Android SDK Documentation

January 27

By Midtrans

Table Of Contents

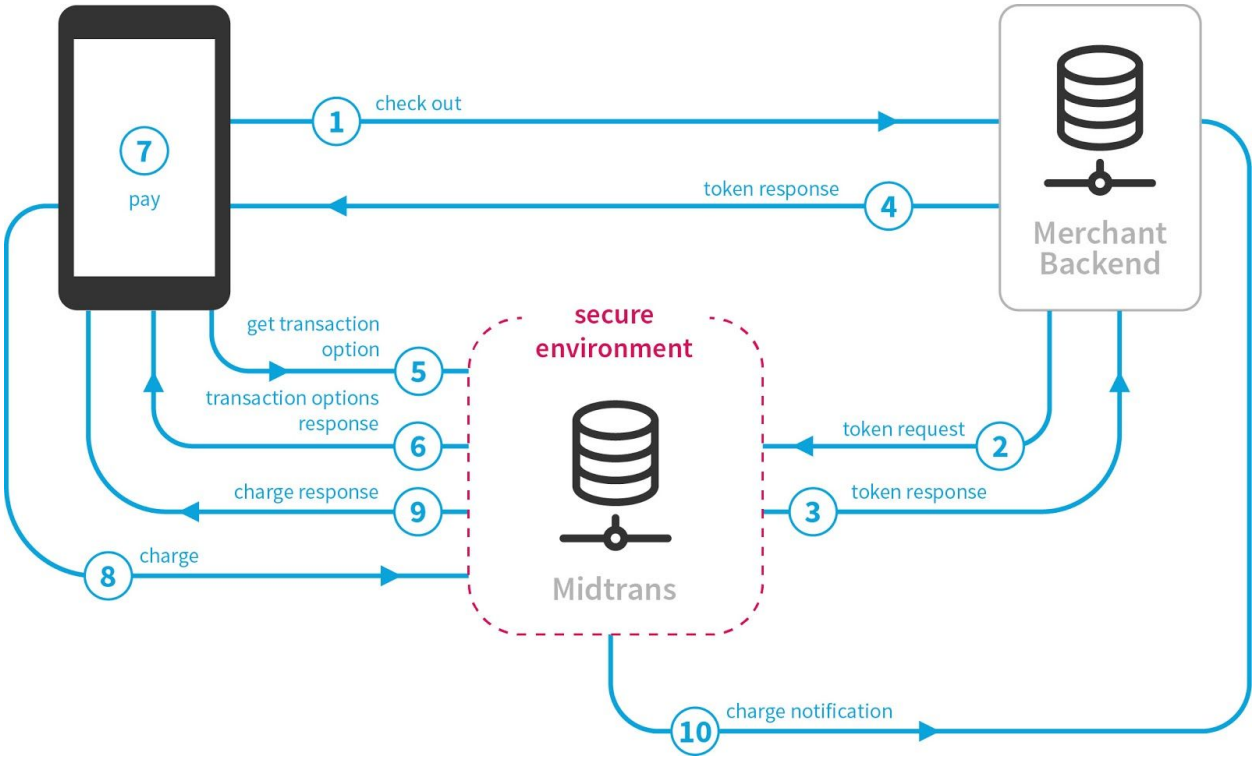
Getting Started

Midtrans mobile SDK enable merchants to accept online payments natively in their mobile apps. We provide the drop in User interface for making transactions on all the payment types supported by Midtrans. Watch the video for the default SDK example.

There are four parties involved in the payment process for making a payment:

1. Merchant Server: The merchant backend implementation
2. Customers
3. Midtrans Backend (Payment Processor)
4. Midtrans Mobile SDK

Transaction flow



1. Checkout: Customer clicks the Checkout button on the Host application and the app makes a request to the Merchant Server
2. Token request: Merchant Server makes a request to Veritrans server with Order Information.
3. Token response: Midtrans responds with a valid transaction token to Merchant server
4. Token response: Merchant server forwards the token to the Mobile SDK
5. Get transaction options: Mobile SDK requests payment/merchant information based on the token

6. Transaction options response: Mobile SDK renders the payment Options and payment information to make the payment
7. Pay: Customers selects the payment method and the payment details and clicks "Pay"
8. Charge: Mobile SDK sends the Charge request to the Veritrans Backend for payment Processing.
9. Charge response: Mobile SDK receives the response from the Veritrans Backend and triggers the handler on Mobile App with success/failure/pending status
10. Charge notification: Midtrans Backend sends a notification to the Merchant backend confirming the completion of transaction.

Supported Payment Methods

1. Credit/Debit Cards - Support for making payments via credit cards and or debit cards using our two-clicks feature. We support Visa, Mastercard, American Express and JCB
2. Mandiri ClickPay
3. CIMB Clicks
4. ePay BRI
5. Indosat Dompetku
6. Mandiri e-Cash
7. Bank Transfer - Support payment using Permata Virtual Account and BCA Virtual Account.
8. Mandiri Bill Payment
9. Indomaret - Payment via convenience Stores
10. BCA Klikpay
11. KlikBCA
12. Kioson
13. Gift Card Indonesia

Security Aspects

- There are 2 separate keys CLIENT_KEY and SERVER_KEY (available on MAP)
    - CLIENT_KEY is used for tokenizing the credit card. It can only be used from the Client(mobile device)
    - SERVER_KEY is used for acquiring the token from the Midtrans server. It is not to be used from the device, all API requests that use the SERVER_KEY need to be made from the Merchant Server.
- We use strong encryption for making connections to Merchant server, please make sure it has valid https Certificate.

Following are configurable parameters of SDK that can be used while performing transaction -

- Merchant server Endpoint / Base URL : URL of server to which transaction data will be sent. This will also be referred to as a merchant server.
- Transaction details - contains payment information like amount, order Id, payment

method etc.
- Midtrans Client Key - token that specified by merchant server to enable the transaction using credit card. Available on the MAP

Prerequisites

1. Create a merchant account in MAP
2. In MAP, setup your merchant accounts settings, in particular Notification URL.
3. Setup your merchant server. A server side implementation is required for midtrans mobile SDK to work. You can check the server implementation reference, and walk through the API's that you may need to implement on your backend server.
4. Minimum requirements: AndroidSDK: Android 4.0 Ice Cream Sandwich API Level 14

Android SDK

This SDK provides an UI to take required information from user to execute transaction.

Installation

Add Midtrans Bintray repository

```
repositories {
jcenter()
// Add the midtrans repository into the list of repositories
maven
{
    url "http://dl.bintray.com/pt-midtrans/maven" }
}
```

Add SDK installation following into your build.gradle

You need to add Midtrans SDK inside your app's module build.gradle.

SDK Sandbox Dependencies

```
dependencies {
// For using the Midtrans Sandbox compile
'com.midtrans:uikit:$VERSION-SANDBOX'
}
```

SDK Production Dependencies

```
dependencies {
// For using the Midtrans Production
compile 'com.midtrans:uikit:$VERSION'
}
```

Then you need to initialize it on your activity or application class.

```
SdkUIFlowBuilder.init(CONTEXT, CLIENT_KEY, BASE_URL, new
TransactionFinishedCallback() {
                @Override
               public void onTransactionFinished(TransactionResult result) {
                    // Handle finished transaction here.
               }
            })
            .buildSDK();
```

Note:

- CONTEXT: Application/activity context
- CLIENT_KEY: Your midtrans client key (provided in MAP)
- BASE_URL: Your merchant server URL

Differentiate Sandbox and Production in one app (Optional)

You can support two payment environments in your app by defining two flavors in your build.gradle.

```
android {
...
// Define Merchant BASE URL and CLIENT KEY for each flavors
productFlavors {
    sandbox {
        buildConfigField "String", "BASE_URL",
"\"https://merchant-url-sandbox.com/\""
        buildConfigField "String", "CLIENT_KEY",
"\"VT-CLIENT-sandbox-client-key\""
    }
    production {
        buildConfigField "String", "BASE_URL",
"\"https://merchant-url-production.com/\""
        buildConfigField "String", "CLIENT_KEY",
"\"VT-CLIENT-production-client-key\""
    }
}
...
}

// Define Midtrans SDK dependencies for each flavors
dependencies {
...
sandboxCompile 'com.midtrans:uikit:$VERSION-SANDBOX'
productionCompile 'com.midtrans:uikit:$VERSION'
...
}
```

Initialize your SDK using merchant BASE_URL and CLIENT_KEY provided by BuildConfig data.

```
SdkUIFlowBuilder.init(CONTEXT, BuildConfig.CLIENT_KEY,
BuildConfig.BASE_URL, new TransactionFinishedCallback() {
            @Override
           public void onTransactionFinished(TransactionResult
result) {
```

```
                // Handle finished transaction here.
            }
        })
        .buildSDK();
```

Initialize Midtrans SDK using provided base URL and client key in BuildConfig

```
SdkUIFlowBuilder.init(CONTEXT, BuildConfig.CLIENT_KEY, BuildConfig.BASE_URL,
new TransactionFinishedCallback() {
 @Override public void onTransactionFinished(TransactionResult result) {
    // Handle finished transaction here. }
})
.buildSDK();
```

Adding external card scanner (Optional)

We provide a plugin to integrate card.io for allowing customers to read the credit card/debit card information using the mobile phone camera.

You can add external card scanner using ScanCardLibrary implementation by midtrans scan card library into your app's dependencies in build.gradle.

```
//..other dependencies compile ('com.midtrans:scancard:$VERSION'){
   exclude module: 'uikit'
}
```

Then, when initialize the SDK you can setExternalScanner(new ScanCard()) on SdkUIFlowBuilder

```
SdkUIFlowBuilder.init(...)
// initialization for using external scancard .setExternalScanner(new
ScanCard()) .buildSDK();
```

Prepare Transaction Details

TRANSACTION_ID and **TOTAL_AMOUNT** was required to create a transaction request that was required for each payment.

Create Transaction Request object

```
TransactionRequest transactionRequest = new TransactionRequest(TRANSACTION_ID
TOTAL_AMOUNT);
```

Item Details

Item details was required for Mandiri Bill and BCA KlikPay. It's optional for other payment.

ItemDetails class holds information about item purchased by user. TransactionRequest takes an array list of item details.

```
ItemDetails itemDetails1 = new ItemDetails(ITEM_ID_1, ITEM_PRICE_1,
ITEM_QUANTITY_1, ITEM_NAME_1);

ItemDetails itemDetails2 = new ItemDetails(ITEM_ID_2, ITEM_PRICE_2,
ITEM_QUANTITY_2, ITEM_NAME_2);

// Create array list and add above item details in it and then set it to
transaction request.

ArrayList<ItemDetails> itemDetailsList = new ArrayList<>();
itemDetailsList.add(itemDetails1); itemDetailsList.add(itemdetails2);

// Set item details into the transaction request.
transactionRequest.setItemDetails(itemDetailsList);
```

Note:

- This was assumed that you have created transactionRequest object using required parameters.
- ITEM_NAME maximum character length is 50.

Bill Info

Bill Info was optional on Mandiri Bill payment only.

BillInfoModel class holds information about billing information that will be shown at billing details.

```
BillInfoModel billInfoModel = new BillInfoModel(BILL_INFO_KEY,
BILL_INFO_VALUE); // Set the bill info on transaction details
transactionRequest.setBillInfoModel(billInfoModel);
```

Set Transaction Request into SDK Instance

After creating transaction request with optional fields above, you must set it into SDK instance.

```
MidtransSDK.getInstance().setTransactionRequest(transactionRequest);
```

Starting Payment

Payment Mode

If you support credit card payment, you must select one of three card click types.

Payment Mode

```
CreditCard creditCardOptions = new CreditCard();

// Set to true if you want to save card to Snap
creditCardOptions.setSaveCard(false);

// Set to true to save card token as `one click` token
creditCardOptions.setSecure(false);

// Set acquiring bank (Optional)
creditCardOptions.setBank(BankType.BANK_NAME);

// Set MIGS channel (ONLY for BCA and Maybank Acquiring bank)
creditCardOptions.setChannel(CreditCard.MIGS);

// Set Credit Card Options
transactionRequest.setCreditCard(creditCardOptions);

// Set card payment info
transactionRequest.setCardPaymentInfo(CARD_CLICK_TYPE, IS_SECURE);

// Set transaction request into SDK instance
```

```
MidtransSDK.getInstance().setTransactionRequest(transactionRequest);
```

Note:

- **CARD_CLICK_TYPE** - type of card use these resource strings:
  - **normal** - for normal transaction
  - **one_click** - for one click
  - **two_click** - for two click
- **IS_SECURE** - set to true if using 3D secure

Start payment method screen

Default mode for Android SDK is showing payment method screen. This screen will show available payment method.

You can set which payment method that's available using Snap Preferences in MAP.

```
MidtransSDK.getInstance().startPaymentUiFlow(ACTIVITY_CONTEXT);
```

Direct payment screen

Users can directly go to payment screen and skip the default payment method screen.

Note: Please make sure the payment method is activated via Setting -> Snap Preferences in MAP.

Start Direct Payment Screen

Start credit card payment

```
MidtransSDK.getInstance().startCreditCardUIFlow(ACTIVITY_CONTEXT);
```

Start Bank transfer payment

```
MidtransSDK.getInstance().startBankTransferUIFlow(ACTIVITY_CONTEXT);
```

Start Permata bank transfer payment

```
MidtransSDK.getInstance().startPermataBankTransferUIFlow(ACTIVITY_CONTEXT);
```

Start BCA bank transfer payment

```
MidtransSDK.getInstance().startBCABankTransferUIFlow(ACTIVITY_CONTEXT);
```

Start Mandiri bank transfer payment

```
MidtransSDK.getInstance().startMandiriBankTransferUIFlow(ACTIVITY_CONTEXT);
```

Start Other bank transfer payment

```
MidtransSDK.getInstance().startOtherBankTransferUIFlow(ACTIVITY_CONTEXT);
```

Start Klik BCA payment

```
MidtransSDK.getInstance().startKlikBCAUIFlow(ACTIVITY_CONTEXT);
```

Start BCA KlikPay payment

```
MidtransSDK.getInstance().startBCAKlikPayUIFlow(ACTIVITY_CONTEXT);
```

Start Mandiri Clickpay payment

```
MidtransSDK.getInstance().startMandiriClickpayUIFlow(ACTIVITY_CONTEXT);
```

Start Mandiri E-Cash payment

```
MidtransSDK.getInstance().startMandiriECashUIFlow(ACTIVITY_CONTEXT);
```

Start CIMB Clicks payment

```
MidtransSDK.getInstance().startCIMBClicksUIFlow(ACTIVITY_CONTEXT);
```

Start BRI Epay payment

```
MidtransSDK.getInstance().startBRIEpayUIFlow(ACTIVITY_CONTEXT);
```

Start Telkomsel Cash payment

```
MidtransSDK.getInstance().startTelkomselCashUIFlow(ACTIVITY_CONTEXT);
```

Start Indosat Dompetku payment

```
MidtransSDK.getInstance().startIndosatDompetkuUIFlow(ACTIVITY_CONTEXT);
```

Start XL Tunai payment

```
MidtransSDK.getInstance().startXlTunaiUIFlow(ACTIVITY_CONTEXT);
```

Start Indomaret payment

```
MidtransSDK.getInstance().startIndomaretUIFlow(ACTIVITY_CONTEXT);
```

Start Kioson payment

```
MidtransSDK.getInstance().startKiosonUIFlow(ACTIVITY_CONTEXT);
```

Start Gift Card payment

```
MidtransSDK.getInstance().startGiftCardUIFlow(ACTIVITY_CONTEXT);
```

Payment Result

TransactionResult is wrapper for UI flow finished transaction object. It contains:

- status : either pending, success, failed or invalid based on payment API.
- transactionResponse: contains payment response from Payment API.
- transactionCanceled : this will be set to true only if transaction was canceled from inside SDK. For example when selecting payment method users click back.

Here the step:

- First one to check is transactionCanceled. If this was set to true then you don't have to check other field.
- You can check based on status: pending will be only use on asynchronous transaction like bank transfer or internet banking.
- You can use API to get transaction status or wait for notification comes to your backend to ensure the latest status of the transaction.
    - success / failed: For synchronous transaction you can immediately know the status of the transaction.
    - invalid : There are unknown error happened. transactionResponse for detailed

transaction response from Payment API.

Additional Payment Features

Credit card payment on this SDK can be customized to save customer credit card details so user can use it on their next transaction.

Two Clicks Payment

For the payment, you need to setup this configuration to enable the two clicks mode.

```
CreditCard creditCardOptions = new CreditCard();

// Set to true if you want to save card

creditCardOptions.setSaveCard(true);

// Set to false to save card token as `two clicks` token
creditCardOptions.setSecure(false);

// Set Credit Card Options
transactionRequest.setCreditCard(creditCardOptions);

// Set card payment info and 3DS enabled here
transactionRequest.setCardPaymentInfo("two_click", false);

// Set transaction request into SDK instance
MidtransSDK.getInstance().setTransactionRequest(transactionRequest);
```

To use two clicks configuration, merchant can use both default token storage on Midtrans backend or use their server to store their customer credential.

Default Token Storage Usage

By default this SDK will use Midtrans token storage to save customer credential so you don't need to setup anything.

Store Token on Merchant Server

Please take a look at this guide to see save card feature implementation in merchant server.

Then you need configure SDK to disable built in token storage.

```
SdkUIFlowBuilder.init(CONTEXT, CLIENT_KEY, BASE_URL, CALLBACK)

// disable built in token storage .useBuiltInTokenStorage(false) .buildSDK();
```

One Click Payment

For one click payment you can only use built token storage as credential storage option.

One click payment configuration

```
CreditCard creditCardOptions = new CreditCard();

// Set to true if you want to save card

creditCardOptions.setSaveCard(true);

// Set to true to save card token as `one click` token
creditCardOptions.setSecure(true);

// Set Credit Card Options
transactionRequest.setCreditCard(creditCardOptions);

// Set card payment info, 3DS must be enabled
transactionRequest.setCardPaymentInfo("one_click", true);

// Set transaction request into SDK instance
MidtransSDK.getInstance().setTransactionRequest(transactionRequest);
```

UI Customization

Custom Fonts

To apply Custom fonts, you can use this code.

Custom Fonts

```
MidtransSDK midtransSDK = MidtransSDK.getInstance();
midtransSDK.setDefaultText("open_sans_regular.ttf");
midtransSDK.setSemiBoldText("open_sans_semibold.ttf");
midtransSDK.setBoldText("open_sans_bold.ttf");
```

Note: open_sans_regular.ttf, open_sans_semibold.ttf, open_sans_bold.ttf is path of the custom font on the assets directory.

Custom Themes

Also you can set the color primary in your theme in styles.xml.

```
<!-- Base application theme. --> <style name="AppTheme"
parent="Theme.AppCompat.Light.NoActionBar">

<item name="colorPrimary">@color/colorPrimary</item> <item
```

```
name="colorAccent">@color/colorAccent</item> <item
name="colorPrimaryDark">@color/colorPrimaryDark</item> <item
name="colorButtonNormal">@color/colorButton</item> </style>
```

Then to ensure this replace library theme, please add these lines into your AndroidManifest.xml application tag.

```
<application
    ...
    android:theme="AppTheme"
    tools:replace="android:theme">
```

Skip Customer Details Screen (Optional)

On the first SDK usage, user needs to fill customer details required by payment.

You can skip this screen if you want by following this guide.

```
// Set user details
UserDetail userDetail = new UserDetail();
userDetail.setUserFullName(FULL_NAME);
userDetail.setEmail(EMAIL);
userDetail.setPhoneNumber(PHONE_NUMBER);
userDetail.setUserId(USER_ID);
// Initiate address list
ArrayList<UserAddress> userAddresses = new ArrayList<>();
// Initiate and add shipping address
UserAddress shippingUserAddress = new UserAddress();
shippingUserAddress.setAddress(shippingAddress);
shippingUserAddress.setCity(shippingCity);
shippingUserAddress.setCountry(shippingCountry);
shippingUserAddress.setZipcode(shippingZipcode);
shippingUserAddress.setAddressType(Constants.ADDRESS_TYPE_SHIPPING);
userAddresses.add(shippingUserAddress);

// Initiate and add billing address
```

```
UserAddress billingUserAddress = new UserAddress();

billingUserAddress.setAddress(billingAddress);
billingUserAddress.setCity(billingCity);
billingUserAddress.setCountry(country);
billingUserAddress.setZipcode(zipcode);
billingUserAddress.setAddressType(Constants.ADDRESS_TYPE_BILLING);

userAddresses.add(billingUserAddress);

// if shipping address is same billing address

// you can user type Constants.ADDRESS_TYPE_BOTH

// NOTE: if you use this, skip initiate shipping and billing address above

UserAddress userAddress = new UserAddress();
userAddress.setAddress(billingAddress);

userAddress.setCity(billingCity);

userAddress.setCountry(country);

userAddress.setZipcode(zipcode);
userAddress.setAddressType(Constants.ADDRESS_TYPE_BOTH);
userAddresses.add(userAddress);


// Set user address to user detail object
userDetail.setUserAddresses(userAddresses);

// Save the user detail. It will skip the user detail screen
LocalDataHandler.saveObject("user_details", userDetail);
```

Note:

- All user details is required to make transactions.
- Minimum one address is required to make transactions.
- USER_ID is required to enable save card and one-click/two-clicks.

UIKitCustomSetting

We provide UIKitCustomSetting to handle more customizable UI in our SDK.

Skip Payment Status

You can skip payment status provided by Midtrans SDK if you want to show your own status page.

```
// Init custom settings UIKitCustomSetting uisetting = new
UIKitCustomSetting(); uisetting.setShowPaymentStatus(true);
```

```
MidtransSDK.getInstance().setUIKitCustomSetting(uiKitCustomSetting);
```

Set Default save card options to true

In credit card payment page, there is checkbox to save card and it's not checked by default. You can make this checkbox checked by default by using this settings.

```
// Init custom settings UIKitCustomSetting uisetting = new
UIKitCustomSetting(); uiKitCustomSetting.setSaveCardChecked(true);
MidtransSDK.getInstance().setUIKitCustomSetting(uiKitCustomSetting);
```