**PBBA Integrated 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. |
| 2.3 Final Draft | 25/07/2016 | * 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 | 11/10/2016 | * 1. Cleaned up examples to remove unwanted fields.   2. Updated the PacyConnect URL in the examples.   3. Added steps to continue polling upon receipt of a payment not confirmed status. |
| 2.5 | 21/10/2016 | * 1. Added the setCookie() description to Appendix A.   2. Added support for Safari |
| 2.6 | 25/10/2016 | * 1. 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 | 22/11/2016 | * 1. Added details on listeners to be registered.   2. Added details on functions to be invoked to stop the timers and close the popup.   3. Added details on getting the mobile cookie for M-COMM journey.   4. Updated example.   5. Updated sequence diagrams. |
| 2.9 | 23/11/2016 | Final Draft |
| 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 | Rename `Custom’ to `Integrated’. |
| 3.3 | 30/03/2017 | Update re polling for current status. |
| 4.0 | 00-03-2017 | Updates to this version |

Contents

[1 About this document 8](#_Toc479674131)

[1.1 Introduction 8](#_Toc479674132)

[1.2 Audience 8](#_Toc479674133)

[1.3 Scope 8](#_Toc479674134)

[1.4 Document conventions 8](#_Toc479674135)

[1.5 Associated documents 8](#_Toc479674136)

[2 Functional overview 9](#_Toc479674137)

[2.1 Introduction 9](#_Toc479674138)

[2.2 M-COMM Journey 9](#_Toc479674139)

[2.2.1 App Picker 11](#_Toc479674140)

[2.3 E-COMM – Pay by Bank app Code Journey 12](#_Toc479674141)

[2.4 E-COMM – PayConnect Journey 14](#_Toc479674142)

[3 Technical Overview 16](#_Toc479674143)

[3.1 Introduction 16](#_Toc479674144)

[3.2 Certified Browsers and Devices 16](#_Toc479674145)

[3.3 Hosting Options 17](#_Toc479674146)

[3.4 Pay by Bank app Web Merchant Button library structure 18](#_Toc479674147)

[3.5 Technical Requirements 19](#_Toc479674148)

[3.5.1 General requirements 19](#_Toc479674149)

[3.5.2 Library hosting requirements 19](#_Toc479674150)

[3.6 Integrated Web Merchant Button with PBBA Popup 20](#_Toc479674151)

[3.6.1 Popup component 21](#_Toc479674152)

[3.6.2 Cookie management component 22](#_Toc479674153)

[3.6.3 Integrated Web Merchant Button Setup 24](#_Toc479674154)

[3.6.4 Integrated Web Merchant Button Sample Code 33](#_Toc479674155)

[4 Additional considerations for M-COMM 34](#_Toc479674156)

[4.1 Prerequisites for M-COMM 34](#_Toc479674157)

[4.2 Mobile App Cookie – retaining PBBA enabled Bank App selection 34](#_Toc479674158)

[4.2.1 Setting hasApp cookie 34](#_Toc479674159)

[4.2.2 Getting hasApp cookie 38](#_Toc479674160)

[A Appendices 39](#_Toc479674161)

[A.1 Merchant Configurable Properties 39](#_Toc479674162)

[A.1.1 Merchant Poll Intervals 39](#_Toc479674163)

[A.1.2 Pay by Bank app Cookie Management Component 39](#_Toc479674164)

[A.2 Integrated Web Merchant Button implementation sample code 41](#_Toc479674165)

[A.2.1 Merchant’s HTML file 41](#_Toc479674166)

[A.2.2 Changes to the custom configuration file - pbbacustomconfig.js 42](#_Toc479674167)

[A.3 Additional Cookie management Information 47](#_Toc479674168)

[A.3.1 Remove all connections Mobile Banking Application (Pingit) Consumer function 47](#_Toc479674169)

[A.3.2 DDoS protection 47](#_Toc479674170)

Tables

[Table 1: Certified Browsers and Devices 16](#_Toc479163010)

[Table 2: Compatibility with mobile devices and operating systems (Android) 16](#_Toc479163011)

[Table 3: Compatibility with mobile devices and operating systems (Apple) 17](#_Toc479163012)

[Table 4: Third party component 17](#_Toc479163013)

[Table 5: Web Merchant Button Library download locations 17](#_Toc479163014)

[Table 6: Web Merchant Button library – Setup requirements 19](#_Toc479163015)

[Table 7: Cookie management component 22](#_Toc479163016)

[Table 8: PayConnectID Mapping to Distributor API 27](#_Toc479163017)

[Table 9: Pay Method - Successful Response to RTP 29](#_Toc479163018)

[Table 10: PayConnect cookie setting for Pay Status Authorised 32](#_Toc479163019)

[Table 11: PayConnectID Set Cookie function 40](#_Toc479163020)

Figures

[Figure 1: Interaction between the components of the M-COMM journey 10](#_Toc479163021)

[Figure 2: App Picker – sample screens 11](#_Toc479163022)

[Figure 3: Interaction between the components of the E-COMM PBBA Code journey 13](#_Toc479163023)

[Figure 4: Interaction between the components of the E-COMM PayConnect journey 15](#_Toc479163024)

[Figure 5: Pay by Bank app Web Merchant button library structure 18](#_Toc479163025)

[Figure 6: Integrated Web Merchant Button with PBBA Popup 20](#_Toc479163026)

[Figure 7: Popup component 21](#_Toc479163027)

[Figure 8: PayConnect Cookie (pcid) 23](#_Toc479163028)

[Figure 9: hasApp Cookie 23](#_Toc479163029)

[Figure 10: Mobile App Cookie 34](#_Toc479163030)

[Figure 11: M-COMM Journey on a mobile browser 36](#_Toc479163031)

[Figure 12: M-COMM Journey with another device 37](#_Toc479163032)

# About this document

## 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.

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 Integrated 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 Branded Web Merchant Button Implementation Guide*
* *Zapp Glossary*

# Functional overview

## 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.

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 covered 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 Branded 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 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 their 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.

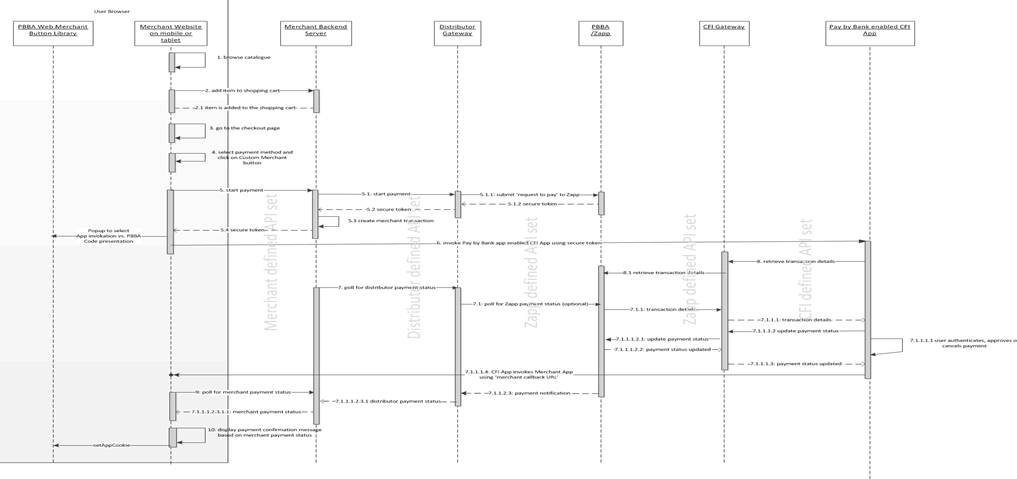


Figure 1: Interaction between the components of the M-COMM journey

### App Picker

In the case 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.

|  |  |
| --- | --- |
| 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 2: App Picker – sample screens

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

## 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
* The Merchant Website displays a six letter code to the Consumer
* The Consumer opens a Pay by Bank app enabled CFI App (Pingit) 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 (Pingit), 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 (Pingit), future Journey will require a Pay by Bank app Code

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.

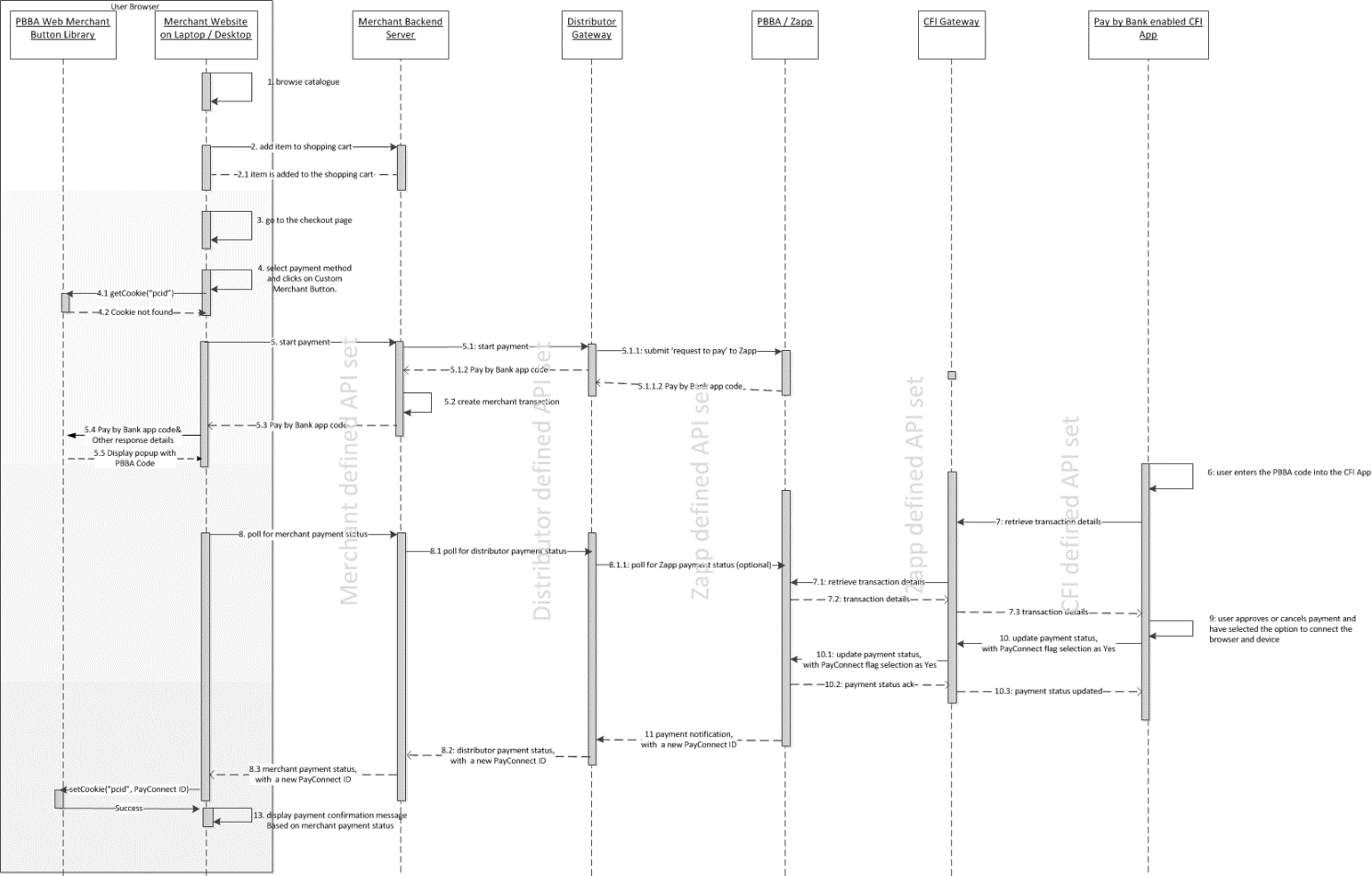


Figure 3: Interaction between the components of the E-COMM PBBA Code journey

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

## 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 Pay by Bank app code Journey with a 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 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 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 taps 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.
* 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.

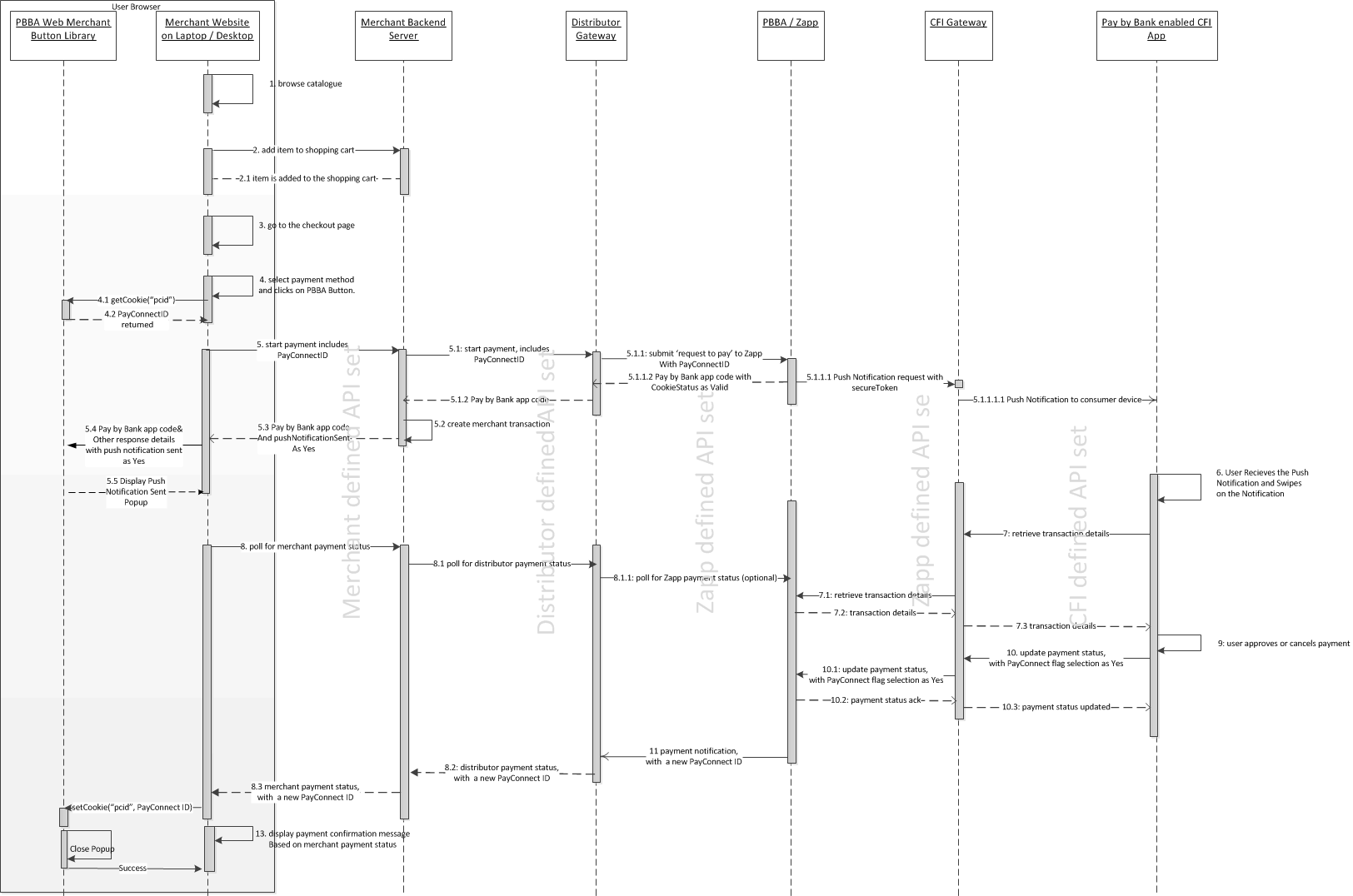


Figure 4: Interaction between the components of the E-COMM PayConnect journey

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

# Technical Overview

## 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

## 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 1: 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 |

Table 2: 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 3: Compatibility with mobile devices and operating systems (Apple)

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

Table 4: 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 5: 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.

## 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.

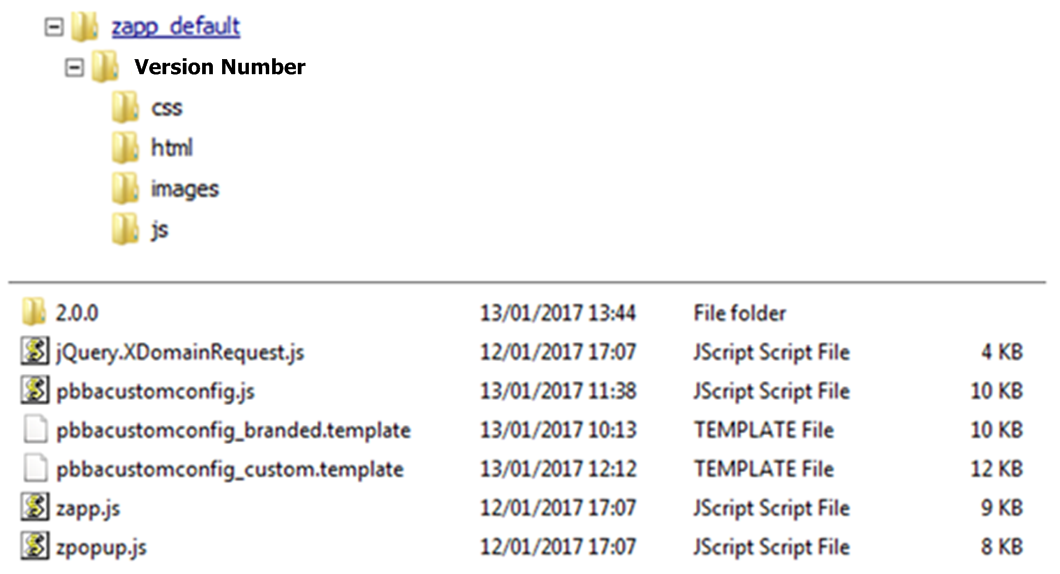


Figure 5: Pay by Bank app Web Merchant button library structure

## 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 6: 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 3.2 Certified Browsers and Devices.

## Integrated Web Merchant Button with PBBA Popup

The Merchant can integrate the Pay by Bank app Popup 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 like Visa, MasterCard, PayPal etc. 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 popup:

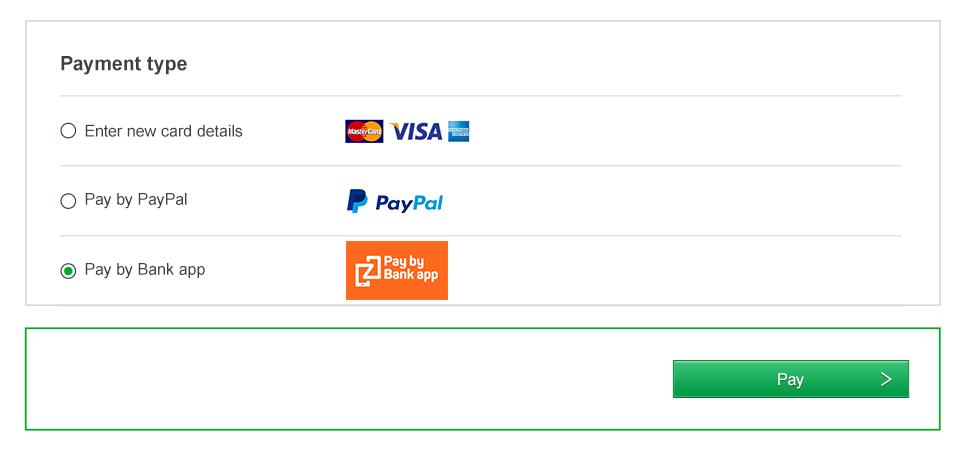


Figure 6: Integrated Web Merchant Button with PBBA Popup

1. There are only two components to be implemented for the Integrated Web Merchant button:
   1. Popup component - This component core focus is the popup 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, PayConnect cookies.

### 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.

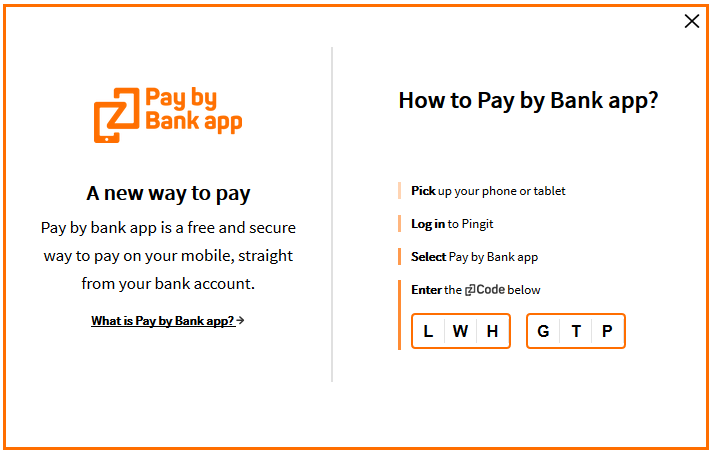


Figure 7: Popup component

### Cookie management component

This component covers the setting and retrieving of multiple cookies as listed in Table 7 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 7: 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:

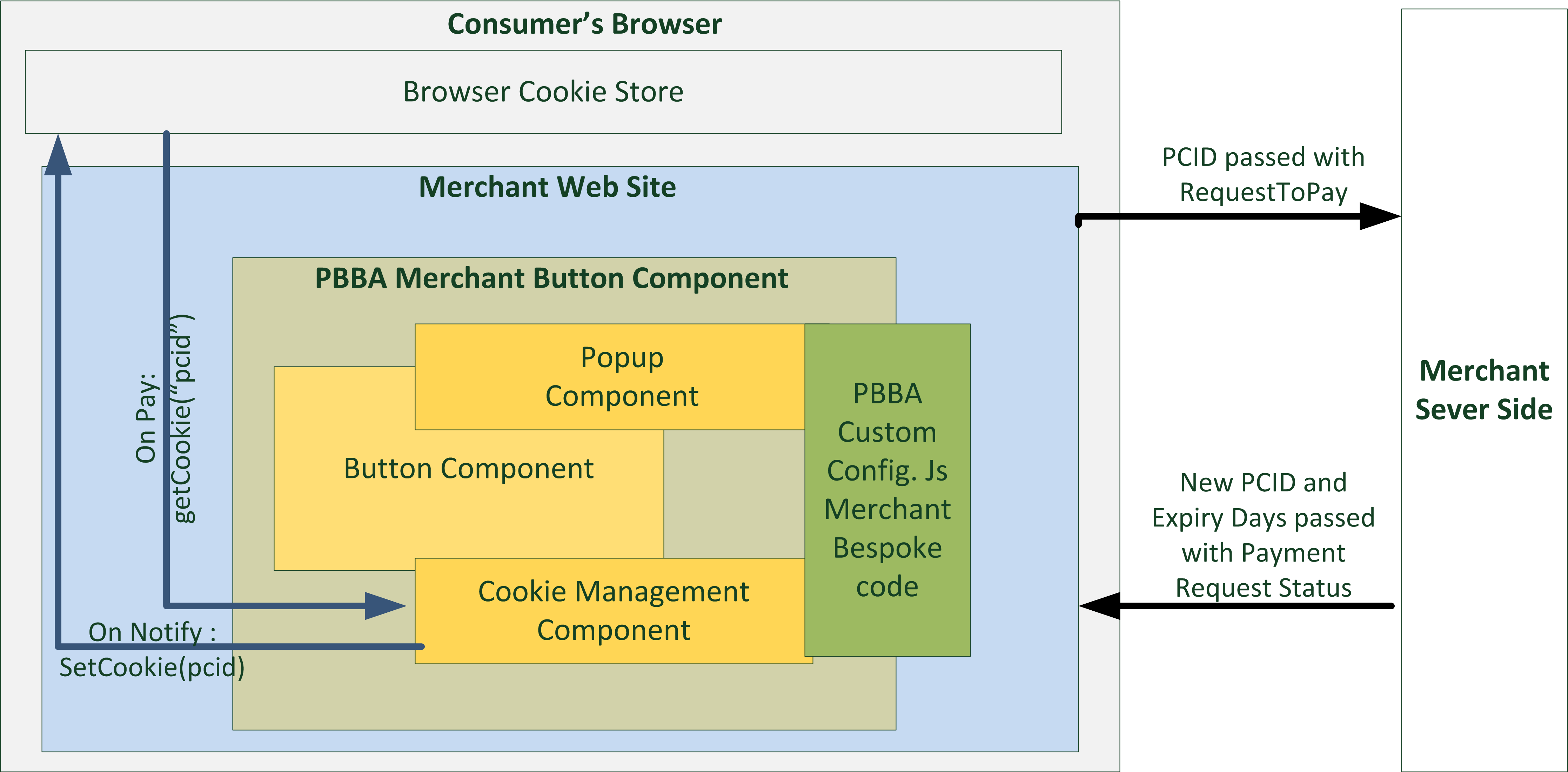


Figure 8: PayConnect Cookie (pcid)

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

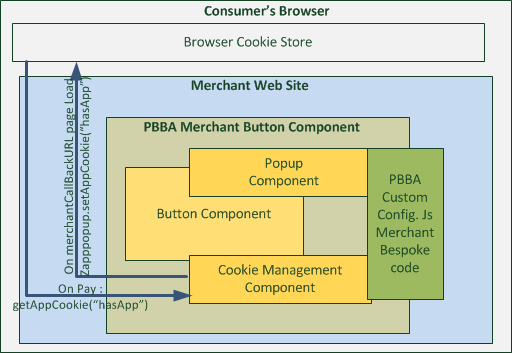


Figure 9: hasApp Cookie

The Setup section of this document (see section 3.6.3 Integrated Web Merchant Button Setup) provides integration details of the popup component with the Integrated web Merchant Button.

### 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 Popup and PBBA Cookie Management component with the Integrated Merchant Button.

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/zpopup.js"*></script>

</head>

</html>

* 1. 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 zpopup.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 **=** "2.0.5";

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

* 1. Initialise the popup engine:

zapppopup.load(zappVersion, {

cookieManagementUrl: cookieManagementUrl

} );

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

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

setupPayConnect(cookieManagementUrl, document);

}

* 1. Register the following listeners.

These listeners will help capture various events like timeout, popup 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 Merrchant.

}

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.

zapppopup.\_stopTimers();

zapppopup.\_removePopup(true);

}

}

* 1. 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 8: PayConnectID Mapping to Distributor API

7a. Show the Pay by Bank app Code on the popup.

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 popup

* 1. Follow this 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 9: 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 popup:

if (merchantRequestToPayResponseObject.cookieSentStatus) {

response.pcid = zapppopup.getCookie('pcid');}

|  |
| --- |
| The notificationSent value was true |
| The notificationSent value was false |

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 popup 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 popup 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);

* 1. `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 popup:

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 popup specific functions and variables.

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 10: PayConnect cookie setting for Pay Status Authorised

9b. After the successful response is received.

Remember to clear the timers and close the popup by calling the following functions:

zapppopup.\_stopTimers(); // This will clear the popup timers

zapppopup.\_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:

zapppopup.\_stopTimers(); // This will clear the popup timers

zapppopup.\_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.

### 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.

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 10: 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 11 and Figure 12 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 selects the Pay by Bank app option and clicks the Integrated Web Merchant Button (Pay) 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 11: 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 selects the Pay by Bank app option and clicks the Integrated Web Merchant Button (Pay) 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 12: M-COMM Journey with another device

### 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 6b 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 (zapppopup.getCookie("hasApp")){

zapppopup.\_invokeAppWithResponse(clickedButton, response);

// Start polling for payment notififcation

return;

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 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 zapppopup.load function as mentioned below:

var merchantPollInterval = 10000; // 10 seconds

* + 1. Pay by Bank app Cookie Management Component

The PBBA Popup sets multiple cookies to provide a rich user experience. Some are Merchant domain specific cookies and others are paybybankapp.co.uk domain specific third party cookies. There are persistent cookies and session cookies which are detailed in Table 7: Cookie management component.

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

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

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

1. 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-1.11.3.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 |
| --- | --- | --- |
| 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 11: 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.

* 1. 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:

1. 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 popup. 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.

* + 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-1.11.3.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>

* + 1. Changes to the custom configuration file - pbbacustomconfig.js

--------------------------------------------START ---------------------------------------------

jQuery.support.cors = true;

if (!window.console) console = {log: function() {}};

var zappVersion = "2.0.5";

var cookieManagementUrl = "https://www.paybybankapp.co.uk/"

var merchantPollInterval = 5000;

*var clickedButton = null;*

zapppopup.load(zappVersion, {

cookieManagementUrl: cookieManagementUrl

} );

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.\_addPopup(*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() {

zapppopup.\_stopTimers(); // Stop the timers

zapppopup.\_removePopup(true); // Remove the popup

}

});

}

--------------------------------------FINISH---------------------------------------------

* 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 PBBA. 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 PBBA 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 PBBA 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 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 PBBA 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