WePay Android SDK

3.0.0

Generated by Doxygen 1.8.12

Contents

1 Getting Started **2**

2 Class Index **12**

2.1 Class List 12

3 Class Documentation **13**

3.1 CardReaderHandler.ApplicationSelectionCallback Interface Reference 13

3.1.1 Detailed Description 13

3.1.2 Member Function Documentation 13

3.2 AuthorizationHandler Interface Reference 13

3.2.1 Detailed Description 13

3.2.2 Member Function Documentation 14

3.3 BatteryLevelHandler Interface Reference 15

3.3.1 Detailed Description 15

3.3.2 Member Function Documentation 15

3.4 CalibrationHandler Interface Reference 16

3.4.1 Detailed Description 16

3.4.2 Member Function Documentation 16

3.5 CalibrationParameters Class Reference 17

3.5.1 Detailed Description 17

3.6 CalibrationResult Enum Reference 17

3.6.1 Detailed Description 17

3.6.2 Member Data Documentation 17

3.7 CardReaderHandler.CardReaderEmailCallback Interface Reference 18

3.7.1 Detailed Description 18

3.7.2 Member Function Documentation 18

3.8 CardReaderHandler Interface Reference 18

3.8.1 Detailed Description 19

3.8.2 Member Function Documentation . . . 19
3.9 CardReaderHandler.CardReaderResetCallback Interface Reference . . . 21
3.9.1 Detailed Description . . . 21
3.9.2 Member Function Documentation . . . 21
3.10 CardReaderHandler.CardReaderSelectionCallback Interface Reference . . . 21
3.10.1 Detailed Description . . . 21
3.10.2 Member Function Documentation . . . 21
3.11 CardReaderStatus Enum Reference . . . 22
3.11.1 Detailed Description . . . 22
3.11.2 Member Data Documentation . . . 22
3.12 CardReaderHandler.CardReaderTransactionInfoCallback Interface Reference . . . 24
3.12.1 Detailed Description . . . 24
3.12.2 Member Function Documentation . . . 24
3.13 CheckoutHandler Interface Reference . . . 25
3.13.1 Detailed Description . . . 25
3.13.2 Member Function Documentation . . . 25
3.14 Config Class Reference . . . 26
3.14.1 Detailed Description . . . 26
3.14.2 Constructor & Destructor Documentation . . . 26
3.14.3 Member Function Documentation . . . 27
3.14.4 Member Data Documentation . . . 30
3.15 CurrencyCode Enum Reference . . . 31
3.15.1 Detailed Description . . . 31
3.15.2 Member Data Documentation . . . 31
3.16 Error Class Reference . . . 31
3.16.1 Detailed Description . . . 32
3.16.2 Member Function Documentation . . . 32
3.16.3 Member Data Documentation . . . 33

3.17 ErrorCode Enum Reference . . . 34
3.17.1 Detailed Description . . . 34
3.17.2 Member Data Documentation . . . 34
3.18 LogLevel Enum Reference . . . 37
3.18.1 Detailed Description . . . 37
3.18.2 Member Data Documentation . . . 37
3.19 MockConfig Class Reference . . . 38
3.19.1 Detailed Description . . . 38
3.19.2 Constructor & Destructor Documentation . . . 38
3.19.3 Member Function Documentation . . . 39
3.20 PaymentInfo Class Reference . . . 44
3.20.1 Detailed Description . . . 44
3.20.2 Constructor & Destructor Documentation . . . 44
3.20.3 Member Function Documentation . . . 45
3.21 PaymentMethod Enum Reference . . . 47
3.21.1 Detailed Description . . . 47
3.21.2 Member Data Documentation . . . 47
3.22 PaymentToken Class Reference . . . 47
3.22.1 Detailed Description . . . 47
3.22.2 Constructor & Destructor Documentation . . . 47
3.22.3 Member Function Documentation . . . 48
3.23 TokenizationHandler Interface Reference . . . 48
3.23.1 Detailed Description . . . 48
3.23.2 Member Function Documentation . . . 48
3.24 WePay Class Reference . . . 49
3.24.1 Detailed Description . . . 49
3.24.2 Constructor & Destructor Documentation . . . 49
3.24.3 Member Function Documentation . . . 50

Index 53

1 Getting Started

Introduction

The WePay Android SDK enables collection of payments via various payment methods.

It is meant for consumption by WePay partners who are developing their own Android apps aimed at merchants and/or consumers.

Regardless of the payment method used, the SDK will ultimately return a Payment Token, which must be redeemed via a server-to-server API call to complete the transaction.

Payment methods

There are two types of payment methods:

- Consumer payment methods - to be used in apps where consumers directly pay and/or make donations
- Merchant payment methods - to be used in apps where merchants collect payments from their customers

The WePay Android SDK supports the following payment methods:

- Card Reader: Using an EMV Card Reader, a merchant can accept in-person payments by prosessing a consumer's EMV-enabled chip card. Traditional magnetic stripe cards can be processed as well.
- Manual Entry (Consumer/Merchant): The Manual Entry payment method lets consumer and merchant apps accept payments by allowing the user to manually enter card info.

Installation

In the following steps, [version] represent one particular sdk version identifier such as 1.0.0 Replace [version] in following steps with the sdk version you are using

- Add the following jars to the libs directory under app directory of your project source:
  1. wepay-android-[version].aar
  2. wepay-android-[version]-javadoc.jar
  3. wepay-android-[version]-sources.jar

  For example, if you are using sdk version 1.0.0, you need to include the following files
  1. wepay-android-1.0.0.aar
  2. wepay-android-1.0.0-javadoc.jar
  3. wepay-android-1.0.0-sources.jar

- Open build.gradle file for your app module (not the build.gradle file of the project) and add the following

```
repositories{
    flatDir{
        dirs 'libs'
    }
}
```

- Also add the following to the dependencies closure

```
compile(name:'wepay-android-[version]', ext:'aar')
compile 'com.google.code.gson:gson:2.2.2'
```

As an example, if you are using sdk version 1.0.0, you need to add the following in dependencies closure

```
compile(name:'wepay-android-1.0.0', ext:'aar')
compile 'com.google.code.gson:gson:2.2.2'
```

- Open your app's manifest.xml and add the following permissions under the manifest tag:

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

- Android 6 / M / API 23 and later require a more complicated mechanism of requesting audio permissions from the user. See the WePayExample app's MainActivity.java for a sample implementation.
- Android 6 also requires COARSE_LOCATION permission in order to scan for Bluetooth devices. This is most likely a bug in Android. You can view a discussion of the topic here.
- Clean and build the project using your IDE or from the command line by going to the project's base directory and running:

```
./gradlew clean build
```

- Done!

Note: Card reader functionality is not available in this SDK by default. If you are interested in using the WePay Card Reader, please contact your sales representative or account manager. If you have yet to be in direct contact with WePay, please email sales@wepay.com.

Documentation

HTML documentation is hosted on our Github Pages Site.

Pdf documentation is available on the releases page or as a direct download.

SDK Organization

com.wepay.android.WePay

The WePay class is the starting point for consuming the SDK, and is the primary class you will interact with. It exposes all the methods you can call to accept payments via the supported payment methods. Detailed reference documentation is available on the reference page for the Wepay class.

Interfaces

The SDK uses interfaces to repond to API calls. You will implement the relevant interfaces to receive responses to the API calls you make. Detailed reference documentation is available on the reference page for each interface:

- com.wepay.android.AuthorizationHandler
- com.wepay.android.BatteryLevelHandler
- com.wepay.android.CalibrationHandler
- com.wepay.android.CardReaderHandler
- com.wepay.android.CheckoutHandler
- com.wepay.android.TokenizationHandler

Data Models and Enums

All other classes in the SDK are data models and Enums that are used to exchange data between your app and the SDK. Detailed reference documentation is available on the reference page for each class.

Next Steps

Head over to the documentation to see all the API methods available. When you are ready, look at the samples below to learn how to interact with the SDK.

Error Handling

com.wepay.android.models.Error serves as documentation for all errors surfaced by the WePay Android SDK.

Samples

See the WePayExample app for a working implementation of all API methods.

Initializing the SDK

- Complete the installation steps (above).
- Include the wepay packages

```
import com.wepay.android.*;
import com.wepay.android.models.*;
import com.wepay.android.enums.*;
```

- Define a property to store the Wepay object

```
WePay wepay;
```

- Create a com.wepay.android.models.Config object

```
String clientId = "your_client_id";
Context context = getApplicationContext();
String environment = Config.ENVIRONMENT_STAGE;

Config config = new Config(context, clientId, environment);
```

- Initialize the WePay object and assign it to the property

```
this.wepay = new WePay(config);
```

(optional) Providing permission to use location services for fraud detection

- Open your app's manifest.xml and add the following permission under the manifest tag:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
```

- Set the option on the config object, before initializing the WePay object

```
config.setUseLocation(true);
```

Integrating the Card Reader payment methods (Swipe+Dip)

- Implement the CardReaderHandler, TokenizationHandler, and AuthorizationHandler interfaces

```
public class MainActivity extends ActionBarActivity implements CardReaderHandler, TokenizationHandler,
      AuthorizationHandler
```

- Implement the CardReaderHandler interface methods

```
@Override
public void onStatusChange(CardReaderStatus status) {
    if (status.equals(CardReaderStatus.NOT_CONNECTED)) {
        // show UI that prompts the user to connect the card reader
        this.setStatusText("Connect card reader and wait");
    } else if (status.equals(CardReaderStatus.WAITING_FOR_CARD)) {
        // show UI that prompts the user to swipe/dip
        this.setStatusText("Swipe/Dip card");
    } else if (status.equals(CardReaderStatus.SWIPE_DETECTED)) {
        // provide feedback to the user that a swipe was detected
        this.setStatusText("Swipe detected");
    } else if (status.equals(CardReaderStatus.CARD_DIPPED)) {
        // provide feedback to the user that a dip was detected
        // also let them know they should not remove the card
        this.setStatusText("Card dipped, do not remove card");
    } else if (status.equals(CardReaderStatus.TOKENIZING)) {
        // provide feedback to the user that the card is being tokenized
        this.setStatusText("Tokenizing card...");
    } else if (status.equals(CardReaderStatus.AUTHORIZING)) {
        // provide feedback to the user that the card is being authorized
        this.setStatusText("Authorizing card...");
    }  else if (status.equals(CardReaderStatus.STOPPED)) {
        // provide feedback to the user that the card reader was stopped
        this.setStatusText("card reader Stopped");
    } else {
        // handle all other status change notifications
        this.setStatusText(status.toString());
    }
}

@Override
public void onCardReaderSelection(final CardReaderSelectionCallback callback, ArrayList<String>
      cardReaderNames) {
    // In production apps, the merchant must choose the card reader they want to use.
    // Here, we always select the first card reader in the array
    int selectedIndex = 0;
    callback.useCardReaderAtIndex(selectedIndex);
}

@Override
public void onReaderResetRequested(CardReaderResetCallback callback) {
    // decide if you want to reset the reader,
    // then execute the callback with the appropriate response
    callback.resetCardReader(false);
}

@Override
public void onTransactionInfoRequested(CardReaderTransactionInfoCallback callback) {
    // provide the amount, currency code and WePay account ID of the merchant
    callback.useTransactionInfo(new BigDecimal("21.61"), CurrencyCode.USD, accountId);
}

@Override
```

```
public void onEMVApplicationSelectionRequested(ApplicationSelectionCallback callback, ArrayList<String>
      applications) {
    // Ask the payer to select an application from the list,
    // then execute the callback with the index of the selected application
    callback.useApplicationAtIndex(0);
}

@Override
public void onPayerEmailRequested(CardReaderEmailCallback callback) {
    // provide the email address of the payer
    callback.insertPayerEmail("android-example@wepay.com");
}

@Override
public void onSuccess(PaymentInfo paymentInfo) {
    // use the payment info (for display/recordkeeping)
    // wait for card tokenization response
}

@Override
public void onError(Error error) {
    // handle the error
}
```

- Implement the TokenizationHandler interface methods

```
@Override
public void onSuccess(PaymentInfo paymentInfo, PaymentToken token) {
    // Send the tokenId (paymentToken.getTokenId()) to your server
    // Your server would use the tokenId to make a /checkout/create call to complete the transaction
}


@Override
public void onError(PaymentInfo paymentInfo, Error error) {
    // Handle error
}
```

- Implement the AuthorizationHandler interface methods

```
@Override
public void onAuthorizationSuccess(PaymentInfo paymentInfo, AuthorizationInfo authorizationInfo) {
    // Send the tokenId (authorizationInfo.getTokenId()) and transactionToken
       (authorizationInfo.getTransactionToken()) to your server
    // Your server will use these values to make a /checkout/create call to complete the transaction
}

@Override
public void onAuthorizationError(PaymentInfo paymentInfo, Error error) {
    // handle the error
}
```

- Make the WePay API call, passing in the instance(s) of the class(es) that implemented the interface methods

```
this.wepay.startCardReaderForTokenizing(this, this, this);
// Show UI asking the user to insert the card reader and wait for it to be ready
```

- That's it! The following sequence of events will occur:

1. The user inserts the card reader (or it is already inserted), or powers on their bluetooth card reader.
2. The SDK tries to detect the card reader and initialize it.
    - The `onStatusChange` method will be called with `status = SEARCHING_FOR_READER`.
    - If any card readers are discovered, the `onCardReaderSelection` method will be called with a list of discovered devices. If anything is plugged into the headphone jack, `"AUDIOJACK"` will be one of the devices discovered.
    - If no card readers are detected, the `onStatusChange` method will be called with `status = NOT_CONNECTED`

- Once `callback.useCardReaderAtIndex()` is called, the SDK will attempt to to connect to the selected card reader.
- If the card reader is successfully connected, then the `onStatusChange` method will be called with `status = CONNECTED`.

3. Next, the SDK checks if the card reader is correctly configured (the `onStatusChange` method will be called with `status = CHECKING_READER`).
   - If the card reader is already configured, the app is given a chance to force configuration. The SDK calls the `onReaderResetRequested` method, and the app must execute the callback method, telling the SDK whether or not the reader should be reset.
   - If the reader was not already configured, or the app requested a reset, the card reader is configured (the `onStatusChange` method will be called with `status = CONFIGURING_READER`)
4. Next, if the card reader is successfully initialized, the SDK asks the app for transaction information by calling the `onTransactionInfoRequested` method. The app must execute the callback method, telling the SDK what the amount, currency code and merchant account id is.
5. Next, the `onStatusChange` method will be called with `status = WAITING_FOR_CARD`
6. If the user swipes a card successfully:
   - The `onStatusChange` method will be called with `status = SWIPE_DETECTED`.
   - The SDK asks the app for the payer's email by calling the `onPayerEmailRequested` method. The app must execute the completion method and pass in the payer's email address.
   - The `onSuccess` method is called with the obtained payment info.
   - The `onStatusChange` method will be called with `status = TOKENIZING`, and the SDK will automatically send the obtained card info to WePay's servers for tokenization.
   - If tokenization succeeds, TokenizationHandler's `onSuccess` method will be called.
   - If tokenization fails, TokenizationHandler's `onError` method will be called with the appropriate error, and processing will stop.
7. Instead, if the user dips a card successfully:
   - The `onStatusChange:` method will be called with `status = CARD_DIPPED`
   - If the card has multiple applications on it, the payer must choose one:
     - The SDK calls the `onEMVApplicationSelectionRequested` method with a list of Applications on the card.
     - The app must display these Applications to the payer and allow them to choose which application they want to use.
     - Once the payer has decided, the app must inform the SDK of the choice by executing the calback method and passing in the index of the chosen application.
   - Next, the SDK obtains card data from the chip on the card.
   - The SDK asks the app for the payer's email by calling the `onPayerEmailRequested` method. The app must execute the completion method and pass in the payer's email address.
   - The `onSuccess` method is called with the obtained payment info.
   - The `onStatusChange` method will be called with `status = AUTHORIZING`, and the SDK will automatically send the obtained EMV card info to WePay's servers for authorization.
   - If authorization succeeds, the `onAuthorizationSuccess` method will be called and processing will stop.
   - If authorization fails, the `onAuthorizationError` method will be called.

8. If a recoverable error occurs during swiping or dipping, one of the `onError` methods will be called. After a few seconds, the `onStatusChange` method will be called with `status = WAITING_FOR_CARD` and the card reader will again wait for the user to swipe/dip a card.
9. If an unrecoverable error occurs, or if the SDK is unable to obtain data from the card, one of the `onError` methods will be called with the appropriate error.
10. When processing stops, the `onStatusChange` method will be called with `status = kWPCardReader↩StatusStopped`.

1. Done!

Note: After the card is inserted into the reader, it must not be removed until a successful auth response (or an error) is returned.

Integrating the Manual payment method

- Implement the TokenizationHandler interface

```
public class MainActivity extends ActionBarActivity implements TokenizationHandler
```

- Implement the TokenizationHandler interface methods

```
@Override
public void onSuccess(PaymentInfo paymentInfo, PaymentToken token) {
    // Send the tokenId (paymentToken.getTokenId()) to your server
    // Your server would use the tokenId to make a /checkout/create call to complete the transaction
}


@Override
public void onError(PaymentInfo paymentInfo, Error error) {
    // Handle error
}
```

- Instantiate a PaymentInfo object using the user's credit card and address data

```
Address address = new Address(Locale.getDefault());
address.setAddressLine(0, "380 Portage ave");
address.setLocality("Palo Alto");
address.setPostalCode("94306");
address.setCountryCode("US");

PaymentInfo paymentInfo = new PaymentInfo("Android", "Tester", "a@b.com",
        "Visa xxxx-1234", address,
        address, PaymentMethod.MANUAL,
        "4242424242424242", "123", "01", "18", true);
```

- Make the WePay API call, passing in the instance of the class that implemented the TokenizationHandler interface methods

```
this.wepay.tokenize(paymentInfo, this);
```

- That's it! The following sequence of events will occur:

1. The SDK will send the obtained payment info to WePay's servers for tokenization
2. If the tokenization succeeds, TokenizationHandler's `onSuccess` method will be called
3. Otherwise, if the tokenization fails, TokenizationHandler's `onError` method will be called with the appropriate error

Integrating the Store Signature API

- Implement the CheckoutHandler interface

```
public class MainActivity extends ActionBarActivity implements CheckoutHandler
```

- Implement the CheckoutHandler interface methods

```
@Override
public void onSuccess(String signatureUrl, String checkoutId) {
    // success! nothing to do here
}


@Override
public void onError(Bitmap image, String checkoutId, Error error) {
    // handle the error
}
```

- Obtain the checkout_id associated with this signature from your server

```
String checkoutId = this.obtainCheckoutId();
```

- Instantiate a Bitmap object containing the user's signature

```
Bitmap signature = BitmapFactory.decodeResource(getApplicationContext().getResources(), R.drawable.
      dd_signature);
```

- Make the WePay API call, passing in the instance of the class that implemented the CheckoutHandler interface methods

```
this.wepay.storeSignatureImage(signature, checkoutId, this);
```

- That's it! The following sequence of events will occur:

1. The SDK will send the obtained signature to WePay's servers for tokenization
2. If the operation succeeds, CheckoutHandler's `onSuccess` method will be called
3. Otherwise, if the operation fails, CheckoutHandler's `onError` method will be called with the appropriate error

Integrating the Calibration API

Sometimes, the card reader will not work with Android devices that we have not seen before. It is possible to calibrate the card reader to these new devices so that it starts working. The calibration only needs to be performed once, and only if the card reader is not detected on first use. After successful calibration, the reader can be used on the user's device as usual.

- Implement the CalibrationHandler interface

```
public class MainActivity extends ActionBarActivity implements CalibrationHandler
```

- Implement the CalibrationHandler interface methods

```
@Override
public void onProgress(final double progress) {
    // show progress
}


@Override
public void onComplete(final CalibrationResult result, final CalibrationParameters params) {
    // show result to the user
    // send the calibration params to WePay
}
```

- Make the WePay API call, passing in the instance of the class that implemented the CalibrationHandler interface methods

```
this.wepay.calibrateCardReader(this);
```

- That's it! The following sequence of events will occur:

  1. The SDK will attempt to calibrate the reader
  2. CalibrationHandler's `onProgress` method will be called periodically to indicate the current progress
  3. When the process is completed, CalibrationHandler's `onComplete` method will be called with the result
  4. The card reader must be plugged in before attempting calibration, otherwise the process will fail

Note: If calibration succeeds, you must obtain the calibration parameters and email them to `mobile@wepay.com`. We will bake these parameters into the SDK, so that future users with the same devices will not have to run the calibration process.

Integrating the Battery Level API

- Implement the BatteryLevelHandler interface

```
public class MainActivity extends ActionBarActivity implements BatteryLevelHandler
```

- Implement the BatteryLevelHandler interface methods

```
@Override
public void onBatteryLevel(int batteryLevel) {
    // show result to the user
}

@Override
public void onBatteryLevelError(Error error) {
    // handle the error
}
```

- Make the WePay API call, passing in the instance of the class that implemented the BatteryLevelHandler interface methods

```
this.wepay.getCardReaderBatteryLevel(this);
```

- That's it! The following sequence of events will occur:

1. The SDK will attempt to read the battery level of the card reader
2. If the operation succeeds, BatteryLevelHandler's `onBatteryLevel` method will be called with the result
3. Otherwise, if the operation fails, BatteryLevelHandler's `onBatteryLevelError` method will be called with the appropriate error
4. The card reader must be plugged in before attempting to get battery level, otherwise the process will fail

Configuring the SDK

The experiences described above can be modified by utilizing the configuration options available on the Config object. Detailed descriptions for each configurable property is available in the documentation for Config.

Test/develop using mock card reader and mock WepayClient

- To use mock card reader implementation instead of using the real reader, instantiate a MockConfig object and pass it to Config:

```
MockConfig mockConfig = new MockConfig().setUseMockCardReader(true);
config.setMockConfig(mockConfig);
```

- To use mock WepayClient implementation instead of interacting with the real WePay server, set the corresponding option on the mockConfig object:

```
mockConfig.setUseMockWepayClient(true);
```

- Other options are also available:

```
mockConfig.setMockPaymentMethod(PaymentMethod.SWIPE) // Payment method to mock; Defaults to SWIPE.
.setCardReadTimeout(true) // To mock a card reader timeout; Defaults to false.
.setCardReadFailure(true) // To mock a failure for card reading; Defaults to false.
.setCardTokenizationFailure(true) // To mock a failure for card tokenization; Defaults to false.
.setEMVAuthFailure(true) // To mock a failure for EMV authorization; Defaults to false.
.setMultipleEMVApplication(true) // To mock multiple EMV applications on card to choose from; Defaults to
       false.
.setBatteryLevelError(true); // To mock an error while fetching battery level; Defaults to false.
.setMockCardReaderDetected(false); // To mock a card reader being available for connection; Defaults to
       true.
```

Integration tests and unit tests

All the integration tests and unit tests are located in the `src/androidTest/java/` directory. The tests are instrumented tests so be sure to have a connected running physical device or emulator before running the tests.

From Android Studio

- To run a single test, right-click the test method and select "Run".
- To run all test methods in a class, right-click the class and select "Run".
- To run all tests in a directory, right-click the directory and select "Run tests".

From the command line

Change to this project's directory and call the connectedAndroidTest (or cAT) task:

```
1 ./gradlew cAT
```

- HTML test result files can be found at: `<path_to_your_project>/app/build/reports/androidTests/connected/` directory.
- XML test result files: `<path_to_your_project>/app/build/outputs/androidTest-results/connected/` directory.

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

CardReaderHandler.ApplicationSelectionCallback 13

AuthorizationHandler 13

BatteryLevelHandler 15

CalibrationHandler 16

CalibrationParameters 17

CalibrationResult 17

CardReaderHandler.CardReaderEmailCallback 18

CardReaderHandler 18

CardReaderHandler.CardReaderResetCallback 21

CardReaderHandler.CardReaderSelectionCallback 21

CardReaderStatus 22

CardReaderHandler.CardReaderTransactionInfoCallback 24

CheckoutHandler 25

Config 26

CurrencyCode 31

Error 31

ErrorCode 34

LogLevel 37

MockConfig 38

PaymentInfo 44

PaymentMethod 47

PaymentToken 47

TokenizationHandler 48

WePay 49

3 Class Documentation

3.1 CardReaderHandler.ApplicationSelectionCallback Interface Reference

Public Member Functions

- void useApplicationAtIndex (int selectedIndex)

3.1.1 Detailed Description

The Interface ApplicationSelectionCallback defines the callback method used to provide information to the card reader during a Dip transaction.

3.1.2 Member Function Documentation

3.1.2.1 void useApplicationAtIndex (int *selectedIndex*)

The callback function that must be executed by the app when onEMVApplicationSelectionRequested() is called by the SDK.

Examples: callback.useApplicationAtIndex(0);

Parameters

selectedIndex	the index of the selected application in the array of applications from the card.

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/Card↩ ReaderHandler.java

3.2 AuthorizationHandler Interface Reference

Public Member Functions

- void onAuthorizationSuccess (PaymentInfo paymentInfo, AuthorizationInfo authorizationInfo)
- void onAuthorizationError (PaymentInfo paymentInfo, Error error)

3.2.1 Detailed Description

The Interface AuthorizationHandler defines the method used to return data in response to an authorization call.

3.2.2 Member Function Documentation

3.2.2.1 void onAuthorizationError (PaymentInfo *paymentInfo,* Error *error*)

Called when an authorization call fails.

Parameters

paymentInfo	the payment info for the card that failed authorization.
error	the error which caused the failure.

3.2.2.2 void onAuthorizationSuccess (PaymentInfo *paymentInfo,* AuthorizationInfo *authorizationInfo*)

Called when an authorization call succeeds.

Parameters

paymentInfo	the payment info for the card that was authorized.
authorizationInfo	the authorization info for the transaction that was authorized.

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/Authorization↩ Handler.java

3.3 BatteryLevelHandler Interface Reference

Public Member Functions

- void onBatteryLevel (int batteryLevel)
- void onBatteryLevelError (com.wepay.android.models.Error error)

3.3.1 Detailed Description

The Interface BatteryLevelHandler defines the methods used to communicate information regarding the card reader's battery level.

3.3.2 Member Function Documentation

3.3.2.1 void onBatteryLevel (int *batteryLevel*)

Gets called when the card reader's battery level is determined.

Parameters

batteryLevel	the card reader's battery charge level (0-100%).

3.3.2.2 void onBatteryLevelError (com.wepay.android.models.Error *error*)

Gets called when we fail to determine the card reader's battery level.

Parameters

error	the error due to which battery level reading failed.

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/Battery↩ LevelHandler.java

3.4 CalibrationHandler Interface Reference

Public Member Functions

- void onProgress (double progress)
- void onComplete (CalibrationResult result, CalibrationParameters params)

3.4.1 Detailed Description

The Interface CalibrationHandler defines the methods used to communicate information regarding the card reader calibration process.

3.4.2 Member Function Documentation

3.4.2.1 void onComplete (CalibrationResult *result,* CalibrationParameters *params*)

Gets called when the calibration process is completed.

Parameters

result	the result of calibration.
params	the calibration parameters that were detected. Will be null if the result is not CalibrationResult.SUCCESS.

3.4.2.2 void onProgress (double *progress*)

Gets called when the card reader calibration makes progress.

Parameters

progress	the completion percentage [0.0 - 1.0].

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/Calibration↩ Handler.java

3.5 CalibrationParameters Class Reference

3.5.1 Detailed Description

The Class CalibrationParameters contains the parameters used to calibrate the card reader.

The documentation for this class was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/models/Calibration↩ Parameters.java

3.6 CalibrationResult Enum Reference

Public Attributes

- SUCCEEDED =(0)
- FAILED =(1)
- INTERRUPTED =(2)

3.6.1 Detailed Description

The Enum CalibrationResult defines all the results that can be returned by the card reader calibration process.

3.6.2 Member Data Documentation

3.6.2.1 FAILED =(1)

Failed.

3.6.2.2 INTERRUPTED =(2)

Interrupted

3.6.2.3 SUCCEEDED =(0)

Succeeded.

The documentation for this enum was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/enums/Calibration↩ Result.java

3.7 CardReaderHandler.CardReaderEmailCallback Interface Reference

Public Member Functions

- void insertPayerEmail (String email)

3.7.1 Detailed Description

The Interface CardReaderEmailCallback defines the method used to provide email information to the card reader after a transaction.

3.7.2 Member Function Documentation

3.7.2.1 void insertPayerEmail (String *email*)

The callback function that must be executed by the app when onPayerEmailRequested() is called by the SDK.

Examples: callback.insertPayerEmail("android-example@wepay.com"); callback.insertPayerEmail(null);

Parameters

email	the payer's email address.

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/CardReaderHandler.java

3.8 CardReaderHandler Interface Reference

Classes

- interface ApplicationSelectionCallback
- interface CardReaderEmailCallback
- interface CardReaderResetCallback
- interface CardReaderSelectionCallback
- interface CardReaderTransactionInfoCallback

Public Member Functions

- void onEMVApplicationSelectionRequested (ApplicationSelectionCallback callback, ArrayList< String > applications)
- void onSuccess (PaymentInfo paymentInfo)
- void onError (Error error)
- void onStatusChange (CardReaderStatus status)
- void onReaderResetRequested (CardReaderResetCallback callback)
- void onTransactionInfoRequested (CardReaderTransactionInfoCallback callback)
- void onPayerEmailRequested (CardReaderEmailCallback callback)
- void onCardReaderSelection (CardReaderSelectionCallback callback, ArrayList< String > cardReaderNames)

3.8.1 Detailed Description

The Interface CardReaderHandler defines the methods used to communicate information regarding the card reader.

3.8.2 Member Function Documentation

3.8.2.1 void onCardReaderSelection (CardReaderSelectionCallback *callback,* ArrayList< String > *cardReaderNames*)

Gets called when card reader devices have been discovered, to give the app an opportunity to select which card reader to initialize. The app must respond by executing callback. The card reader will not be initialized until the callback is executed.

Parameters

callback	the callback object.
cardReaderNames	the list of device names.

3.8.2.2 void onEMVApplicationSelectionRequested (ApplicationSelectionCallback *callback,* ArrayList< String > *applications*)

Called when the EMV card contains more than one application. The applications should be presented to the payer for selection. Once the payer makes a choice, the app must execute callback.useApplicationAtIndex() with the index of the selected application. The transaction cannot proceed until the callback is executed.

Example: callback.useApplicationAtIndex(0);

Parameters

callback	the callback object.
applications	the array of String containing application names from the card.

3.8.2.3 void onError (Error *error*)

Gets called when the card reader fails to read a card's information.

Parameters

error	the error due to which card reading failed.

3.8.2.4 void onPayerEmailRequested (CardReaderEmailCallback *callback*)

Gets called so that an email address can be provided before a transaction is authorized. The app must respond by executing callback.insertPayerEmail(). The transaction cannot proceed until the callback is executed.

Parameters

callback	the callback object.

3.8.2.5 void onReaderResetRequested (CardReaderResetCallback *callback*)

Gets called when the connected card reader is previously configured, to give the app an opportunity to reset the device. The app must respond by executing callback.resetCardReader(). The transaction cannot proceed until this callback is executed. The card reader must be reset here if the merchant manually resets the reader via the hardware reset button on the reader.

Parameters

callback	the callback object.

3.8.2.6 void onStatusChange (CardReaderStatus *status*)

Gets called whenever the card reader changes status.

Parameters

status	the status.

3.8.2.7 void onSuccess (PaymentInfo *paymentInfo*)

Gets called when the card reader reads a card's information successfully.

Parameters

paymentInfo	the payment info read from a card.

3.8.2.8 void onTransactionInfoRequested (CardReaderTransactionInfoCallback *callback*)

Gets called so that the app can provide the amount, currency code and the WePay account Id of the merchant. The app must respond by executing callback.useTransactionInfo(). The transaction cannot proceed until this callback is executed.

Parameters

callback	the callback object.

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/CardReaderHandler.java

3.9 CardReaderHandler.CardReaderResetCallback Interface Reference

Public Member Functions

- void resetCardReader (boolean shouldReset)

3.9.1 Detailed Description

The Interface CardReaderResetCallback defines the method used to provide information to the card reader before a transaction.

3.9.2 Member Function Documentation

3.9.2.1 void resetCardReader (boolean *shouldReset*)

The callback function that must be executed by the app when onReaderResetRequested() is called by the SDK.

Examples: callback.resetCardReader(true); callback.resetCardReader(false);

Parameters

shouldReset	The answer to the question: "Should the card reader be reset?".

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/CardReaderHandler.java

3.10 CardReaderHandler.CardReaderSelectionCallback Interface Reference

Public Member Functions

- void useCardReaderAtIndex (int selectedIndex)

3.10.1 Detailed Description

The Interface CardReaderSelectionCallback defines the callback method used to select which card reader to initialize.

3.10.2 Member Function Documentation

3.10.2.1 void useCardReaderAtIndex (int *selectedIndex*)

The callback function that must be executed by the app when onCardReaderSelection() is called by the SDK.

Examples: callback.useCardReaderAtIndex(0);

Parameters

selectedIndex	the index of the selected card reader in the array of detected card readers.

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/CardReaderHandler.java

3.11 CardReaderStatus Enum Reference

Public Attributes

- NOT_CONNECTED =(0)
- WAITING_FOR_CARD =(1)
- TOKENIZING =(2)
- STOPPED =(3)
- CONNECTED =(4)
- SWIPE_DETECTED =(5)
- CHECK_CARD_ORIENTATION =(6)
- CHECKING_READER =(7)
- CONFIGURING_READER =(8)
- SHOULD_NOT_SWIPE_EMV_CARD =(9)
- CHIP_ERROR_SWIPE_CARD =(10)
- CARD_DIPPED =(11)
- AUTHORIZING =(12)
- SWIPE_ERROR_SWIPE_AGAIN =(13)
- SEARCHING_FOR_READER =(14)

3.11.1 Detailed Description

The Enum CardReaderStatus defines all the statuses that can be returned by the card reader.

3.11.2 Member Data Documentation

3.11.2.1 AUTHORIZING =(12)

Authorizing.

3.11.2.2 CARD_DIPPED =(11)

Card dipped.

3.11.2.3 CHECK_CARD_ORIENTATION =(6)

Check card orientation.

3.11.2.4 CHECKING_READER =(7)

Checking reader.

3.11.2.5 CHIP_ERROR_SWIPE_CARD =(10)

Chip error, swipe card.

3.11.2.6 CONFIGURING_READER =(8)

Configuring reader.

3.11.2.7 CONNECTED =(4)

Connected.

3.11.2.8 NOT_CONNECTED =(0)

Not connected.

3.11.2.9 SEARCHING_FOR_READER =(14)

Searching for a card reader.

3.11.2.10 SHOULD_NOT_SWIPE_EMV_CARD =(9)

Should not swipe EMV card.

3.11.2.11 STOPPED =(3)

Stopped.

3.11.2.12 SWIPE_DETECTED =(5)

Swipe detected.

3.11.2.13 SWIPE_ERROR_SWIPE_AGAIN =(13)

Swipe error, swipe again.

3.11.2.14 TOKENIZING =(2)

Tokenizing.

3.11.2.15 WAITING_FOR_CARD =(1)

Waiting for card.

The documentation for this enum was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/enums/CardReaderStatus.java

3.12 CardReaderHandler.CardReaderTransactionInfoCallback Interface Reference

Public Member Functions

- void useTransactionInfo (BigDecimal amount, CurrencyCode currencyCode, long accountId)

3.12.1 Detailed Description

The Interface CardReaderTransactionInfoCallback defines the method used to provide transaction information to the card reader before a transaction.

3.12.2 Member Function Documentation

3.12.2.1 void useTransactionInfo (BigDecimal *amount,* CurrencyCode *currencyCode,* long *accountId*)

The callback function that must be executed when onTransactionInfoRequested() is called by the SDK. Note: In the staging environment, use amounts of 20.61, 120.61, 23.61 and 123.61 to simulate authorization errors. Amounts of 21.61, 121.61, 22.61, 122.61, 24.61, 124.61, 25.61 and 125.61 will simulate successful auth.

Example: callback.useTransactionInfo(new BigDecimal("21.61"), CurrencyCode.USD, 1234567);

Parameters

amount	the amount for the transaction. For USD amounts, there can be a maximum of two places after the decimal point.
currencyCode	the currency code for the transaction. e.g. CurrencyCode.USD.
accountId	the WePay account id of the merchant.

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/CardReaderHandler.java

3.13 CheckoutHandler Interface Reference

Public Member Functions

- void onSuccess (String signatureUrl, String checkoutId)
- void onError (Bitmap image, String checkoutId, Error error)

3.13.1 Detailed Description

The Interface CheckoutHandler defines the methods used to return results of a storeSignature operation.

3.13.2 Member Function Documentation

3.13.2.1 void onError (Bitmap *image,* String *checkoutId,* Error *error*)

Gets called when an error occurs while storing a signature.

Parameters

image	the signature image to be stored.
checkoutId	the checkout id associated with the signature.
error	the error which caused the failure.

3.13.2.2 void onSuccess (String *signatureUrl,* String *checkoutId*)

Gets called when a signature is successfully stored for the given checkout id.

Parameters

signatureUrl	the url for the signature image.
checkoutId	the checkout id associated with the signature.

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/CheckoutHandler.java

3.14 Config Class Reference

Public Member Functions

- Config (Context context, String clientId, String environment)
- Context getContext ()
- String getClientId ()
- String getEnvironment ()
- boolean isUseLocation ()
- Config setUseLocation (boolean useLocation)
- boolean isUseTestEMVCards ()
- Config setUseTestEMVCards (boolean useTestEMVCards)
- boolean shouldStopCardReaderAfterOperation ()
- Config setStopCardReaderAfterOperation (boolean stopCardReaderAfterOperation)
- boolean shouldRestartTransactionAfterSuccess ()
- Config setRestartTransactionAfterSuccess (boolean restartTransactionAfterSuccess)
- boolean shouldRestartTransactionAfterGeneralError ()
- Config setRestartTransactionAfterGeneralError (boolean restartTransactionAfterGeneralError)
- boolean shouldRestartTransactionAfterOtherErrors ()
- Config setRestartTransactionAfterOtherErrors (boolean restartTransactionAfterOtherErrors)
- LogLevel getLogLevel ()
- Config setLogLevel (LogLevel logLevel)
- MockConfig getMockConfig ()
- Config setMockConfig (MockConfig mockConfig)

Static Public Attributes

- final static String ENVIRONMENT_STAGE = "stage"
- final static String ENVIRONMENT_PRODUCTION = "production"

3.14.1 Detailed Description

The Class Config contains the configuration required to initialize the sdk.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 Config (Context *context*, String *clientId*, String *environment*)

Instantiates a new config.

Parameters

context	the application context
clientId	the client id for your WePay app
environment	the environment (use one of the provided constants - ENVIRONMENT_STAGING or ENVIRONMENT_PRODUCTION)

3.14.3 Member Function Documentation

3.14.3.1 String getClientId ( )

Gets the client id.

Returns

the client id

3.14.3.2 Context getContext ( )

Gets the context.

Returns

the context

3.14.3.3 String getEnvironment ( )

Gets the environment.

Returns

the environment

3.14.3.4 LogLevel getLogLevel ( )

Gets the log level.

Returns

the logLevel

3.14.3.5 MockConfig getMockConfig ( )

Gets the MockConfig instance.

Returns

the MockConfig instance

3.14.3.6 boolean isUseLocation ( )

Determines if we should use location services.

Returns

the use location config

3.14.3.7 boolean isUseTestEMVCards ( )

Determines if we should use test EMV cards.

Returns

the use test EMV cards config

3.14.3.8 Config setLogLevel (LogLevel *logLevel*)

Sets the log level.

Parameters

logLevel	determines which SDK log messages should show

Returns

the config

3.14.3.9 Config setMockConfig (MockConfig *mockConfig*)

Sets the MockConfig instance to be used.

Parameters

mockConfig	the MockConfig instance

Returns

the config

3.14.3.10 Config setRestartTransactionAfterGeneralError (boolean *restartTransactionAfterGeneralError*)

Sets the option for the transaction to automatically restart after a general error (errorCategory:ERROR_CATEGORY↩
_CARD_READER, errorCode:CARD_READER_GENERAL_ERROR). If not explicitly set to false, defaults to true.

Parameters

restartTransactionAfterGeneralError	the flag to determine if the transaction should automatically restart after a general error.

Returns

the config

3.14.3.11 Config setRestartTransactionAfterOtherErrors (boolean *restartTransactionAfterOtherErrors*)

Sets the option for the transaction to automatically restart after an error other than general error. If not explicitly set to true, defaults to false.

Parameters

restartTransactionAfterOtherErrors	the flag to determine if the transaction should automatically restart after an error other than general error.

Returns

the config

3.14.3.12 Config setRestartTransactionAfterSuccess (boolean *restartTransactionAfterSuccess*)

Sets the option for the transaction to automatically restart after a successful swipe. If not explicitly set to true, defaults to false.

Parameters

restartTransactionAfterSuccess	the flag to determine if the transaction should automatically restart after a successful swipe.

Returns

the config

3.14.3.13 Config setStopCardReaderAfterOperation (boolean *stopCardReaderAfterOperation*)

Sets the option for the card reader to automatically stop after an operation. If not explicitly set to false, defaults to true.

Parameters

stopCardReaderAfterOperation	the flag to determine if the card reader should automatically stop after an operation.

Returns

the config

3.14.3.14 Config setUseLocation (boolean *useLocation*)

Sets the option for using location services for fraud detection purposes. If not explicitly set to true, defaults to false.

Parameters

useLocation	the permission to use location

Returns

the config

3.14.3.15 Config setUseTestEMVCards (boolean *useTestEMVCards*)

Sets the option for using test EMV cards. If not explicitly set to true, defaults to false.

Parameters

useTestEMVCards	the permission to use location

Returns

the config

3.14.3.16 boolean shouldRestartTransactionAfterGeneralError ( )

Determines if the transaction should automatically restart after a general error (errorCategory:ERROR_CATEGORY↩
_CARD_READER, errorCode:CARD_READER_GENERAL_ERROR).

Returns

true, if the transaction restarts after a general error.

3.14.3.17 boolean shouldRestartTransactionAfterOtherErrors ( )

Determines if the transaction should automatically restart after an error other than general error.

Returns

true, if the transaction restarts after an error other than general error.

3.14.3.18 boolean shouldRestartTransactionAfterSuccess ( )

Determines if the transaction should automatically restart after a successful swipe.

Returns

true, if the transaction restarts after success.

3.14.3.19 boolean shouldStopCardReaderAfterOperation ( )

Determines if the card reader should automatically stop after a transaction is completed.

Returns

true, if the card reader restarts after success

3.14.4 Member Data Documentation

3.14.4.1 final static String ENVIRONMENT_PRODUCTION = "production" [static]

The constant string representing the production environment.

3.14.4.2 final static String ENVIRONMENT_STAGE = "stage" [static]

The constant string representing the staging environment.

The documentation for this class was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/models/Config.↩ java

3.15 CurrencyCode Enum Reference

Public Attributes

- USD =(0)

3.15.1 Detailed Description

The Enum CurrencyCode defines all currency codes supported by the sdk.

3.15.2 Member Data Documentation

3.15.2.1 USD =(0)

USD

The documentation for this enum was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/enums/Currency↩ Code.java

3.16 Error Class Reference

Inherits Exception.

Public Member Functions

- String getErrorCategory ()
- String getErrorDomain ()
- String getErrorDescription ()
- Integer getErrorCode ()
- Exception getInnerException ()

Static Public Attributes

- final static String ERROR_DOMAIN_API = "com.wepay.api"
- final static String ERROR_DOMAIN_SDK = "com.wepay.sdk"
- final static String ERROR_CATEGORY_CARD_READER = "card_reader_error"
- final static String ERROR_CATEGORY_API = "api_error"
- final static String ERROR_CATEGORY_SDK = "sdk_error"

3.16.1 Detailed Description

The Class Error contains information about an error that occurs in the sdk.

3.16.2 Member Function Documentation

3.16.2.1 String getErrorCategory ( )

Gets the error category.

Returns

the error category

3.16.2.2 Integer getErrorCode ( )

Gets the error code.

Returns

the error code

3.16.2.3 String getErrorDescription ( )

Gets the error description.

Returns

the error description

3.16.2.4 String getErrorDomain ( )

Gets the error domain.

Returns

the error domain

3.16.2.5 Exception getInnerException ( )

Gets the inner exception.

Returns

the inner exception

3.16.3 Member Data Documentation

3.16.3.1 final static String ERROR_CATEGORY_API = "api_error" [static]

The constant string representing the error category API Error.

3.16.3.2 final static String ERROR_CATEGORY_CARD_READER = "card_reader_error" [static]

The constant string representing the error category Card reader Error.

3.16.3.3 final static String ERROR_CATEGORY_SDK = "sdk_error" [static]

The constant string representing the error category SDK Error.

3.16.3.4 final static String ERROR_DOMAIN_API = "com.wepay.api" [static]

The constant ERROR_DOMAIN_API

3.16.3.5 final static String ERROR_DOMAIN_SDK = "com.wepay.sdk" [static]

The constant ERROR_DOMAIN_SDK

The documentation for this class was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/models/Error.↩ java

3.17 ErrorCode Enum Reference

Public Attributes

- UNKNOWN_ERROR =(10000)
- NO_DATA_RETURNED_ERROR =(10015)
- CARD_READER_GENERAL_ERROR =(10016)
- CARD_READER_INITIALIZATION_ERROR =(10017)
- CARD_READER_TIME_OUT_ERROR =(10018)
- CARD_READER_STATUS_ERROR =(10019)
- INVALID_SIGNATURE_IMAGE_ERROR =(10020)
- NAME_NOT_FOUND_ERROR =(10021)
- INVALID_CARD_DATA =(10022)
- CARD_NOT_SUPPORTED =(10023)
- EMV_TRANSACTION_ERROR =(10024)
- INVALID_APPLICATION_ID =(10025)
- DECLINED_BY_CARD =(10026)
- CARD_BLOCKED =(10027)
- CARD_DECLINED_BY_ISSUER =(10028)
- ISSUER_UNREACHABLE =(10029)
- INVALID_TRANSACTION_INFO =(10030)
- TRANSACTION_INFO_NOT_PROVIDED =(10031)
- PAYMENT_METHOD_CANNOT_BE_TOKENIZED =(10032)
- FAILED_TO_GET_BATTERY_LEVEL =(10033)
- CARD_READER_NOT_CONNECTED_ERROR =(10034)
- CARD_READER_MODEL_NOT_SUPPORTED =(10035)
- INVALID_TRANSACTION_AMOUNT =(10036)
- INVALID_TRANSACTION_CURRENCY_CODE =(10037)
- INVALID_TRANSACTION_ACCOUNT_ID =(10038)
- INVALID_CARD_READER_SELECTION =(10039)
- CARD_READER_BATTERY_TOO_LOW =(10040)
- CARD_READER_UNABLE_TO_CONNECT =(10041)

3.17.1 Detailed Description

The Enum ErrorCode defines all error codes returned by the sdk itself. For error codes returned by the server api, visit https://www.wepay.com/developer/reference/errors

3.17.2 Member Data Documentation

3.17.2.1 CARD_BLOCKED =(10027)

The card blocked error.

3.17.2.2 CARD_DECLINED_BY_ISSUER =(10028)

The declined by issuer error.

3.17.2.3 CARD_NOT_SUPPORTED =(10023)

The card not supported error.

3.17.2.4 CARD_READER_BATTERY_TOO_LOW =(10040)

The card reader battery too low error.

3.17.2.5 CARD_READER_GENERAL_ERROR =(10016)

The card reader general error.

3.17.2.6 CARD_READER_INITIALIZATION_ERROR =(10017)

The card reader initialization error.

3.17.2.7 CARD_READER_MODEL_NOT_SUPPORTED =(10035)

The card reader model not supported error.

3.17.2.8 CARD_READER_NOT_CONNECTED_ERROR =(10034)

The card reader not connected error.

3.17.2.9 CARD_READER_STATUS_ERROR =(10019)

The card reader status error.

3.17.2.10 CARD_READER_TIME_OUT_ERROR =(10018)

The card reader time out error.

3.17.2.11 CARD_READER_UNABLE_TO_CONNECT =(10041)

Unable to connect to card reader. It may not be a supported model or it could be that the plugged in headphone jack card reader has insufficient battery.

3.17.2.12 DECLINED_BY_CARD =(10026)

The declined by card error.

3.17.2.13 EMV_TRANSACTION_ERROR =(10024)

The EMV transaction error.

3.17.2.14 FAILED_TO_GET_BATTERY_LEVEL =(10033)

The failed to get battery info error.

3.17.2.15 INVALID_APPLICATION_ID =(10025)

The invalid application error.

3.17.2.16 INVALID_CARD_DATA =(10022)

The invalid card data error.

3.17.2.17 INVALID_CARD_READER_SELECTION =(10039)

The invalid card reader selection error.

3.17.2.18 INVALID_SIGNATURE_IMAGE_ERROR =(10020)

The invalid signature image error.

3.17.2.19 INVALID_TRANSACTION_ACCOUNT_ID =(10038)

The invalid transaction account id error.

3.17.2.20 INVALID_TRANSACTION_AMOUNT =(10036)

The invalid transaction amount error.

3.17.2.21 INVALID_TRANSACTION_CURRENCY_CODE =(10037)

The invalid transaction currency code error.

3.17.2.22 INVALID_TRANSACTION_INFO =(10030)

The invalid transaction info.

3.17.2.23 ISSUER_UNREACHABLE =(10029)

The issuer unreachable error.

3.17.2.24 NAME_NOT_FOUND_ERROR =(10021)

The name not found error.

3.17.2.25 NO_DATA_RETURNED_ERROR =(10015)

The no data returned error.

3.17.2.26 PAYMENT_METHOD_CANNOT_BE_TOKENIZED =(10032)

The payment method cannot be tokenized error.

3.17.2.27 TRANSACTION_INFO_NOT_PROVIDED =(10031)

The transaction info not provided error.

3.17.2.28 UNKNOWN_ERROR =(10000)

The unknown error.

The documentation for this enum was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/enums/Error↩ Code.java

3.18 LogLevel Enum Reference

Public Attributes

- NONE =(0)
- ALL =(1)

3.18.1 Detailed Description

The levels of messages the SDK will log. We currently support showing either no log messages or all of them. The SDK log level can be set in a Config object and passed into the WePay constructor.

See also

com.wepay.android.models.Config
com.wepay.android.WePay

3.18.2 Member Data Documentation

3.18.2.1 ALL =(1)

All log messages.

3.18.2.2 NONE =(0)

No log messages.

The documentation for this enum was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/enums/Log↩ Level.java

3.19 MockConfig Class Reference

Public Member Functions

- MockConfig ()
- MockConfig (boolean useMockCardReader, boolean useMockWepayClient)
- boolean isUseMockCardReader ()
- MockConfig setUseMockCardReader (boolean useMockCardReader)
- boolean isUseMockWepayClient ()
- MockConfig setUseMockWepayClient (boolean useMockWepayClient)
- PaymentMethod getMockPaymentMethod ()
- MockConfig setMockPaymentMethod (PaymentMethod paymentMethod)
- boolean isCardReadTimeout ()
- MockConfig setCardReadTimeout (boolean cardReadTimeout)
- boolean isCardReadFailure ()
- MockConfig setCardReadFailure (boolean cardReadFailure)
- boolean isCardTokenizationFailure ()
- MockConfig setCardTokenizationFailure (boolean cardTokenizationFailure)
- boolean isEMVAuthFailure ()
- MockConfig setEMVAuthFailure (boolean EMVAuthFailure)
- boolean isBatteryLevelError ()
- MockConfig setBatteryLevelError (boolean batteryLevelError)
- boolean isMultipleEMVApplication ()
- MockConfig setMultipleEMVApplication (boolean multipleEMVApplication)
- boolean isMockCardReaderDetected ()
- MockConfig setMockCardReaderDetected (boolean isDetected)
- String getMockedDeviceName ()
- MockConfig setMockedDeviceName (String mockedDeviceName)

3.19.1 Detailed Description

The Class MockConfig contains the configuration required when using mock card reader and/or WepayClient implementation.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 MockConfig ( )

Default constructor.

3.19.2.2 MockConfig (boolean *useMockCardReader,* boolean *useMockWepayClient*)

Constructor with parameters to indicate whether mock card reader/WepayClient implementations will be used.

Parameters

useMockCardReader	if the mock card reader implementation will be used.
useMockWepayClient	If the mock WepayClient implementation will be used.

3.19.3 Member Function Documentation

3.19.3.1 String getMockedDeviceName ( )

Determines name of the device being mocked.

Returns

name of mocked device

3.19.3.2 PaymentMethod getMockPaymentMethod ( )

Determines the mocked payment method used.

Returns

the mocked payment method

3.19.3.3 boolean isBatteryLevelError ( )

Determines if a battery level error is mocked.

Returns

the battery level error config

3.19.3.4 boolean isCardReadFailure ( )

Determines if a card reading failure is mocked.

Returns

the card reading failure config

3.19.3.5 boolean isCardReadTimeout ( )

Determines if card reader timeout is mocked.

Returns

the card reader timeout config

3.19.3.6 boolean isCardTokenizationFailure ( )

Determines if a card tokenization failure is mocked.

Returns

the card tokenization failure config

3.19.3.7 boolean isEMVAuthFailure ( )

Determines if an EMV authorization failure is mocked.

Returns

the EMV authorization failure config

3.19.3.8 boolean isMockCardReaderDetected ( )

Determines if the mock card reader is available for the purpose of establishing a connection.

Returns

the card reader isDetected config

3.19.3.9 boolean isMultipleEMVApplication ( )

Determines if card having multiple EMV applications is mocked.

Returns

the multiple EMV applications config

3.19.3.10 boolean isUseMockCardReader ( )

Determines whether mocked card reader is used.

Returns

if using mocked card reader

3.19.3.11 boolean isUseMockWepayClient ( )

Determines if mocked WepayClient is used.

Returns

if using mocked WepayClient

3.19.3.12 MockConfig setBatteryLevelError (boolean *batteryLevelError*)

Sets the option for whether to mock a battery level error. If not explicitly set to true, defaults to false.

Parameters

batteryLevelError	the battery level error config

Returns

the MockConfig instance

3.19.3.13 MockConfig setCardReadFailure (boolean *cardReadFailure*)

Sets the option for whether to mock a card reading failure. If not explicitly set to true, defaults to false.

Parameters

cardReadFailure	whether to mock a card reading failure

Returns

the MockConfig instance

3.19.3.14 MockConfig setCardReadTimeout (boolean *cardReadTimeout*)

Sets the option for whether to mock card reader timeout. If not explicitly set to true, defaults to false.

Parameters

cardReadTimeout	whether to mock card reader timeout

Returns

the MockConfig instance

3.19.3.15 MockConfig setCardTokenizationFailure (boolean *cardTokenizationFailure*)

Sets the option for whether to mock card tokenization failure. If not explicitly set to true, defaults to false.

Parameters

cardTokenizationFailure	the card tokenization failure config

Returns

the MockConfig instance

3.19.3.16 MockConfig setEMVAuthFailure (boolean *EMVAuthFailure*)

Sets the option for whether to mock an EMV authorization failure. If not explicitly set to true, defaults to false.

Parameters

EMVAuthFailure	the EMV authorization failure config

Returns

the MockConfig instance

3.19.3.17 MockConfig setMockCardReaderDetected (boolean *isDetected*)

Sets the option for whether to mock a card reader that is available for the purpose of establishing a connection. If not explicitly set to false, defaults to true.

Parameters

isDetected	the card reader isDetected config

Returns

the MockConfig instance

3.19.3.18 MockConfig setMockedDeviceName (String *mockedDeviceName*)

Sets name for the mocked device.

Parameters

mockedDeviceName	name of mocked device

Returns

the MockConfig instance

3.19.3.19 MockConfig setMockPaymentMethod (PaymentMethod *paymentMethod*)

Sets the option for mocked payment method to use. If not explicitly set to DIP, defaults to SWIPE.

Parameters

paymentMethod	payment method to use

Returns

the MockConfig instance

3.19.3.20 MockConfig setMultipleEMVApplication (boolean *multipleEMVApplication*)

Sets the option for whether to mock a card with multiple EMV applications. If not explicitly set to true, defaults to false.

Parameters

multipleEMVApplication	whether to mock card with multiple EMV applications

Returns

the MockConfig instance

3.19.3.21 MockConfig setUseMockCardReader (boolean *useMockCardReader*)

Sets the option for whether to use mocked card reader. If not explicitly set to false, defaults to true.

Parameters

useMockCardReader	whether to use mocked card reader

Returns

the MockConfig instance

3.19.3.22 MockConfig setUseMockWepayClient (boolean *useMockWepayClient*)

Sets the option for whether to use mocked WepayClient. If not explicitly set to false, defaults to true.

Parameters

useMockWepayClient	whether to used mocked WepayClient

Returns

the MockConfig instance

The documentation for this class was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/models/MockConfig.java

3.20 PaymentInfo Class Reference

Public Member Functions

- PaymentInfo (String firstName, String lastName, String email, String paymentDescription, Address billingAddress, Address shippingAddress, PaymentMethod paymentMethod, String ccNumber, String cvv, String expMonth, String expYear, boolean virtualTerminal)
- String getFirstName ()
- String getLastName ()
- String getEmail ()
- String getPaymentDescription ()
- Address getBillingAddress ()
- Address getShippingAddress ()
- PaymentMethod getPaymentMethod ()
- Object getManualInfo ()
- boolean isVirtualTerminal ()
- void addEmail (String email)
- String getFullName ()

3.20.1 Detailed Description

The Class PaymentInfo represents all the information obtained via a particular payment method.

3.20.2 Constructor & Destructor Documentation

3.20.2.1 PaymentInfo (String *firstName,* String *lastName,* String *email,* String *paymentDescription,* Address *billingAddress,* Address *shippingAddress,* PaymentMethod *paymentMethod,* String *ccNumber,* String *cvv,* String *expMonth,* String *expYear,* boolean *virtualTerminal*)

Instantiates a new payment info. Use this constructor when representing manually obtained card data. Note: For virtual terminal, name is optional. A placeholder name will be inserted if it is not provided.

Parameters

firstName	the first name
lastName	the last name
email	the email
paymentDescription	the payment description
billingAddress	the billing address
shippingAddress	the shipping address
paymentMethod	the payment method
ccNumber	the cc number
cvv	the cvv
expMonth	the expiration month
expYear	the expiration year
virtualTerminal	the virtual terminal flag

3.20.3 Member Function Documentation

3.20.3.1 void addEmail (String *email*)

Allows adding an email if one is not already present. The call will be ignored if an email is already present.

Parameters

email	the email to be added

3.20.3.2 Address getBillingAddress ( )

Gets the billing address.

Returns

the billingAddress

3.20.3.3 String getEmail ( )

Gets the email.

Returns

the email

3.20.3.4 String getFirstName ( )

Gets the first name.

Returns

the firstName

3.20.3.5 String getFullName ( )

Gets the full name

Returns

full name if available, otherwise null

3.20.3.6 String getLastName ( )

Gets the last name.

Returns

the lastName

3.20.3.7 Object getManualInfo ( )

Gets the manual info.

Returns

the manualInfo

3.20.3.8 String getPaymentDescription ( )

Gets the payment description.

Returns

the paymentDescription

3.20.3.9 PaymentMethod getPaymentMethod ( )

Gets the payment method.

Returns

the paymentMethod

3.20.3.10 Address getShippingAddress ( )

Gets the shipping address.

Returns

the shippingAddress

3.20.3.11 boolean isVirtualTerminal ( )

Determines if the card info was obtained via Virtual Terminal.

Returns

true if virtual terminal, else false

The documentation for this class was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/models/PaymentInfo.java

3.21 PaymentMethod Enum Reference

Public Attributes

- MANUAL =(0)
- SWIPE =(1)
- DIP =(2)

3.21.1 Detailed Description

The Enum PaymentMethod defines all the payment methods available in the sdk.

3.21.2 Member Data Documentation

3.21.2.1 DIP =(2)

Dip

3.21.2.2 MANUAL =(0)

Manual.

3.21.2.3 SWIPE =(1)

Swipe.

The documentation for this enum was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/enums/Payment↩ Method.java

3.22 PaymentToken Class Reference

Public Member Functions

- PaymentToken (String tokenId)
- String getTokenId ()

3.22.1 Detailed Description

The Class PaymentToken represents payment information that was obtained from the user and is stored on WePay servers. This token can be used to complete the payment transaction via WePay's web APIs.

3.22.2 Constructor & Destructor Documentation

3.22.2.1 PaymentToken (String *tokenId*)

Instantiates a new payment token.

Parameters

tokenId	the token id

3.22.3 Member Function Documentation

3.22.3.1 String getTokenId ( )

Gets the token id.

Returns

the token id

The documentation for this class was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/models/PaymentToken.java

3.23 TokenizationHandler Interface Reference

Public Member Functions

- void onSuccess (PaymentInfo paymentInfo, PaymentToken token)
- void onError (PaymentInfo paymentInfo, Error error)

3.23.1 Detailed Description

The Interface TokenizationHandler defines the method used to return data in response to a tokenization call.

3.23.2 Member Function Documentation

3.23.2.1 void onError (PaymentInfo *paymentInfo,* Error *error*)

Gets called when a tokenization call fails.

Parameters

paymentInfo	the payment info.
error	the error due to which tokenization failed.

3.23.2.2 void onSuccess (PaymentInfo *paymentInfo,* PaymentToken *token*)

Gets called when a tokenization calls succeeds.

Parameters

paymentInfo	the payment info passed to the tokenization call.
token	the token representing the payment info.

The documentation for this interface was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/TokenizationHandler.java

3.24 WePay Class Reference

Public Member Functions

- WePay (Config config)
- void startTransactionForReading (final CardReaderHandler cardReaderHandler)
- void startTransactionForTokenizing (final CardReaderHandler cardReaderHandler, final TokenizationHandler tokenizationHandler, final AuthorizationHandler authorizationHandler)
- void stopCardReader ()
- void calibrateCardReader (final CalibrationHandler calibrationHandler)
- void getCardReaderBatteryLevel (final CardReaderHandler cardReaderHandler, final BatteryLevelHandler batteryLevelHandler)
- void tokenize (final PaymentInfo paymentInfo, final TokenizationHandler tokenizationHandler)
- void storeSignatureImage (final Bitmap image, final String checkoutId, final CheckoutHandler checkoutHandler)
- String getRememberedCardReader ()
- void forgetRememberedCardReader ()

3.24.1 Detailed Description

Main Class containing all public endpoints.

3.24.2 Constructor & Destructor Documentation

3.24.2.1 WePay (Config *config*)

Instantiates a new WePay instance.

Parameters

config	the WePay config

3.24.3 Member Function Documentation

3.24.3.1 void calibrateCardReader (final CalibrationHandler *calibrationHandler*)

Use this method to try calibrating a charged card reader that doesn't seem to work on a device. If successful, we will store the calibration parameters on the device, and use them to connect to the card reader in future transactions. This operation only needs to be performed once during first-time setup, and only if the card reader is not automatically detected on a device.

Parameters

calibrationHandler	the calibration handler

3.24.3.2 void forgetRememberedCardReader ( )

Use this method to clear the name of the most recently used card reader.

3.24.3.3 void getCardReaderBatteryLevel (final CardReaderHandler *cardReaderHandler,* final BatteryLevelHandler *batteryLevelHandler*)

Use this method to get the current battery level of the card reader. If no card reader is currently connected, this method will try to find and connect to one.

Parameters

cardReaderHandler	the card reader handler
batteryLevelHandler	the battery level handler

3.24.3.4 String getRememberedCardReader ( )

Use this method to get the name of the most recently used card reader.

Returns

the name of the card reader.

3.24.3.5 void startTransactionForReading (final CardReaderHandler *cardReaderHandler*)

Use this method if you just want to read non-sensitive data from the card, without actually charging the card. Non-sensitive info from the card will be returned via the CardReaderHandler interface.

The reader will wait 60 seconds for a card, and then return a timout error if a card is not detected. The reader will automatically stop waiting for card if:

- a timeout occurs

- a successful swipe/dip is detected
- an unexpected error occurs
- stopReader is called

However, if a general error (errorCategory:ERROR_CATEGORY_CARD_READER, errorCode:CARD_READER_GENERAL_ERROR) occurs while reading, after a few seconds delay, the reader will automatically start waiting again for another 60 seconds. At that time, CardReaderHandler's onStatusChange() method will be called with status = WAITING_FOR_CARD, and the user can try to swipe/dip again. This behavior can be configured with com.wepay.android.models.Config.

WARNING: When this method is called, if the "AUDIOJACK" device is selected via the onCardReaderSelection method in the CardReaderHandler interface, a (normally inaudible) signal is sent to the headphone jack of the phone, where the reader is expected to be connected. If headphones are connected instead of the reader, they may emit a very loud audible tone on receiving this signal. This method should only be called when the user intends to use a reader.

Parameters

cardReaderHandler	the card reader handler

3.24.3.6 void startTransactionForTokenizing (final CardReaderHandler *cardReaderHandler,* final TokenizationHandler *tokenizationHandler,* final AuthorizationHandler *authorizationHandler*)

Use this method if you want to tokenize the card info. Non-sensitive info from the card will be returned via the CardReaderHandler interface. The card info will be tokenized by WePay's servers, and the token will be returned via the TokenizationHandler interface.

The reader will wait 60 seconds for a card, and then return a timout error if a card is not detected. The reader will automatically stop waiting for card if:

- a timeout occurs
- a successful swipe/dip is detected
- an unexpected error occurs
- stopReader is called

However, if a general error (errorCategory:ERROR_CATEGORY_CARD_READER, errorCode:CARD_READER_GENERAL_ERROR) occurs while reading, after a few seconds delay, the reader will automatically start waiting again for another 60 seconds. At that time, CardReaderHandler's onStatusChange() method will be called with status = WAITING_FOR_CARD, and the user can try to swipe/dip again. This behavior can be configured with com.wepay.android.models.Config.

WARNING: When this method is called, if the "AUDIOJACK" device is selected via the onCardReaderSelection method in the CardReaderHandler interface, a (normally inaudible) signal is sent to the headphone jack of the phone, where the reader is expected to be connected. If headphones are connected instead of the reader, they may emit a very loud audible tone on receiving this signal. This method should only be called when the user intends to use a reader.

Parameters

cardReaderHandler	the card reader handler
tokenizationHandler	the tokenization handler
authorizationHandler	the authorization handler

3.24.3.7 void stopCardReader ( )

Stops the reader. In response, CardReaderHandler's onStatusChange() method will be called with status = STOPPED. The status can only be returned if you've provided a CardReaderHandler by starting a card reader operation after the WePay object was initialized. Any operation in progress may not stop, and its result will be delivered to the appropriate handler.

3.24.3.8 void storeSignatureImage (final Bitmap *image,* final String *checkoutId,* final **CheckoutHandler** *checkoutHandler*)

Use this method to store a signature image associated with a checkout id on WePay's servers. The signature can be retrieved via a server-to-server call that fetches the checkout object. The aspect ratio (width:height) of the image must be between 1:4 and 4:1. If needed, the image will internally be scaled to fit inside 256x256 pixels, while maintaining the original aspect ratio.

Parameters

image	the signature image to be stored.
checkoutId	the checkout id associated with the signature
checkoutHandler	the signature handler

3.24.3.9 void tokenize (final **PaymentInfo** *paymentInfo,* final **TokenizationHandler** *tokenizationHandler*)

Use this method to tokenize any PaymentInfo object, such as one representing credit card info obtained manually. The payment info will be tokenized by WePay's servers, and the token will be returned via the TokenizationHandler interface.

Parameters

paymentInfo	the payment info to be tokenized
tokenizationHandler	the tokenization handler

The documentation for this class was generated from the following file:

- /Users/chaitanya.bagaria/sandbox/wepay-android-priv/WePay/app/src/main/java/com/wepay/android/WePay.java

Index

ALL
com::wepay::android::enums::LogLevel, 37
AUTHORIZING
com::wepay::android::enums::CardReaderStatus, 22
addEmail
com::wepay::android::models::PaymentInfo, 45
AuthorizationHandler, 13

BatteryLevelHandler, 15

CARD_BLOCKED
com::wepay::android::enums::ErrorCode, 34
CARD_DECLINED_BY_ISSUER
com::wepay::android::enums::ErrorCode, 34
CARD_DIPPED
com::wepay::android::enums::CardReaderStatus, 22
CARD_NOT_SUPPORTED
com::wepay::android::enums::ErrorCode, 34
CARD_READER_BATTERY_TOO_LOW
com::wepay::android::enums::ErrorCode, 35
CARD_READER_GENERAL_ERROR
com::wepay::android::enums::ErrorCode, 35
CARD_READER_INITIALIZATION_ERROR
com::wepay::android::enums::ErrorCode, 35
CARD_READER_MODEL_NOT_SUPPORTED
com::wepay::android::enums::ErrorCode, 35
CARD_READER_NOT_CONNECTED_ERROR
com::wepay::android::enums::ErrorCode, 35
CARD_READER_STATUS_ERROR
com::wepay::android::enums::ErrorCode, 35
CARD_READER_TIME_OUT_ERROR
com::wepay::android::enums::ErrorCode, 35
CARD_READER_UNABLE_TO_CONNECT
com::wepay::android::enums::ErrorCode, 35
CHECK_CARD_ORIENTATION
com::wepay::android::enums::CardReaderStatus, 22
CHECKING_READER
com::wepay::android::enums::CardReaderStatus, 23
CHIP_ERROR_SWIPE_CARD
com::wepay::android::enums::CardReaderStatus, 23
CONFIGURING_READER
com::wepay::android::enums::CardReaderStatus, 23
CONNECTED
com::wepay::android::enums::CardReaderStatus, 23
calibrateCardReader
com::wepay::android::WePay, 50
CalibrationHandler, 16
CalibrationParameters, 17
CalibrationResult, 17
CardReaderHandler, 18
CardReaderHandler.ApplicationSelectionCallback, 13
CardReaderHandler.CardReaderEmailCallback, 18
CardReaderHandler.CardReaderResetCallback, 21
CardReaderHandler.CardReaderSelectionCallback, 21
CardReaderHandler.CardReaderTransactionInfoCallback, 24
CardReaderStatus, 22
CheckoutHandler, 25
com::wepay::android::AuthorizationHandler
onAuthorizationError, 14
onAuthorizationSuccess, 15
com::wepay::android::BatteryLevelHandler
onBatteryLevel, 15
onBatteryLevelError, 15
com::wepay::android::CalibrationHandler
onComplete, 16
onProgress, 16
com::wepay::android::CardReaderHandler
onCardReaderSelection, 19
onEMVApplicationSelectionRequested, 19
onError, 19
onPayerEmailRequested, 19
onReaderResetRequested, 20
onStatusChange, 20
onSuccess, 20
onTransactionInfoRequested, 20
com::wepay::android::CardReaderHandler::ApplicationSelectionCallback
useApplicationAtIndex, 13
com::wepay::android::CardReaderHandler::CardReaderEmailCallback
insertPayerEmail, 18
com::wepay::android::CardReaderHandler::CardReaderResetCallback
resetCardReader, 21
com::wepay::android::CardReaderHandler::CardReaderSelectionCallback
useCardReaderAtIndex, 21
com::wepay::android::CardReaderHandler::CardReaderTransactionInfoCallback
useTransactionInfo, 24
com::wepay::android::CheckoutHandler
onError, 25
onSuccess, 25
com::wepay::android::TokenizationHandler
onError, 48
onSuccess, 48
com::wepay::android::WePay
calibrateCardReader, 50
forgetRememberedCardReader, 50
getCardReaderBatteryLevel, 50
getRememberedCardReader, 50
startTransactionForReading, 50

startTransactionForTokenizing, 51
stopCardReader, 52
storeSignatureImage, 52
tokenize, 52
WePay, 49
com::wepay::android::enums::CalibrationResult
FAILED, 17
INTERRUPTED, 17
SUCCEEDED, 17
com::wepay::android::enums::CardReaderStatus
AUTHORIZING, 22
CARD_DIPPED, 22
CHECK_CARD_ORIENTATION, 22
CHECKING_READER, 23
CHIP_ERROR_SWIPE_CARD, 23
CONFIGURING_READER, 23
CONNECTED, 23
NOT_CONNECTED, 23
SEARCHING_FOR_READER, 23
SHOULD_NOT_SWIPE_EMV_CARD, 23
STOPPED, 23
SWIPE_DETECTED, 23
SWIPE_ERROR_SWIPE_AGAIN, 23
TOKENIZING, 23
WAITING_FOR_CARD, 24
com::wepay::android::enums::CurrencyCode
USD, 31
com::wepay::android::enums::ErrorCode
CARD_BLOCKED, 34
CARD_DECLINED_BY_ISSUER, 34
CARD_NOT_SUPPORTED, 34
CARD_READER_BATTERY_TOO_LOW, 35
CARD_READER_GENERAL_ERROR, 35
CARD_READER_INITIALIZATION_ERROR, 35
CARD_READER_MODEL_NOT_SUPPORTED, 35
CARD_READER_NOT_CONNECTED_ERROR, 35
CARD_READER_STATUS_ERROR, 35
CARD_READER_TIME_OUT_ERROR, 35
CARD_READER_UNABLE_TO_CONNECT, 35
DECLINED_BY_CARD, 35
EMV_TRANSACTION_ERROR, 35
FAILED_TO_GET_BATTERY_LEVEL, 35
INVALID_APPLICATION_ID, 36
INVALID_CARD_DATA, 36
INVALID_CARD_READER_SELECTION, 36
INVALID_SIGNATURE_IMAGE_ERROR, 36
INVALID_TRANSACTION_ACCOUNT_ID, 36
INVALID_TRANSACTION_AMOUNT, 36
INVALID_TRANSACTION_CURRENCY_CODE, 36
INVALID_TRANSACTION_INFO, 36
ISSUER_UNREACHABLE, 36
NAME_NOT_FOUND_ERROR, 36
NO_DATA_RETURNED_ERROR, 36
PAYMENT_METHOD_CANNOT_BE_TOKENIZED, 37
TRANSACTION_INFO_NOT_PROVIDED, 37
UNKNOWN_ERROR, 37
com::wepay::android::enums::LogLevel
ALL, 37
NONE, 37
com::wepay::android::enums::PaymentMethod
DIP, 47
MANUAL, 47
SWIPE, 47
com::wepay::android::models::Config
Config, 26
ENVIRONMENT_PRODUCTION, 30
ENVIRONMENT_STAGE, 30
getClientId, 27
getContext, 27
getEnvironment, 27
getLogLevel, 27
getMockConfig, 27
isUseLocation, 27
isUseTestEMVCards, 27
setLogLevel, 27
setMockConfig, 28
setRestartTransactionAfterGeneralError, 28
setRestartTransactionAfterOtherErrors, 28
setRestartTransactionAfterSuccess, 29
setStopCardReaderAfterOperation, 29
setUseLocation, 29
setUseTestEMVCards, 29
shouldRestartTransactionAfterGeneralError, 30
shouldRestartTransactionAfterOtherErrors, 30
shouldRestartTransactionAfterSuccess, 30
shouldStopCardReaderAfterOperation, 30
com::wepay::android::models::Error
ERROR_CATEGORY_API, 33
ERROR_CATEGORY_CARD_READER, 33
ERROR_CATEGORY_SDK, 33
ERROR_DOMAIN_API, 33
ERROR_DOMAIN_SDK, 33
getErrorCategory, 32
getErrorCode, 32
getErrorDescription, 32
getErrorDomain, 32
getInnerException, 32
com::wepay::android::models::MockConfig
getMockPaymentMethod, 39
getMockedDeviceName, 39
isBatteryLevelError, 39
isCardReadFailure, 39
isCardReadTimeout, 39
isCardTokenizationFailure, 39
isEMVAuthFailure, 40
isMockCardReaderDetected, 40

isMultipleEMVApplication, 40
isUseMockCardReader, 40
isUseMockWepayClient, 40
MockConfig, 38
setBatteryLevelError, 40
setCardReadFailure, 41
setCardReadTimeout, 41
setCardTokenizationFailure, 41
setEMVAuthFailure, 41
setMockCardReaderDetected, 42
setMockPaymentMethod, 42
setMockedDeviceName, 42
setMultipleEMVApplication, 43
setUseMockCardReader, 43
setUseMockWepayClient, 43
com::wepay::android::models::PaymentInfo
addEmail, 45
getBillingAddress, 45
getEmail, 45
getFirstName, 45
getFullName, 45
getLastName, 45
getManualInfo, 46
getPaymentDescription, 46
getPaymentMethod, 46
getShippingAddress, 46
isVirtualTerminal, 46
PaymentInfo, 44
com::wepay::android::models::PaymentToken
getTokenId, 48
PaymentToken, 47
Config, 26
com::wepay::android::models::Config, 26
CurrencyCode, 31

DECLINED_BY_CARD
com::wepay::android::enums::ErrorCode, 35
DIP
com::wepay::android::enums::PaymentMethod, 47

EMV_TRANSACTION_ERROR
com::wepay::android::enums::ErrorCode, 35
ENVIRONMENT_PRODUCTION
com::wepay::android::models::Config, 30
ENVIRONMENT_STAGE
com::wepay::android::models::Config, 30
ERROR_CATEGORY_API
com::wepay::android::models::Error, 33
ERROR_CATEGORY_CARD_READER
com::wepay::android::models::Error, 33
ERROR_CATEGORY_SDK
com::wepay::android::models::Error, 33
ERROR_DOMAIN_API
com::wepay::android::models::Error, 33
ERROR_DOMAIN_SDK
com::wepay::android::models::Error, 33
Error, 31
ErrorCode, 34

FAILED_TO_GET_BATTERY_LEVEL
com::wepay::android::enums::ErrorCode, 35
FAILED
com::wepay::android::enums::CalibrationResult, 17
forgetRememberedCardReader
com::wepay::android::WePay, 50

getBillingAddress
com::wepay::android::models::PaymentInfo, 45
getCardReaderBatteryLevel
com::wepay::android::WePay, 50
getClientId
com::wepay::android::models::Config, 27
getContext
com::wepay::android::models::Config, 27
getEmail
com::wepay::android::models::PaymentInfo, 45
getEnvironment
com::wepay::android::models::Config, 27
getErrorCategory
com::wepay::android::models::Error, 32
getErrorCode
com::wepay::android::models::Error, 32
getErrorDescription
com::wepay::android::models::Error, 32
getErrorDomain
com::wepay::android::models::Error, 32
getFirstName
com::wepay::android::models::PaymentInfo, 45
getFullName
com::wepay::android::models::PaymentInfo, 45
getInnerException
com::wepay::android::models::Error, 32
getLastName
com::wepay::android::models::PaymentInfo, 45
getLogLevel
com::wepay::android::models::Config, 27
getManualInfo
com::wepay::android::models::PaymentInfo, 46
getMockConfig
com::wepay::android::models::Config, 27
getMockPaymentMethod
com::wepay::android::models::MockConfig, 39
getMockedDeviceName
com::wepay::android::models::MockConfig, 39
getPaymentDescription
com::wepay::android::models::PaymentInfo, 46
getPaymentMethod
com::wepay::android::models::PaymentInfo, 46
getRememberedCardReader
com::wepay::android::WePay, 50

getShippingAddress
com::wepay::android::models::PaymentInfo, 46
getTokenId
com::wepay::android::models::PaymentToken, 48

INTERRUPTED
com::wepay::android::enums::CalibrationResult, 17
INVALID_APPLICATION_ID
com::wepay::android::enums::ErrorCode, 36
INVALID_CARD_DATA
com::wepay::android::enums::ErrorCode, 36
INVALID_CARD_READER_SELECTION
com::wepay::android::enums::ErrorCode, 36
INVALID_SIGNATURE_IMAGE_ERROR
com::wepay::android::enums::ErrorCode, 36
INVALID_TRANSACTION_ACCOUNT_ID
com::wepay::android::enums::ErrorCode, 36
INVALID_TRANSACTION_AMOUNT
com::wepay::android::enums::ErrorCode, 36
INVALID_TRANSACTION_CURRENCY_CODE
com::wepay::android::enums::ErrorCode, 36
INVALID_TRANSACTION_INFO
com::wepay::android::enums::ErrorCode, 36
ISSUER_UNREACHABLE
com::wepay::android::enums::ErrorCode, 36
insertPayerEmail
com::wepay::android::CardReaderHandler::Card↩
ReaderEmailCallback, 18
isBatteryLevelError
com::wepay::android::models::MockConfig, 39
isCardReadFailure
com::wepay::android::models::MockConfig, 39
isCardReadTimeout
com::wepay::android::models::MockConfig, 39
isCardTokenizationFailure
com::wepay::android::models::MockConfig, 39
isEMVAuthFailure
com::wepay::android::models::MockConfig, 40
isMockCardReaderDetected
com::wepay::android::models::MockConfig, 40
isMultipleEMVApplication
com::wepay::android::models::MockConfig, 40
isUseLocation
com::wepay::android::models::Config, 27
isUseMockCardReader
com::wepay::android::models::MockConfig, 40
isUseMockWepayClient
com::wepay::android::models::MockConfig, 40
isUseTestEMVCards
com::wepay::android::models::Config, 27
isVirtualTerminal
com::wepay::android::models::PaymentInfo, 46

LogLevel, 37

MANUAL
com::wepay::android::enums::PaymentMethod, 47
MockConfig, 38
com::wepay::android::models::MockConfig, 38

NAME_NOT_FOUND_ERROR
com::wepay::android::enums::ErrorCode, 36
NO_DATA_RETURNED_ERROR
com::wepay::android::enums::ErrorCode, 36
NONE
com::wepay::android::enums::LogLevel, 37
NOT_CONNECTED
com::wepay::android::enums::CardReaderStatus, 23

onAuthorizationError
com::wepay::android::AuthorizationHandler, 14
onAuthorizationSuccess
com::wepay::android::AuthorizationHandler, 15
onBatteryLevel
com::wepay::android::BatteryLevelHandler, 15
onBatteryLevelError
com::wepay::android::BatteryLevelHandler, 15
onCardReaderSelection
com::wepay::android::CardReaderHandler, 19
onComplete
com::wepay::android::CalibrationHandler, 16
onEMVApplicationSelectionRequested
com::wepay::android::CardReaderHandler, 19
onError
com::wepay::android::CardReaderHandler, 19
com::wepay::android::CheckoutHandler, 25
com::wepay::android::TokenizationHandler, 48
onPayerEmailRequested
com::wepay::android::CardReaderHandler, 19
onProgress
com::wepay::android::CalibrationHandler, 16
onReaderResetRequested
com::wepay::android::CardReaderHandler, 20
onStatusChange
com::wepay::android::CardReaderHandler, 20
onSuccess
com::wepay::android::CardReaderHandler, 20
com::wepay::android::CheckoutHandler, 25
com::wepay::android::TokenizationHandler, 48
onTransactionInfoRequested
com::wepay::android::CardReaderHandler, 20

PAYMENT_METHOD_CANNOT_BE_TOKENIZED
com::wepay::android::enums::ErrorCode, 37
PaymentInfo, 44
com::wepay::android::models::PaymentInfo, 44
PaymentMethod, 47
PaymentToken, 47
com::wepay::android::models::PaymentToken, 47

resetCardReader
com::wepay::android::CardReaderHandler::Card↩ ReaderResetCallback, 21

SEARCHING_FOR_READER
com::wepay::android::enums::CardReaderStatus, 23
SHOULD_NOT_SWIPE_EMV_CARD
com::wepay::android::enums::CardReaderStatus, 23
STOPPED
com::wepay::android::enums::CardReaderStatus, 23
SUCCEEDED
com::wepay::android::enums::CalibrationResult, 17
SWIPE_DETECTED
com::wepay::android::enums::CardReaderStatus, 23
SWIPE_ERROR_SWIPE_AGAIN
com::wepay::android::enums::CardReaderStatus, 23
SWIPE
com::wepay::android::enums::PaymentMethod, 47
setBatteryLevelError
com::wepay::android::models::MockConfig, 40
setCardReadFailure
com::wepay::android::models::MockConfig, 41
setCardReadTimeout
com::wepay::android::models::MockConfig, 41
setCardTokenizationFailure
com::wepay::android::models::MockConfig, 41
setEMVAuthFailure
com::wepay::android::models::MockConfig, 41
setLogLevel
com::wepay::android::models::Config, 27
setMockCardReaderDetected
com::wepay::android::models::MockConfig, 42
setMockConfig
com::wepay::android::models::Config, 28
setMockPaymentMethod
com::wepay::android::models::MockConfig, 42
setMockedDeviceName
com::wepay::android::models::MockConfig, 42
setMultipleEMVApplication
com::wepay::android::models::MockConfig, 43
setRestartTransactionAfterGeneralError
com::wepay::android::models::Config, 28
setRestartTransactionAfterOtherErrors
com::wepay::android::models::Config, 28
setRestartTransactionAfterSuccess
com::wepay::android::models::Config, 29
setStopCardReaderAfterOperation
com::wepay::android::models::Config, 29
setUseLocation
com::wepay::android::models::Config, 29
setUseMockCardReader
com::wepay::android::models::MockConfig, 43
setUseMockWepayClient
com::wepay::android::models::MockConfig, 43
setUseTestEMVCards
com::wepay::android::models::Config, 29
shouldRestartTransactionAfterGeneralError
com::wepay::android::models::Config, 30
shouldRestartTransactionAfterOtherErrors
com::wepay::android::models::Config, 30
shouldRestartTransactionAfterSuccess
com::wepay::android::models::Config, 30
shouldStopCardReaderAfterOperation
com::wepay::android::models::Config, 30
startTransactionForReading
com::wepay::android::WePay, 50
startTransactionForTokenizing
com::wepay::android::WePay, 51
stopCardReader
com::wepay::android::WePay, 52
storeSignatureImage
com::wepay::android::WePay, 52

TOKENIZING
com::wepay::android::enums::CardReaderStatus, 23
TRANSACTION_INFO_NOT_PROVIDED
com::wepay::android::enums::ErrorCode, 37
TokenizationHandler, 48
tokenize
com::wepay::android::WePay, 52

UNKNOWN_ERROR
com::wepay::android::enums::ErrorCode, 37
USD
com::wepay::android::enums::CurrencyCode, 31
useApplicationAtIndex
com::wepay::android::CardReaderHandler::↩ ApplicationSelectionCallback, 13
useCardReaderAtIndex
com::wepay::android::CardReaderHandler::Card↩ ReaderSelectionCallback, 21
useTransactionInfo
com::wepay::android::CardReaderHandler::Card↩ ReaderTransactionInfoCallback, 24

WAITING_FOR_CARD
com::wepay::android::enums::CardReaderStatus, 24
WePay, 49
com::wepay::android::WePay, 49