey on another device with a PBBA Enabled CFI App (Pingit) |
|  |  |  | C:\Chinmay\Web Merchant Button\Documents\Images\Pingit.PNG |
| Consumer clicks on the PBBA Branded Web Merchant Button to make a payment | Consumer selects `Get PBBA Code’ in the `Pay with another device’ section of the pop up | Consumer enters the PBBA Code into the PBBA enabled CFI App (Pingit) on another device | The `hasApp’ cookie is not set |

Figure : M-COMM Journey with another device

## 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.

Example:



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.

1. Appendices
   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.

* + 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

* + 1. Co-branded Pay by Bank app Button Images

The property `imageKey’ allows the Merchant to select the image to be used on the Merchant button including a co-branded image.

In order to specify this image, use the `imageKey’ attribute in PBBACustomConfig.js. This attribute takes an Integer as an input. The imageKey attribute defaults to 1, which is the default PBBA branded Pay by Bank app image. In order to use a custom co-branded image, the Merchant must provide the imageKey for their specific image.

The image naming convention for co-branded images is:

* button\_pbba\_<imageKey>.svg, the image has to be in `svg’ format and has to be approved by PBBA/Zapp
* The co-branded images must reside in the co-branding sub-folder within the images folder.
* Pass this property to the zapp.load function as shown below:

var imageKey = 1; // Default Image.

Sample implementation are shown below:

|  |  |
| --- | --- |
| **imageKey = 1 (default)**  Standard orange Pay by Bank app assets on the Pay by Bank app buttons |  |
| **imageKey = 2**  Barclays / Pay by Bank app light  co-branded assets on the Pay by Bank app buttons |  |
| **imageKey = 3**  Barclays / Pay by Bank app dark  co-branded assets on the Pay by Bank app buttons |  |

* 1. Pay by Bank app Button implementation sample code

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

1. 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 = "2.0.0"; // This is the current web merchant button library version. Packaged with button releases

**var** cookieManagementUrl = "https://www.paybybankapp.co.uk/" // Cookie Management, packaged with Button releases.

**var** imageKey = 1; // Default image key*. Could be overridden by mMrchants*

**var** merchantPollInterval = 5000; // Default 5 Seconds. *Could be overridden by merchants* Seconds

// 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** zapppopup.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** zapppopup.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 mMrchant 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** zapppopup.response.notify(*merchantGetPaymentStatusObject*);

\_callback(response);

\_confirmOrder(*merchantGetPaymentStatusObject*);

}

},

error : **function**(*merchantGetPaymentStatusObject*) {

console.log('error');

*merchantGetPaymentStatusObject*.success = **false**;

**var** response = **new** zapppopup.response.notify(*merchantGetPaymentStatusObject*);

\_callback(response);

}

});

},

error : **function**(errors) {

console.log(errors);

alert(errors);

},

cookieManagementUrl: cookieManagementUrl,

imageKey: imageKey,

merchantPollInterval: merchantPollInterval

});

--------------------------------------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 {padding: 10px 0;width: 100%;}

1. 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 your wrap this Button within a <DIV> tag to control the height and width of this Button.
   1. Changing the look and feel of the Button

Merchants may wish to change the colour and look and feel of the Pay by Bank app Branded Web Merchant Button. Zapp provides a PHP based image conversion utility. The utility and its supporting documentation can be made available upon request.

* 1. 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.

* + 1. Remove all connections Mobile Banking Application (Pingit) Consumer function

Consumers have an option within the Mobile Banking application (Pingit) 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 returned as invalid. The Consumer will be re-offered the opportunity to connect as part of the payment confirmation journey.

* + 1. DDoS protection

To prevent a potential Pay by Bank app DDoS threat, an individual cookie token can only be submitted three 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 three 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.

* 1. 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.

* + 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.

1. 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,

…

});

* 1. 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()

* 1. Change Http GET to Http POST:

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

* 1. 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 impacts this URL/ URI and not any other area that requires caching.
  1. 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.