## #ZappMerchantLibrary SDK integration documentation

ZappMerchantLibrary is for Android Merchant applications to integrate Pay by Bank app payments.

## Build from source

### Since the debug and release versions of the Zapp Merchant Android Library .aar files are provided with the release, building the library from source is optional

Gradle (on Mac):

- open a Terminal Window  
 - change directory to be the root folder of "Sample" (Pay by Bank app - Demo)  
 Android Application Project  
 - execute command "./gradlew :zapp-merchant-library:assembleRelease"  
 - the release Android Archive of the library (zapp-merchant-library-release.aar)  
 is located in the ./zapp-merchant-library/build/outputs/aar folder

Gradle (on Windows):

- open a Command Prompt Window  
 - change directory to be the root folder of "Sample" (Pay by Bank app - Demo)  
 Android Application Project  
 - execute command "gradlew :zapp-merchant-library:assembleRelease"  
 - the release Android Archive of the library (zapp-merchant-library-release.aar)  
 is located in the zapp-merchant-library\build\outputs\aar directory

### Import the debug or release version Zapp Merchant Library .aar to the Merchant Application Project. This is the recommended way of using the library

Android Studio:

- select "File" -> "New" -> "New Module"  
 - select "Import .JAR or .AAR package"  
 - click "Next"  
 - browse "zapp-merchant-library-release.aar" file in to the "File name" field  
 - enter "zapp-merchant-libarary-1.0.0" in to the "Subproject name" field  
 - click "Finish"

## Add the source

### Adding the source of the Zapp Merchant Library to the Merchant Application Project is not recommended. However, it may be useful to do this during the development phase

Android Studio:

- select "File" -> "New" -> "Import Module"  
 - browse "zapp-merchant-library" subfolder of the Sample (Pay by Bank app - demo) application in to the "Source directory" field  
 - observe that the "zapp-core-library" appears in the additional required modules list  
 - click "Finish"

The source code structure

* com.zapp.library.merchant
  + exception - predefined errors
  + model - payment request builder
    - callback - callback listeners for delegate calls
  + service
    - delegate - the delegate interfaces to be implemented by the Merchant Application
    - impl - Zapp merchant service implementation
    - provider - Zapp merchant service interface
  + ui - Pay by Bank app popup activity
    - fragment - Pay by Bank app popup fragments (mcomm, ecomm, error)
  + util - utility classes
  + view - Zapp custom widgets and Pay by Bank app button

## Usage

### Add Pay by Bank App Button To View Layout

Add the following to your layout resource:

<com.zapp.library.merchant.view.PBBAButton  
 android:id="@+id/pbba\_button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />

### Create a Zapp Payment Request

There are several types of possible payments. Here are examples of how to create payment request for "Quick Payment", "Paying a bill" or "SMB" journey More examples are present in the Sample (Pay by Bank app - Demo) Application source code.

#### Quick Payment

final CurrencyAmount amount = new CurrencyAmount(1509, CurrencyAmount.POUNDS); //£15.09  
  
 final Address merchantAddress = new Address("200 Queen Victoria Street", /\*line2\*/ null, /\*line3\*/ null, /\*line4\*/ null, /\*line5\*/ null, /\*line6\*/ null,  
 /\* postCode \*/ null, Address.UK);  
 //logoUrl is null because it is set in Zapp Core already  
 final Merchant merchant = new Merchant("000001", "ZappMerchantDemo", "merchantdemo@zapp.co.uk", /\*phone\*/ null, merchantAddress,  
 Uri.parse("https://play.google.com/store/apps/details?id=merchant.application.package").toString(), /\* logoUrl \*/ null);  
  
 final PaymentRequestBuilder paymentRequestBuilder = new PaymentRequestBuilder();  
 paymentRequestBuilder  
 .withPaymentType(PaymentType.INSTANT\_PAYMENT)  
 .withCheckoutType(CheckoutType.QUICK)  
 .withDeliveryType(DeliveryType.ADDRESS)  
 .withAmount(amount)  
 .withMerchant(merchant)  
 .withMerchantCallbackUrl(getString(R.string.app\_scheme));  
  
 final PaymentRequest paymentRequest = paymentRequestBuilder.build();

#### Paying a bill

final CurrencyAmount amount = new CurrencyAmount(1825, CurrencyAmount.POUNDS); //£18.25  
  
 final Date periodFrom = DATE\_FORMATTER.parse("2015-08-01");  
 final Date periodTo = DATE\_FORMATTER.parse("2015-08-30");  
 final BillDetails billDetails = new BillDetails("8500 222 66138", "Y6557802", periodFrom, periodTo);  
  
 final Address userAddress = new Address("10 Downing Street", /\*line2\*/ "Westminster", /\*line3\*/ "London", /\*line4\*/ null, /\*line5\*/ null, /\*line6\*/ null,  
 /\*postCode\*/ "SW1A 2AA", Address.UK);  
 final User user = new User("John", "Smith", /\*middle name\*/ null, /\*title\*/ null, /\*email\*/ null, /\*phone\*/ null);  
  
 final Address merchantAddress = new Address("200 Queen Victoria Street", /\*line2\*/ null, /\*line3\*/ null, /\*line4\*/ null, /\*line5\*/ null, /\*line6\*/ null,  
 /\*postCode\*/ null, Address.UK);  
 //logoUrl is null because it is set in Zapp Core already  
 final Merchant merchant = new Merchant("000001", "ZappMerchantDemo", "merchantdemo@zapp.co.uk", /\*phone\*/ null, merchantAddress,  
 Uri.parse("https://play.google.com/store/apps/details?id=merchant.application.package").toString(), /\* logoUrl \*/ null);  
  
 final PaymentRequestBuilder paymentRequestBuilder = new PaymentRequestBuilder();  
 paymentRequestBuilder  
 .withPaymentType(PaymentType.BILL\_PAY)  
 .withCheckoutType(CheckoutType.NORMAL)  
 .withDeliveryType(DeliveryType.ADDRESS)  
 .withAmount(amount)  
 .withAddress(userAddress)  
 .withUser(user)  
 .withMerchant(merchant)  
 .withBillDetails(billDetails)  
 .withMerchantCallbackUrl(getString(R.string.app\_scheme));  
  
 final PaymentRequest paymentRequest = paymentRequestBuilder.build();

#### SMB Journey

final CurrencyAmount amount = new CurrencyAmount(4800, CurrencyAmount.POUNDS); //£48.00  
  
 final Address merchantAddress = new Address("200 Queen Victoria Street", /\*line2\*/ null, /\*line3\*/ null, /\*line4\*/ null, /\*line5\*/ null, /\*line6\*/ null,  
 /\*postCode\*/ null, Address.UK);  
 //logoUrl is null because it is set in Zapp Core already  
 final Merchant merchant = new Merchant("000001", "ZappMerchantDemo", "merchantdemo@zapp.co.uk", /\*phone\*/ null, merchantAddress,  
 Uri.parse("https://play.google.com/store/apps/details?id=merchant.application.package").toString(), /\* logoUrl \*/ null);  
  
 final PaymentRequestBuilder paymentRequestBuilder = new PaymentRequestBuilder();  
 paymentRequestBuilder  
 .withPaymentType(PaymentType.SMB)  
 .withCheckoutType(CheckoutType.NORMAL)  
 .withDeliveryType(DeliveryType.ADDRESS)  
 .withAmount(amount)  
 .withMerchant(merchant);  
  
 final PaymentRequest paymentRequest = paymentRequestBuilder.build();

### Initialize Zapp Merchant Service

Initialize the delegates:

final IMerchantNetworkServiceDelegate serviceDelegate = new MerchantNetworkServiceDelegateImpl(context);  
 final ZappMerchantUIDelegate uiDelegate = new MerchantUIDelegateImpl(context);

Pass the delegates to the Zapp Merchant Service

final IMerchantService merchantService = ZappMerchantServiceFactory.createMerchantService(serviceDelegate, uiDelegate);

Implement the **IMerchantNetworkServiceDelegate** and **ZappMerchantUIDelegate** in order to communicate with the Merchant in-house system to enable the user to see the payment status in the Merchant application.

#### Delegates Implementation

The Merchant application should implement **IMerchantNetworkServiceDelegate** and extend the **ZappMerchantUIDelegate** which are used by the Zapp Merchant Library in order to communicate with the user and the backend.

##### IMerchantNetworkServiceDelegate:

/\*\*  
 \* Initiates a payment using the provided {@link com.zapp.core.Transaction} details  
 \* which describes all the details about a payment, like {@link com.zapp.core.PaymentType PaymentType}  
 \* or {@link com.zapp.core.BillDetails BillDetails}.  
 \* <br>  
 \* Implementers are completely responsible for handling network or DB calls. <br>  
 \* Implementation can execute blocking calls (network/DB calls) in current Thread or have an  
 \* asynchronous execution in place.  
 \* <br>  
 \* This method is invoked only after a successful validation of payment details and a new RTP(request-to-pay)  
 \* is to be created.  
 \* <br>  
 \* Caller should be notified through {@link com.zapp.library.merchant.model.callback.OnResponseListener responseListener}  
 \* in case of success or failure.  
 \*  
 \* @param paymentRequest All details required to initiate a merchant payment journey, like  
 \* {@link com.zapp.core.PaymentType PaymentType}  
 \* or {@link com.zapp.core.BillDetails BillDetails}.  
 \* You can use {@link com.zapp.library.merchant.model.PaymentRequestBuilder} to  
 \* build a {@link PaymentRequest} object you need.  
 \* @param onResponseListener {@link OnResponseListener OnResponseListener}  
 \* providing callbacks for success and error scenarios while initiating  
 \* an RTP using the provided {@link com.zapp.core.Transaction} details or any  
 \* network-related {@link com.zapp.library.merchant.exception.Error error}.  
 \*/  
void initiatePayment(PaymentRequest paymentRequest, OnResponseListener<Transaction> onResponseListener);  
  
/\*\*  
 \* Checks a payment status using the provided {@link com.zapp.core.Transaction} details  
 \* which should contain, at least, a non-null {@link com.zapp.core.TransactionId transactionId}  
 \* containing an <code>aptrId</code> which uniquely identifies a {@link com.zapp.core.Transaction}.  
 \* <br>  
 \* Implementers are completely responsible for handling network or DB calls. <br>  
 \* Implementation can execute blocking calls (network/DB calls) in current Thread or have an  
 \* asynchronous execution in place.  
 \* <br>  
 \* Caller should be notified through {@link com.zapp.library.merchant.model.callback.OnResponseListener responseListener}  
 \* in case of success or failure.  
 \*  
 \* @param transaction Payment details containing a non-null {@link com.zapp.core.TransactionId transactionId}  
 \* containing an <code>aptrID</code> which uniquely identifies a  
 \* pending {@link com.zapp.core.Transaction}.  
 \* <br>  
 \* @param onResponseListener {@link OnResponseListener OnResponseListener}  
 \* providing callbacks for success and error scenarios while checking  
 \* payment status using the provided <code>aptrID</code> or any  
 \* network-related {@link com.zapp.library.merchant.exception.Error error}.  
 \* If an error with errorCode  
 \* {@link com.zapp.library.merchant.exception.ErrorType#PAYMENT\_NOT\_CONFIRMED ErrorType#PAYMENT\_NOT\_CONFIRMED}  
 \* is received on failure, then merchant library automatically send  
 \* another call for checking the payment status using the current method.  
 \*/  
void notifyMerchantPayment(Transaction transaction, OnResponseListener<Transaction> onResponseListener);  
  
/\*\*  
 \* Retrieves the payment status using the merchantId, aptId and aptrId wrapped into the provided payment object.  
 \* <br>  
 \* <br>  
 \* Caller should be notified through {@link com.zapp.library.merchant.model.callback.OnResponseListener responseListener}  
 \* in case of success or failure.  
 \*  
 \* @param transaction Transaction that contains the merchantId, aptId and aptrId  
 \* @param responseListener {@link OnResponseListener OnResponseListener}  
 \* providing callbacks for success and error scenarios while checking  
 \* payment status using the provided <code>aptrID</code> or any  
 \* network-related {@link com.zapp.library.merchant.exception.Error error}.  
 \*/  
void getPaymentStatus(Transaction transaction, OnResponseListener<TransactionStatus> responseListener);

##### ZappMerchantUIDelegate:

/\*\*  
 \* Shows a payment confirmation screen to the user with the status whether it has been successful or rejected.  
 \* Also it allows the merchant app to display the payment details to the user.  
 \*  
 \* @param transaction Transaction details to be displayed.  
 \* @param error The error message in case that payment has failed.  
 \*/  
void showPaymentConfirmationScreen(Transaction transaction, com.zapp.library.merchant.exception.Error error);  
  
/\*\*  
 \* Shows the screen with the Pay by Bank app code to the user for Small-Micro Business (SMB) transactions.  
 \*  
 \* @param transaction The SMB payment transaction details.  
 \*/  
void showSMBScreen(Transaction transaction);

### Sample implementations

1. "Merchant Demo" implementation which gives and example of how to use the SDK is at uk.co.zapp.samplezappmerchantapp.
2. Sample implementation of the **IMerchantNetworkServiceDelegate** is at ***uk.co.zapp.samplezappmerchantapp.network.delegate / MerchantNetworkServiceDelegateImpl.java***.
3. Sample implementation of the ***ZappMerchantUIDelegate*** is at ***uk.co.zapp.samplezappmerchantapp.ui / MerchantUIDelegateImpl.java***.