DOKU iOS SDK DOCUMENTATION

Version 1

June 2016

PT Nusa Satu Inti Artha
Plaza Asia Office Park Unit 3
Jl. Jenderal Sudirman Kav. 59
Jakarta 12190 Indonesia

Table of Contents

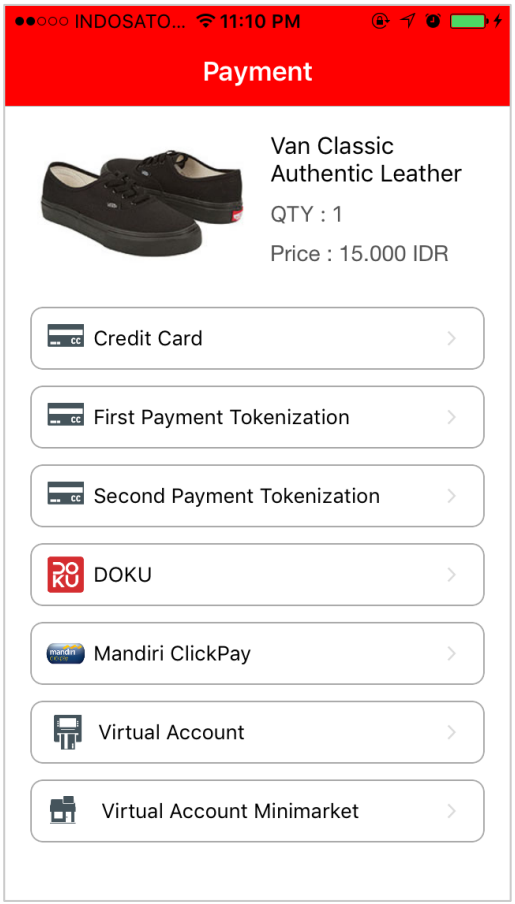
1.0 Introduction

This document will act as a tutorial to help you integrate your iOS mobile application to DOKU API and start receiving payments from mobile transactions. The DOKU Mobile SDK enables you to accept payments from customers who make a purchase on mobile devices through your iOS application, and currently supports the following payment methods:

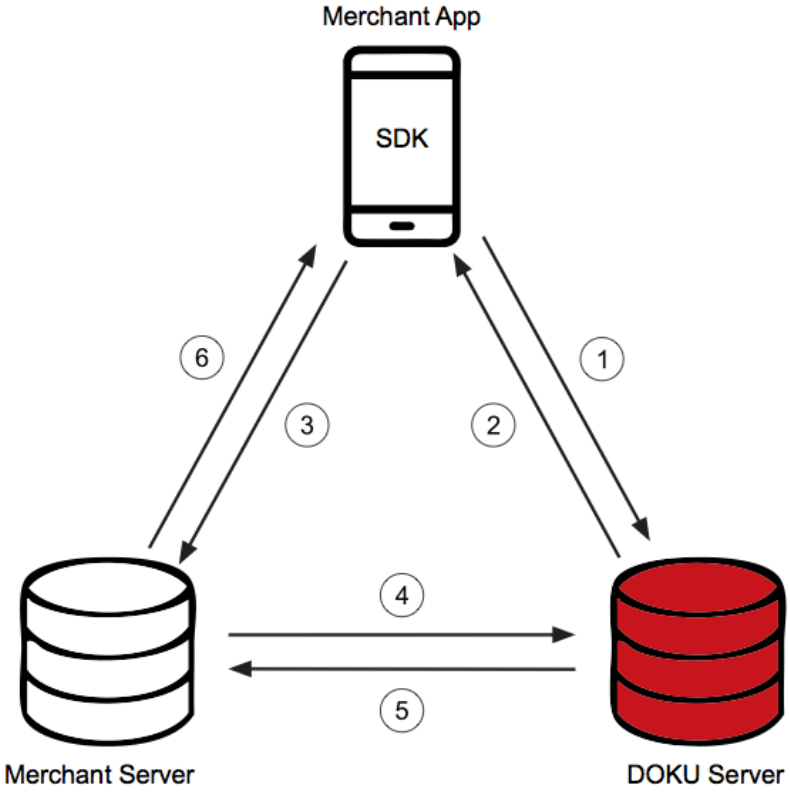
- Credit Card
- DOKU Wallet
- Bank Transfer
- Convenience Store
- Mandiri Clickpay

The payment page and data input is native to the merchant's mobile app, without having to redirect to a DOKU-hosted page. Having the payment form on the merchant page does not compromise the security of the cardholder however, as DOKU is PCI (Level 1) certified, and none of the cardholder data will actually be stored on the merchant's server.

The DOKU API only provides the fields to be filled in by the customer. All other parts of the payment page, including the logo, colour and 'Process Payment' button are customizable to your needs for a completely white label payment flow. Learn how to customize your page in Section 6.0. Once you have installed your SDK, the payment page will look something like this:



1.1 Payment Flow



1. When the customer inputs their card/payment data into the payment form on your app, the embedded SDK will send a token request with the card information to the DOKU server.
2. The DOKU server will generate a **cc_token** and **pairing_code** and send the response to the DOKU SDK within the merchant app. Through this process, none of the credit card data is captured by the merchant, and everything is securely processed by DOKU.
3. The merchant app sends the card data to the merchant server, according to the action parameter in the SDK.
4. The merchant server will send a payment request to DOKU, containing the payment data such as price, customer information and the token.
5. DOKU processes the payment and sends a response in JSON format.
6. The merchant displays the payment result on the app, according to the response sent by DOKU.

See Appendix (Section 7.0) for response codes.

1.2 Integration

The following section gives an example of how you can integrate the various payments into your iOS mobile application. Once you have confirmed to become a DOKU merchant through our Sales process, you will be contacted by our integration team to proceed to the technical integration stage. All new merchants will receive a *shared key* and a *merchant code*. Take note of this information as you will need to enter them into the API script during integration. The response codes are categorized by payment method, and can be found in the appendix.

The instructions are divided into separate sections for each payment method, as the integration process will differ for each method. Credit Card and DOKU Wallet payments use a mobile SDK, while Bank Transfer, Convenience Store and Internet Banking payments use the Merchant-hosted web API.

iOS SDK integration has the following requirements:

- iPhone 4s, 5, and 6 with at least iOS 9.x
- iPad with at least iOS 9.x

2.0 Credit Card

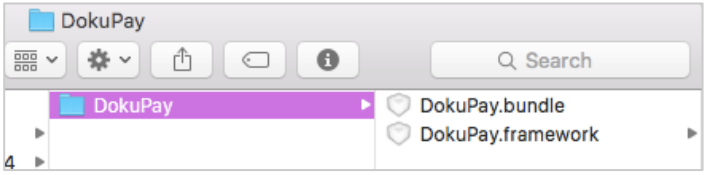
By default all credit card payments processed by DOKU will undergo 3D secure. Non-3D secure payments are available, however would require further assessment by DOKU and the bank.

Credit card integration comprises 3 steps:

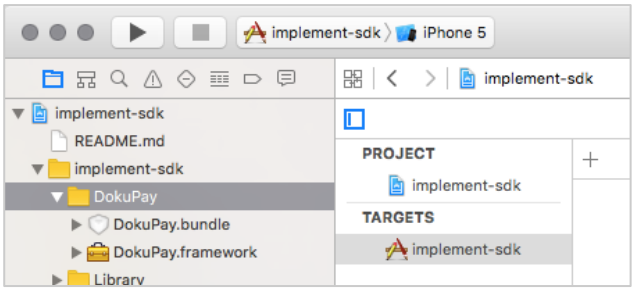
1. Initialize the DOKU iOS SDK
2. Create dependencies
3. Retrieve token
4. Send payment request

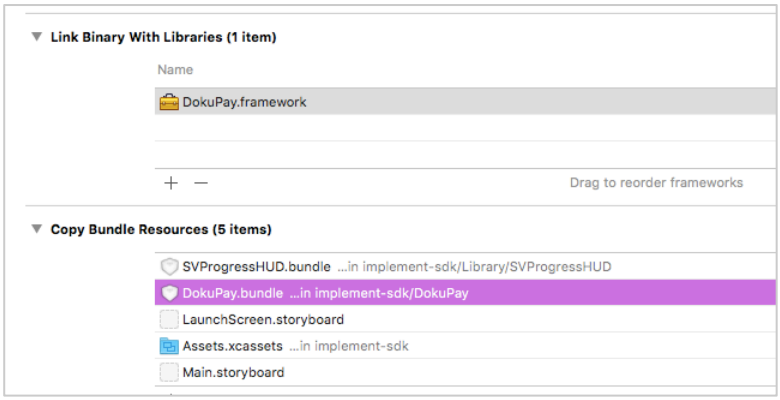
1. Initializing the DOKU iOS SDK

a. Download the DokuPay folder containing DokuPay.framework and DokuPay.bundle from http://doku.com/iOS-OCO/SDKOCO-DummyMerchant_V2-iOS.zip and copy it into your App's project folder. See example:



Select the project folder and insert the copied DokuPay folder. Ensure that the DokuPay.framework is within the 'General' -> 'Linked Frameworks and Libraries' list and the DokuPay.bundle is within the 'Build Phases' -> 'Copy Bundle Resources' list.



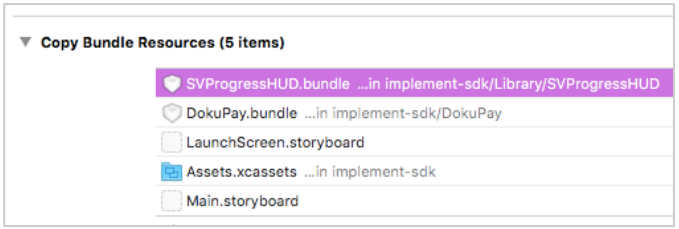


Call the DokuPay.h class from the framework using '#import' then add '' in the interface class project:

```
//

#import "ViewController.h"
#import <DokuPay/DokuPay.h>
#import "AFNetworking.h"
#import "SVProgressHUD.h"

@interface ViewController () <DokuPayDelegate>
{
    DKLayout *dokuLayout;
}
@end

@implementation ViewController
```

2. Create dependencies

Add SVProgressHUD from https://github.com/SVProgressHUD/SVProgressHUD and install it to your project folder. Ensure that the SVProgressHUD.bundle is within the 'Build Phases' -> 'Copy Bundle Resources'



3. Retrieve Token

a. Below is an example code for the parameters which will be sent by the merchant app to the SDK (see the **Payment Flow** diagram for details). Set credit card as the chosen payment channel. See the Appendix (Section 7.2) for parameter format details.

```
-(DKPaymentItem*)getPaymentItem
{
DKPaymentItem *paymentItem = [[DKPaymentItem alloc] init];
NSArray *basket =
@[@{@"name":@"sayur",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"},
@{@"name":@"buah",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"}];

paymentItem.dataAmount = @"15000.00";
paymentItem.dataBasket = basket;
paymentItem.dataImei = [self getMyUUID];
paymentItem.dataCurrency = @"360";
paymentItem.dataMerchantChain = @"NA";
paymentItem.dataSessionID = … session ID app merchant ... ;
paymentItem.dataTransactionID = … kode transaksi merchant ...;
paymentItem.isProduction = false;
paymentItem.dataMerchantCode = MerchantSharedMallID;
paymentItem.publicKey = MerchantPublicKey;
paymentItem.sharedKey = MerchantSharedKey;
paymentItem.dataWords = [paymentItem generateWords];
paymentItem.mobilePhone = @"08123123112";

[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeCC];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];

return paymentItem;
}
```

b. Get device ID by using the script below.

```
-(NSString*)getMyUUID
{
NSString *uuid = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
return [uuid stringByReplacingOccurrencesOfString:@"-" withString:@""];
}
```

c. In order for your iOS mobile app to receive response from the SDK, you will need to create a method shown in the example below:

```
-(void)onDokuPaySuccess:(NSDictionary *)dictData
{
NSLog(@"catch Success delegate : %@", dictData);
}

-(void)onDokuPayError:(NSError *)error
{
NSLog(@"catch Error delegate : %@", error);
}
```

d. Once you have entered all the conditions above correctly, the merchant app should receive the following response token from the SDK which indicates success.

```
{
  "res_token_id": "53beb0d8da617828d1c6295d822d84b4a12c33ea",
  "res_pairing_code": "07041613200336243915",
  "res_name": "test",
  "res_data_email": "test@mail.com",
  "res_response_msg": "SUCCESS",
  "res_device_id": "867804025368595",
  "res_token_code": "0000",
  "res_amount": "15000.00",
  "res_response_code": "0000",
  "res_payment_channel": "15",
  "res_transaction_id": "1410757974"
}
```

e. After getting your token, there will be instances where your app send it forward to your server, for example for charging payments. To do this, please see the script below as an example.

```
NSURL *URL = [NSURL URLWithString:@"http://crm.doku.com/doku-library/example-payment-
mobile/merchant-example.php "];

NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:URL];
[request setValue:@"application/x-www-form-urlencoded" forHTTPHeaderField:@"Content-Type"];
[request setHTTPMethod:@"POST"];

NSString *params = [NSString stringWithFormat:@"data=%@", [self jsonStringWithPrettyPrint:NO
fromDictionary:dictData]];

NSData *data = [params dataUsingEncoding:NSUTF8StringEncoding];
[request setHTTPBody:data];

AFHTTPSessionManager *manager = [AFHTTPSessionManager manager];
manager.responseSerializer.acceptableContentTypes = [NSSet setWithObject:@"text/html"];

NSURLSessionUploadTask *task = [manager uploadTaskWithStreamedRequest:request progress:nil
completionHandler:^(NSURLResponse * _Nonnull response, id  _Nullable responseObject, NSError *
_Nullable error) {

[SVProgressHUD dismiss];

[self popError:error withReponse:responseObject];
}];
[SVProgressHUD setDefaultMaskType:SVProgressHUDMaskTypeGradient];
[SVProgressHUD showWithStatus:@"Mohon Tunggu..."];
[task resume];
```

4. Send Payment Request

a. Your app will send the data response to your server, which you can pass on to the DOKU server to process the payment:

```
<?php
require_once('../Doku.php');

Doku_Initiate::$sharedKey = '<Put Your Shared Key Here>';
Doku_Initiate::$mallId = '<Put Your Merchant Code Here>';

$token = $_POST['doku-token'];
$pairing_code = $_POST['doku-pairing-code'];
$invoice_no = $_POST['doku-invoice-no'];

$params = array(
    'amount' => '10000.00',
    'invoice' => $invoice_no,
    'currency' => '360',
    'pairing_code' => $pairing_code,
    'token' => $token
);

$words = Doku_Library::doCreateWords($params);

$basket[] = array(
    'name' => 'sayur',
    'amount' => '10000.00',
    'quantity' => '1',
    'subtotal' => '10000.00'
);

$customer = array(
    'name' => 'TEST NAME',
    'data_phone' => '08121111111',
    'data_email' => 'test@test.com',
    'data_address' => 'bojong gede #1 08/01'
);

    $dataPayment = array(
        'req_mall_id' => Doku_Initiate::$mallId,
        'req_chain_merchant' => 'NA',
        'req_amount' => '10000.00',
        'req_words' => $words,
        'req_purchase_amount' => '10000.00',
        'req_trans_id_merchant' => $invoice_no,
        'req_request_date_time' => date('YmdHis'),
        'req_currency' => '360',
        'req_purchase_currency' => '360',
        'req_session_id' => sha1(date('YmdHis')),
        'req_name' => $customer['name'],
        'req_payment_channel' => 15,
        'req_basket' => $basket,
        'req_email' => $customer['data_email'],
        'req_token_id' => $token
    );

    $result = Doku_Api::doPayment($dataPayment);

    if($result->res_response_code == '0000'){
        echo 'SUCCESS';
        //success
    }else{
        echo 'FAILED';
        //failed
    }
```

b. If you have successfully charged payments from a credit card, you will receive a response from DOKU Server as shown below. **(see response codes definitions in Appendix)**

Example success response for payments made using a credit card:


```
{
  "res_tid": "13019501",
  "res_trx_code": "ae9e54a6f0788c41a8c4d468f83fc7c268fc0725",
  "res_currency": "IDR",
  "res_approval_code": "900059",
  "res_eci": "",
  "res_chain_mall_id": "",
  "res_card_number": "4***********1111",
  "res_amount": "15000.00",
  "res_message": "PAYMENT APPROVED",
  "res_issuer_bank": "JPMORGAN CHASE BANK",
  "res_mall_id": "2",
  "res_liability": "MERCHANT",
  "res_mid": "000100013000195",
  "res_result": "SUCCESS",
  "res_payment_date": "20160411180526",
  "res_three_d_secure_status": "FALSE",
  "res_bank": "BNI",
  "res_invoice_number": "1210090970",
  "res_response_code": "0000",
  "res_session_id": "7a573175e3762c01145be0ae155fcacedce030e3",
  "res_payment_channel": "15"
}
```

2.1 Advanced Features

2.1.1 2-Click Payment

2-click payment enables the customer to make a purchase without having to input card details or personal information, apart from the CVV number. This process is typically used by merchants that have repeat customers who will benefit from a faster checkout by reducing the number of fields the customer needs to fill in. If the card issuer requires 3D secure verification process, the customer will still have to complete this to make a purchase. In order for this process to work, the customer enters all of the card information only during the very first time they make a purchase. DOKU stores this data in a secure form and gives the merchant a token, which is paired to the customer's login credentials on the merchant website. After this process has been completed, each time they make a payment from hereon out, they only have to input the CVV.

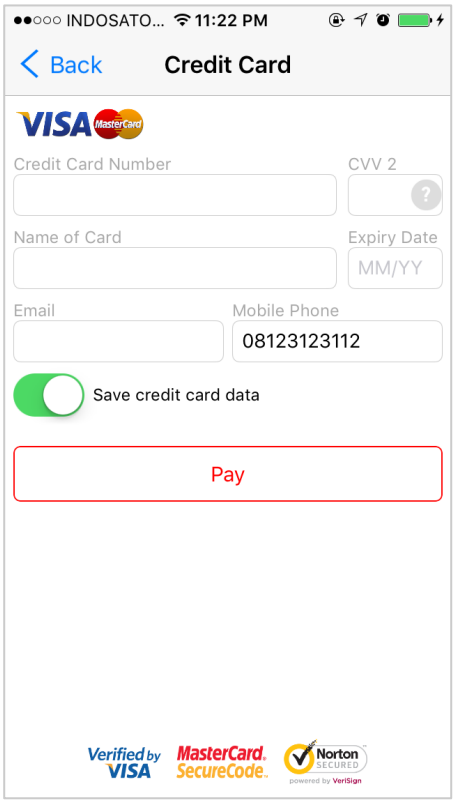
Follow these steps to apply 2-click payment to your credit card payment process:

1. Insert the additional script to your server under the payment data.
2. Generate and save the token during the first payment.
3. For subsequent payments, retrieve the token from your database and send it to the DOKU server.

1. To initialize 2-click payment, follow the same steps as general credit card processing, but add the additional parameter 'CustomerID' to your script under the payment data. The 'CustomerID' parameter may represent the customer ID that you assign to each customer within your database. This ID will be paired with the token that DOKU gives in the status response. Additionally, change the payment channel type to **DokuPaymentChannelTypeCCFirst.**

```
-(DKPaymentItem*)getPaymentItem
{
DKPaymentItem *paymentItem = [[DKPaymentItem alloc] init];
NSArray *basket =
@[@{@"name":@"sayur",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"},
@{@"name":@"buah",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"}];

paymentItem.dataAmount = @"15000.00";
paymentItem.dataBasket = basket;
paymentItem.dataImei = [self getMyUUID];
paymentItem.dataCurrency = @"360";
paymentItem.dataMerchantChain = @"NA";
paymentItem.dataSessionID = … session ID app merchant ... ;
paymentItem.dataTransactionID = … kode transaksi merchant ...;
paymentItem.isProduction = false;
paymentItem.dataMerchantCode = MerchantSharedMallID;
paymentItem.publicKey = MerchantPublicKey;
paymentItem.sharedKey = MerchantSharedKey;
paymentItem.dataWords = [paymentItem generateWords];
paymentItem.mobilePhone = @"08123123112";
paymentItem.customerID = @"12124";

[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeCCFirst];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];

return paymentItem;
}
```

The script with the additional parameter will generate the following payment form, which enables the customer to save their credit card, for faster payment. When a transaction is sent as a 2-Click payment to DOKU, in addition to the 4 fields for credit card data, DOKU will also display on the merchant website a tick box asking the customer's approval to save the card.



2. When the customer has filled in their card details and clicked "Process Payment", the data is sent to the DOKU server. Because the 'CustomerID' parameter has been added to the payment form, the DOKU server will create a token to pair with the Customer ID. If the customer checks the box next to "Save credit card data" the payment response to the Merchant server will include this token. See example response:

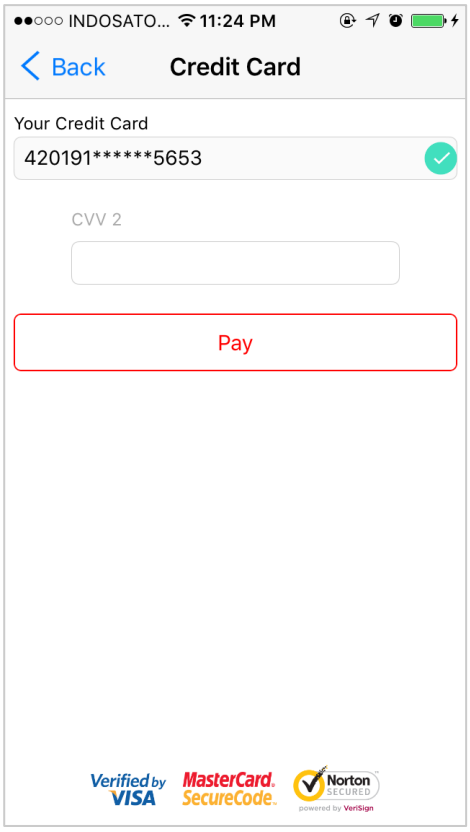

```
{
   "res_approval_code":"844647",
   "res_trans_id_merchant":"1706221359",
   "res_amount":"30000.00",
   "res_payment_date_time":"20160319114638",
   "res_verify_score":"-1",
   "res_verify_id":"",
   "res_verify_status":"NA",
   "res_words":"7553a51a091775a2462eb9150c7135f4a8d58ff161db022ca42e0ef65666ebf0",
   "res_response_msg":"SUCCESS",
   "res_mcn":"4***********1111",
   "res_mid":"000100013000195",
   "res_bank":"JPMORGAN CHASE BANK",
   "res_response_code":"0000",
   "res_session_id":"4cf212f141a1d7fe672db93db75cc069,PRODUCTION",
   "res_payment_channel":"15",
   "res_bundle_token":"{"res_token_payment":"0bea1c1c653dbc8e1e6c24155c629fe237325a06",
                        "res_token_msg":"SUCCESS",
                        "res_token_code":"0000"
                       }"
}
```


When the payment response is received, store it in your database for the next payment using the 2-Click service.

3. After a successful first payment, (assuming that the merchant has been correctly storing the Token data) only a slight modification needs to be made to the script. Add the extra parameter 'TokenPayment' as seen below, by using the token value that was obtained during the first payment. Additionally, change the payment channel type to **DokuPaymentChannelTypeCCSecond.**

```
…
paymentItem.dataTransactionID = … kode transaksi merchant ...;
paymentItem.isProduction = false;
paymentItem.dataMerchantCode = MerchantSharedMallID;
paymentItem.publicKey = MerchantPublicKey;
paymentItem.sharedKey = MerchantSharedKey;
paymentItem.dataWords = [paymentItem generateWords];
paymentItem.mobilePhone = @"08123123112";
paymentItem.customerID = @"12124";
paymentItem.tokenPayment = @"0bea1c1c653dbc8e1e6c24155c629fe237325a06";

[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeCCSecond];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];
…
```

The above script will generate the following payment form:



As you can see from the screenshot above, the customer no longer needs to fill out the credit card data apart from the CVV number. When the customer clicks the "Process Payment" button, it will follow the same process as regular card payments.

2.1.3 1-Click Payment

Using the same principles as 2-Click Payment, 1-Click payment takes it a step further and allows the customer to make a purchase with a single click on the mobile app. This means that they can skip the process of inputting their card details, personal information, CVV number *and* 3D secure. The customer will have to enter the card details and complete the 3D secure verification process only during the first time they make a purchase. By eliminating the extra steps, you are able to create a more seamless and easy checkout process, which may lead to a lower drop-off rate. However, please note that this is **subject to DOKU's *and* the bank's approval due to an increase in fraud risk**. Please contact DOKU if you are interested to implement the 1-Click Payment feature.

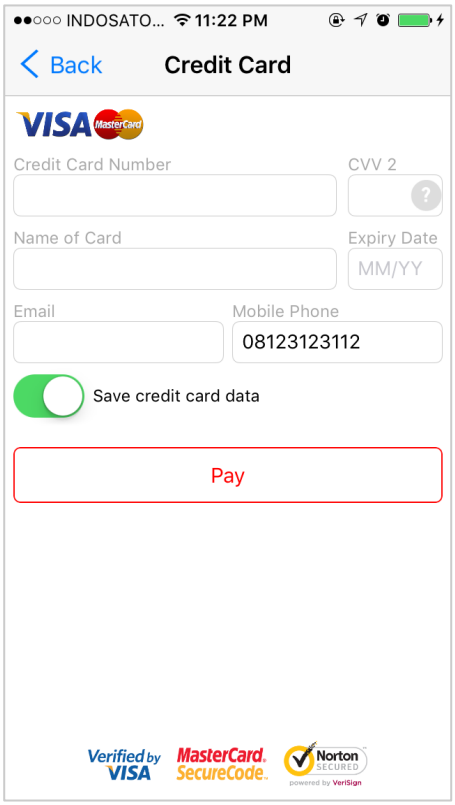
Follow these steps to apply 1-click payment to your credit card payment process:

1. Insert the additional script to your server under the payment data.
2. Generate and save the token during the first payment.
3. Change method of payment in the payment request form.

1. To initialize 1-click payment, follow the same steps as general credit card processing, but add the additional parameter 'CustomerID' to your script under the payment data. The 'CustomerID' parameter may represent the customer ID that you assign to each customer within your database. This ID will be paired with the token that DOKU gives in the status response. Additionally, change the payment channel type to **DokuPaymentChannelTypeCCFirst.**

```
-(DKPaymentItem*)getPaymentItem
{
DKPaymentItem *paymentItem = [[DKPaymentItem alloc] init];
NSArray *basket =
@[@{@"name":@"sayur",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"},
@{@"name":@"buah",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"}];

paymentItem.dataAmount = @"15000.00";
paymentItem.dataBasket = basket;
paymentItem.dataImei = [self getMyUUID];
paymentItem.dataCurrency = @"360";
paymentItem.dataMerchantChain = @"NA";
paymentItem.dataSessionID = … session ID app merchant ... ;
paymentItem.dataTransactionID = … kode transaksi merchant ...;
paymentItem.isProduction = false;
paymentItem.dataMerchantCode = MerchantSharedMallID;
paymentItem.publicKey = MerchantPublicKey;
paymentItem.sharedKey = MerchantSharedKey;
paymentItem.dataWords = [paymentItem generateWords];
paymentItem.mobilePhone = @"08123123112";
paymentItem.customerID = @"12124";

[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeCCFirst];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];

return paymentItem;
}
```

The script with the additional parameter will generate the following payment form, which enables the customer to save their credit card:



2. When the customer has filled in their card details and clicked “Process Payment”, the data is sent to the DOKU server. Because the ‘CustomerID’ parameter has been added to the payment form, the DOKU server will create a token to pair with the Customer ID. If the customer checks the box next to “Save credit card data”, the payment response to the Merchant server will include this token. See example response:


```
{
   "res_approval_code":"844647",
   "res_trans_id_merchant":"1706221359",
   "res_amount":"30000.00",
   "res_payment_date_time":"20160319114638",
   "res_verify_score":"-1",
   "res_verify_id":"",
   "res_verify_status":"NA",
   "res_words":"7553a51a091775a2462eb9150c7135f4a8d58ff161db022ca42e0ef65666ebf0",
   "res_response_msg":"SUCCESS",
   "res_mcn":"4**********1111",
   "res_mid":"000100013000195",
   "res_bank":"JPMORGAN CHASE BANK",
   "res_response_code":"0000",
   "res_session_id":"4cf212f141a1d7fe672db93db75cc069,PRODUCTION",
   "res_payment_channel":"15",
   "res_bundle_token":"{"res_token_payment":"0bea1c1c653dbc8e1e6c24155c629fe237325a06",
                                    "res_token_msg":"SUCCESS",
                                    "res_token_code":"0000"
                                   }"
}
```

When the payment response is received, store it in your database for the next payment using the 1-Click service.

3. For future payments, the request will work server-to-server without involving the SDK. When you retrieve the data from your server, create the payment request following the instructions for regular credit card payments. However, when you make the payment request, you must change the method of payment to **`Doku_Api::doDirectPayment`** instead of **`Doku_Api::doPayment`**. See example:

```
<?php
require_once('../Doku.php');

Doku_Initiate::$sharedKey = '<Put Your Shared Key Here>';
Doku_Initiate::$mallId = '<Put Your Merchant Code Here>';

$params = array(
    'amount' => '100000.00',
    'invoice' => '123456789'
);

$words = Doku_Library::doCreateWords($params);

$customer = array(
    'name' => 'TEST NAME',
    'data_phone' => '08121111111',
    'data_email' => 'test@test.com',
    'data_address' => 'bojong gede #1 08/01'
);

$basket[] = array(
    'name' => 'sayur',
    'amount' => '10000.00',
    'quantity' => '1',
    'subtotal' => '10000.00'
);

$basket[] = array(
    'name' => 'buah',
    'amount' => '10000.00',
    'quantity' => '1',
    'subtotal' => '10000.00'
);

$dataPayment = array(
    'req_mall_id' => Doku_Initiate::$mallId,
    'req_chain_merchant' => 'NA',
    'req_amount' => $params['amount'],
    'req_words' => $words,
    'req_purchase_amount' => $params['amount'],
    'req_trans_id_merchant' => $params['invoice'],
    'req_request_date_time' => date('YmdHis'),
    'req_currency' => '360',
    'req_purchase_currency' => '360',
    'req_session_id' => sha1(date('YmdHis')),
    'req_name' => $customer['name'],
    'req_payment_channel' => '15',
    'req_email' => $customer['data_email'],
    'req_basket' => $basket,
    'req_address' => $customer['data_address'],
    'req_token_payment' =>
'0bea1c1c653dbc8e1e6c24155c629fe237325a06',
    'req_customer_id' => '12124'
);

$response = Doku_Api::doDirectPayment($dataPayment);

if($response->res_response_code == '0000'){
    echo 'PAYMENT SUCCESS -- ';
}else{
    echo 'PAYMENT FAILED -- ';
}

?>
```

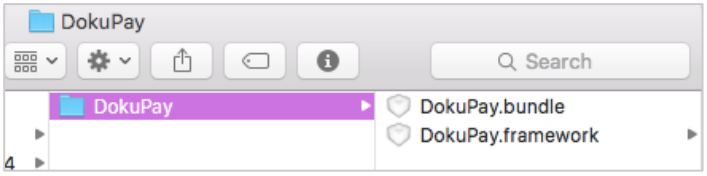
3.0 DOKU Wallet

DOKU Wallet integration comprises 3 steps:

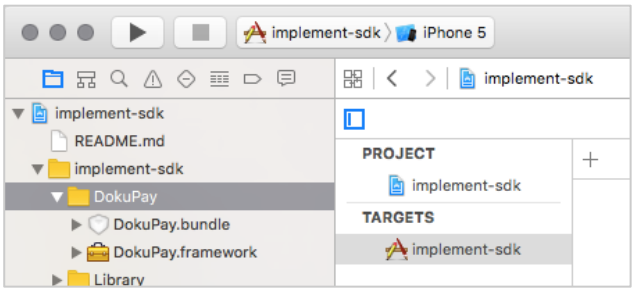
1. Initialize the DOKU iOS SDK
2. Create dependencies
3. Retrieve token
4. Send payment request

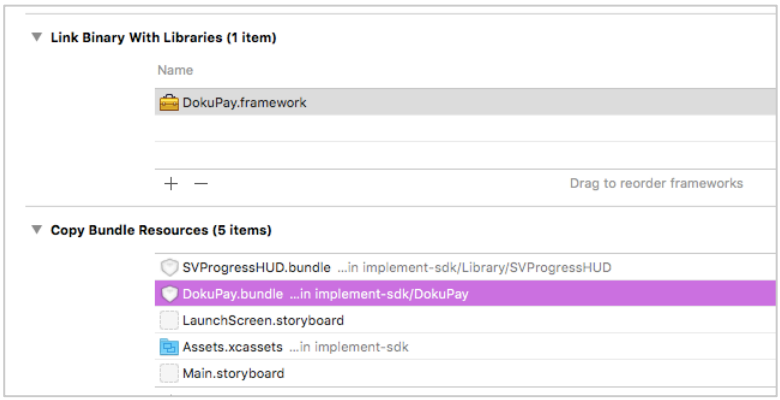
1. Initializing the DOKU iOS SDK

b. Download the DokuPay folder containing DokuPay.framework and DokuPay.bundle from http://doku.com/iOS-OCO/SDKOCO-DummyMerchant_V2-iOS.zip and copy it into your App's project folder. See example:



Select the project folder and insert the copied DokuPay folder. Ensure that the DokuPay.framework is within the 'General' -> 'Linked Frameworks and Libraries' list and the DokuPay.bundle is within the 'Build Phases' -> 'Copy Bundle Resources' list.



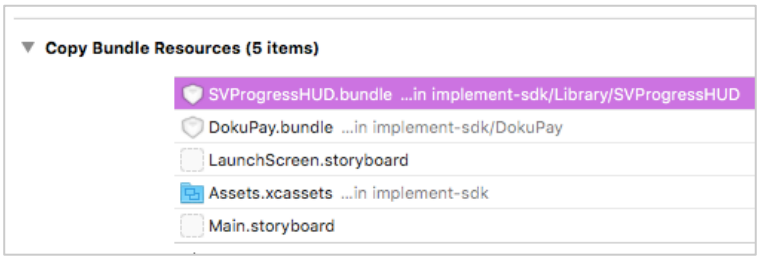


Call the DokuPay.h class from the framework using '#import' then add '' in the interface class project:

```
//

#import "ViewController.h"
#import <DokuPay/DokuPay.h>
#import "AFNetworking.h"
#import "SVProgressHUD.h"

@interface ViewController () <DokuPayDelegate>
{
    DKLayout *dokuLayout;
}
@end

@implementation ViewController
```

2. Create dependencies

Download SVProgressHUD from https://github.com/SVProgressHUD/SVProgressHUD and install it to your project folder. Ensure that the SVProgressHUD.bundle is within the 'Build Phases' -> 'Copy Bundle Resources'



3. Retrieve Token

f. Below is an example code for the parameters which will be sent by the merchant app to the SDK (see the **Payment Flow** diagram for details). Set credit card as the chosen payment channel. See the Appendix (Section 7.2) for parameter format details.

```
-(DKPaymentItem*)getPaymentItem
{
DKPaymentItem *paymentItem = [[DKPaymentItem alloc] init];
NSArray *basket =
@[@{@"name":@"sayur",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"},
@{@"name":@"buah",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"}];

paymentItem.dataAmount = @"15000.00";
paymentItem.dataBasket = basket;
paymentItem.dataImei = [self getMyUUID];
paymentItem.dataCurrency = @"360";
paymentItem.dataMerchantChain = @"NA";
paymentItem.dataSessionID = … session ID app merchant ... ;
paymentItem.dataTransactionID = … kode transaksi merchant ...;
paymentItem.isProduction = false;
paymentItem.dataMerchantCode = MerchantSharedMallID;
paymentItem.publicKey = MerchantPublicKey;
paymentItem.sharedKey = MerchantSharedKey;
paymentItem.dataWords = [paymentItem generateWords];
paymentItem.mobilePhone = @"08123123112";

[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeWallet];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];

return paymentItem;
}
```

g. Get device ID by using the script below.

```
-(NSString*)getMyUUID
{
NSString *uuid = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
return [uuid stringByReplacingOccurrencesOfString:@"-" withString:@""];
}
```

h. In order for your iOS mobile app to receive response from the SDK, you will need to create a method shown in the example below:

```
-(void)onDokuPaySuccess:(NSDictionary *)dictData
{
NSLog(@"catch Success delegate : %@", dictData);
}

-(void)onDokuPayError:(NSError *)error
{
NSLog(@"catch Error delegate : %@", error);
}
```

i. Once you have entered all the conditions above correctly, the merchant app should receive the following response token from the SDK which indicates success.

```
{
  "res_token_id":
"53beb0d8da617828d1c6295d822d84b4a12c33ea",
  "res_pairing_code": "07041613200336243915",
  "res_name": "test",
  "res_data_email": "test@mail.com",
  "res_response_msg": "SUCCESS",
  "res_device_id": "867804025368595",
  "res_token_code": "0000",
  "res_amount": "15000.00",
  "res_response_code": "0000",
  "res_payment_channel": "04",
  "res_transaction_id": "1410757974"
}
```

j. After getting your token, there will be instances where your app send it forward to your server, for example for charging payments. To do this, please see the script below as an example.

```
NSURL *URL = [NSURL URLWithString:@"http://crm.doku.com/doku-library/example-payment-
mobile/merchant-example.php "];

NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:URL];
[request setValue:@"application/x-www-form-urlencoded" forHTTPHeaderField:@"Content-Type"];
[request setHTTPMethod:@"POST"];

NSString *params = [NSString stringWithFormat:@"data=%@", [self jsonStringWithPrettyPrint:NO
fromDictionary:dictData]];

NSData *data = [params dataUsingEncoding:NSUTF8StringEncoding];
[request setHTTPBody:data];

AFHTTPSessionManager *manager = [AFHTTPSessionManager manager];
manager.responseSerializer.acceptableContentTypes = [NSSet setWithObject:@"text/html"];

NSURLSessionUploadTask *task = [manager uploadTaskWithStreamedRequest:request progress:nil
completionHandler:^(NSURLResponse * _Nonnull response, id  _Nullable responseObject, NSError *
_Nullable error) {

[SVProgressHUD dismiss];

[self popError:error withReponse:responseObject];
}];
[SVProgressHUD setDefaultMaskType:SVProgressHUDMaskTypeGradient];
[SVProgressHUD showWithStatus:@"Mohon Tunggu..."];
[task resume];
```

4. Send Payment Request

c. Your app will send the data response to your server, which you can pass on to the DOKU server to process the payment:

```
<?php
require_once('../Doku.php');

Doku_Initiate::$sharedKey = '<Put Your Shared Key Here>';
Doku_Initiate::$mallId = '<Put Your Merchant Code Here>';

$token = $_POST['doku-token'];
$pairing_code = $_POST['doku-pairing-code'];
$invoice_no = $_POST['doku-invoice-no'];

$params = array(
    'amount' => '10000.00',
    'invoice' => $invoice_no,
    'currency' => '360',
    'pairing_code' => $pairing_code,
    'token' => $token
);

$words = Doku_Library::doCreateWords($params);

$basket[] = array(
    'name' => 'sayur',
    'amount' => '10000.00',
    'quantity' => '1',
    'subtotal' => '10000.00'
);

$customer = array(
    'name' => 'TEST NAME',
    'data_phone' => '08121111111',
    'data_email' => 'test@test.com',
    'data_address' => 'bojong gede #1 08/01'
);

    $dataPayment = array(
        'req_mall_id' => Doku_Initiate::$mallId,
        'req_chain_merchant' => 'NA',
        'req_amount' => '10000.00',
        'req_words' => $words,
        'req_purchase_amount' => '10000.00',
        'req_trans_id_merchant' => $invoice_no,
        'req_request_date_time' => date('YmdHis'),
        'req_currency' => '360',
        'req_purchase_currency' => '360',
        'req_session_id' => sha1(date('YmdHis')),
        'req_name' => $customer['name'],
        'req_payment_channel' => 04,
        'req_basket' => $basket,
        'req_email' => $customer['data_email'],
        'req_token_id' => $token
    );

    $result = Doku_Api::doPayment($dataPayment);

    if($result->res_response_code == '0000'){
        echo 'SUCCESS';
        //success
    }else{
        echo 'FAILED';
        //failed
    }
```

d. If you have successfully charged payments from DOKU Wallet, you will receive a response from DOKU Server as shown below. **(see response codes definitions in Appendix)**

Example success response for payments made using a credit card:


```
{
  "res_tid": "13019501",
  "res_trx_code": "ae9e54a6f0788c41a8c4d468f83fc7c268fc0725",
  "res_currency": "IDR",
  "res_approval_code": "900059",
  "res_eci": "",
  "res_chain_mall_id": "",
  "res_card_number": "4***********1111",
  "res_amount": "15000.00",
  "res_message": "PAYMENT APPROVED",
  "res_issuer_bank": "JPMORGAN CHASE BANK",
  "res_mall_id": "2",
  "res_liability": "MERCHANT",
  "res_mid": "000100013000195",
  "res_result": "SUCCESS",
  "res_payment_date": "20160411180526",
  "res_three_d_secure_status": "FALSE",
  "res_bank": "BNI",
  "res_invoice_number": "1210090970",
  "res_response_code": "0000",
  "res_session_id": "7a573175e3762c01145be0ae155fcacedce030e3",
  "res_payment_channel": "04"
}
```

4.0 Virtual Account

DOKU Virtual Account aggregates the funds using 3 different entities – Bank Permata, Bank Sinarmas and Alfa Group. When the customer clicks 'Process Payment', DOKU will generate a one-time use, 11 digit payment code which is valid at any Prima, ALTO or Bersama ATM as well as all of Alfa Group's convenience stores. For each of the different acquiring entities, the first 5 digit codes will define where the payment should be made. See codes below:

Permata: 89650
Sinarmas: 88900
Alfa: 88888

So a payment code that is valid for payment at an Alfa store would look like this: 88888-39421877483. And a bank transfer with Permata acquiring would look like this: 89650-39421877483.

Integration for ATM transfer and convenient store is practically identical; however, keep in mind that you will have to set the first 5 digits according to the payment method, and the last 11 digits will be queried from the DOKU server.

4.1 Bank Transfer

Follow these simple steps for Bank Transfer integration:

1. Generate Payment Code
2. Display Payment Code in your app
3. Receive Payment Notification
4. Notify DOKU server that Payment Notification has been received

To get started on your integration, follow these steps one by one by pasting these scripts into your app.

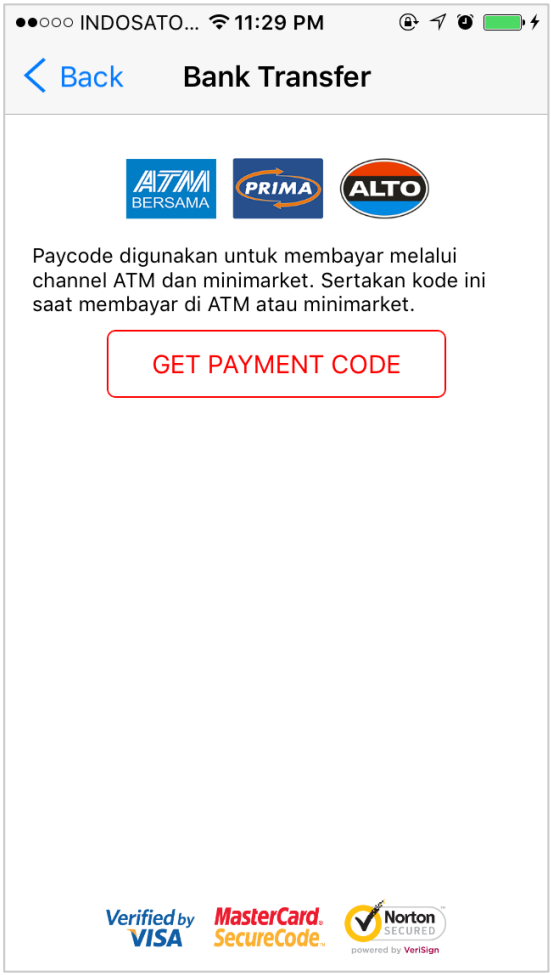
1. Send the payment parameter to the DOKU SDK using the script below while setting the payment channel to virtual account.

```
-(DKPaymentItem*)getPaymentItem
{
DKPaymentItem *paymentItem = [[DKPaymentItem alloc] init];
NSArray *basket =
@[@{@"name":@"sayur",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"},
@{@"name":@"buah",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"}];

paymentItem.dataAmount = @"15000.00";
paymentItem.dataBasket = basket;
paymentItem.dataImei = [self getMyUUID];
paymentItem.dataCurrency = @"360";
paymentItem.dataMerchantChain = @"NA";
paymentItem.dataSessionID = … session ID app merchant ... ;
paymentItem.dataTransactionID = … kode transaksi merchant ...;
paymentItem.isProduction = false;
paymentItem.dataMerchantCode = MerchantSharedMallID;
paymentItem.publicKey = MerchantPublicKey;
paymentItem.sharedKey = MerchantSharedKey;
paymentItem.dataWords = [paymentItem generateWords];
paymentItem.mobilePhone = @"08123123112";

[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeVirtualAccount];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];

return paymentItem;
}
```

Get the device ID from your customer by pasting the script into your app:

```
-(NSString*)getMyUUID
{
NSString *uuid = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
return [uuid stringByReplacingOccurrencesOfString:@"-" withString:@""];
}
```

Create a “Get Payment Code” button, and display it in your app like this:



By using the DOKU PHP Library, you can make a payment code request with ease. The request process is performed host to host. Examples of the request is seen below:

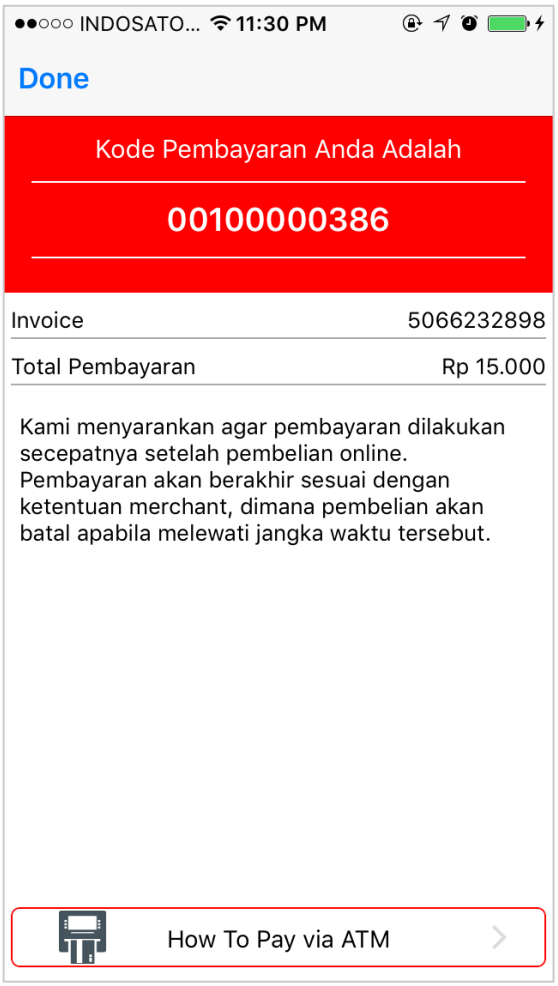
```
<?php
require_once('../Doku.php');

date_default_timezone_set('Asia/Jakarta');
Doku_Initiate::$sharedKey = <Put Your Shared Key Here>;
Doku_Initiate::$mallId = <Put Your Merchant Code Here>

$params = array(
    'amount' => $_POST['amount'],
    'invoice' => $_POST['trans_id'],
    'currency' => $_POST['currency']
);

$words = Doku_Library::doCreateWords($params);

$customer = array(
    'name' => 'TEST NAME',
    'data_phone' => '08121111111',
    'data_email' => 'test@test.com',
    'data_address' => 'bojong gede #1 08/01'
);

$dataPayment = array(
    'req_mall_id' => $_POST['mall_id'],
    'req_chain_merchant' => $_POST['chain_merchant'],
    'req_amount' => $params['amount'],
    'req_words' => $words,
    'req_trans_id_merchant' => $_POST['trans_id'],
    'req_purchase_amount' => $params['amount'],
    'req_request_date_time' => date('YmdHis'),
    'req_session_id' => sha1(date('YmdHis')),
    'req_email' => $customer['data_email'],
    'req_name' => $customer['name'],
    'req_basket' => 'sayur,10000.00,1,10000.00;',
    'req_address' => 'Plaza Asia Office Park Unit 3 Kav 59',
    'req_mobile_phone' => '081987987999'
    'req_expiry_time' => '60'
);

$response = Doku_Api::doGeneratePaycode($dataPayment);

if($response->res_response_code == '0000'){
    echo 'GENERATE SUCCESS -- ';
}else{
    echo 'GENERATE FAILED -- ';
}
?>
```

The parameter ‘reg_expiry_time’ refers to the custom expiry window for the payment to be made. Exceeding this time limit will render the payment code invalid. You may set the time limit however you like, in *minute format*. If you do not set the expiry time parameter, DOKU will set it at the default time of 360 minutes (6 hours).

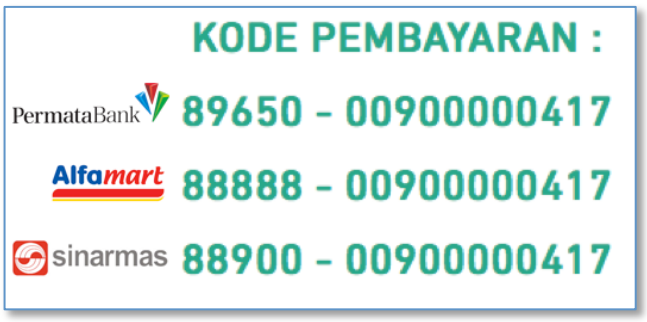
The DOKU server responds in JSON, like this:

```
{
   "res_pay_code":"62700000003",
   "res_pairing_code":"290316110837531987",
   "res_response_msg":"SUCCESS",
   "res_response_code":"0000"
}
```

2. Display the result in your app however you wish. If you choose to display the 11 digits only, letting the customer choose their payment method (remember to add a “How To” in the instructions), the result can be display like this:



Alternatively, you can display all three options like this:



3. Once the customer has made a payment, DOKU will send a payment notification containing the payment parameters to your server. The notification sent from DOKU will look something like this:

```
PAYMENTDATETIME=20160329110948
PURCHASECURRENCY=360
PAYMENTCHANNEL=05
AMOUNT=10000.00
PAYMENTCODE=00100000029
WORDS=01d9b362d3c1b80ff9196c6a565c49e5d9b03b8a
RESULTMSG=SUCCESS
TRANSIDMERCHANT=ZA912172
BANK=PERMATA
STATUSTYPE=P
APPROVALCODE=068992
RESPONSECODE=0000
SESSIONID=7b6647973dd13211a7fcf42eba79acea68aa69a1
```

4. Notify the DOKU server that you have received the payment notification, using the following example script:

```
<?php

$PAYMENTDATETIME = $_POST['PAYMENTDATETIME'];
$PURCHASECURRENCY = $_POST['PURCHASECURRENCY'];
$PAYMENTCHANNEL = $_POST['PAYMENTCHANNEL'];
$AMOUNT = $_POST['AMOUNT'];
$PAYMENTCODE = $_POST['PAYMENTCODE'];
$WORDS = $_POST['WORDS'];
$RESULTMSG = $_POST['RESULTMSG'];
$TRANSIDMERCHANT = $_POST['TRANSIDMERCHANT'];
$BANK = $_POST['BANK'];
$STATUSTYPE = $_POST['STATUSTYPE'];
$APPROVALCODE = $POST['APPROVALCODE'];
$RESPONSECODE = $_POST['RESPONSECODE'];
$SESSIONID = $_POST[' SESSIONID']

$WORDS_GENERATED = <function to generate words>

if ( $WORDS == $WORDS_GENERATED )
{
        echo "CONTINUE";
}
else
{
        echo "WORDS NOT MATCH";
}

?>
```

4.2 Convenience Store

Follow these simple steps for Convenience Store integration:

1. Generate Payment Code
2. Display Payment Code in your app
3. Receive Payment Notification
4. Notify DOKU server that Payment Notification has been received

To get started on your integration, follow these steps one by one by pasting these scripts into your app.

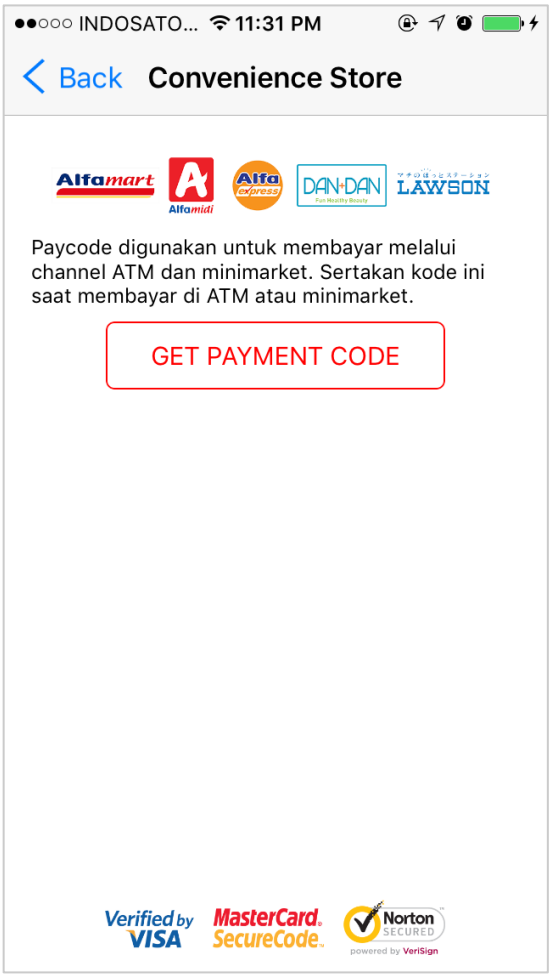
1. Send the payment parameter to the DOKU SDK using the script below while setting the payment channel to virtual account.

```
-(DKPaymentItem*)getPaymentItem
{
DKPaymentItem *paymentItem = [[DKPaymentItem alloc] init];
NSArray *basket =
@[@{@"name":@"sayur",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"},
@{@"name":@"buah",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"}];

paymentItem.dataAmount = @"15000.00";
paymentItem.dataBasket = basket;
paymentItem.dataImei = [self getMyUUID];
paymentItem.dataCurrency = @"360";
paymentItem.dataMerchantChain = @"NA";
paymentItem.dataSessionID = … session ID app merchant ... ;
paymentItem.dataTransactionID = … kode transaksi merchant ...;
paymentItem.isProduction = false;
paymentItem.dataMerchantCode = MerchantSharedMallID;
paymentItem.publicKey = MerchantPublicKey;
paymentItem.sharedKey = MerchantSharedKey;
paymentItem.dataWords = [paymentItem generateWords];
paymentItem.mobilePhone = @"08123123112";

[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeVirtualMini];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];

return paymentItem;
}
```

Get the device ID from your customer by pasting the script into your app:

```
-(NSString*)getMyUUID
{
NSString *uuid = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
return [uuid stringByReplacingOccurrencesOfString:@"-" withString:@""];
}
```

Create a “Get Payment Code” button, and display it in your app like this:



By using the DOKU PHP Library, you can make a payment code request with ease. The request process is performed host to host. Examples of the request is seen below:

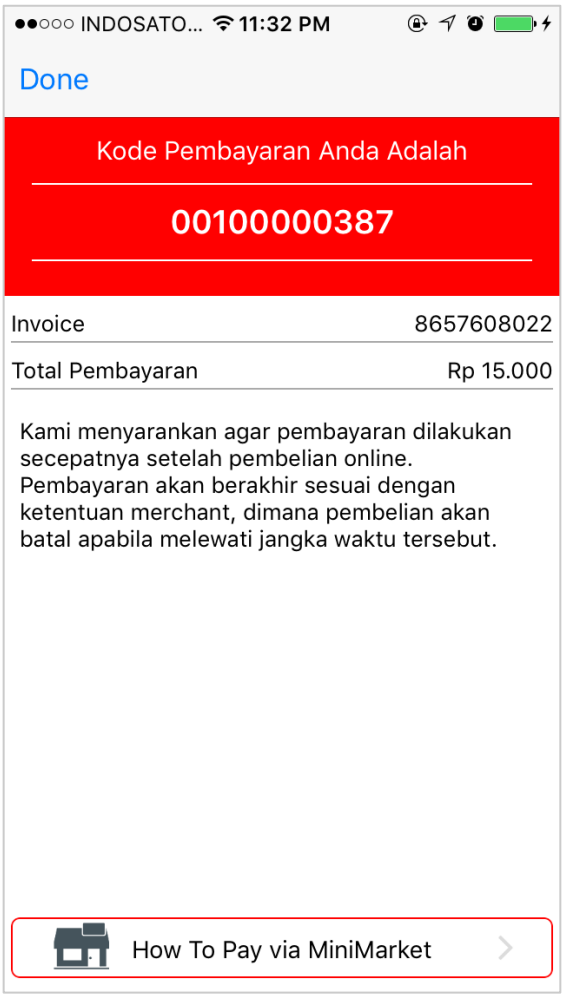
```
<?php
require_once('../Doku.php');

date_default_timezone_set('Asia/Jakarta');
Doku_Initiate::$sharedKey = <Put Your Shared Key Here>;
Doku_Initiate::$mallId = <Put Your Merchant Code Here>

$params = array(
    'amount' => $_POST['amount'],
    'invoice' => $_POST['trans_id'],
    'currency' => $_POST['currency']
);

$words = Doku_Library::doCreateWords($params);

$customer = array(
    'name' => 'TEST NAME',
    'data_phone' => '08121111111',
    'data_email' => 'test@test.com',
    'data_address' => 'bojong gede #1 08/01'
);

$dataPayment = array(
    'req_mall_id' => $_POST['mall_id'],
    'req_chain_merchant' => $_POST['chain_merchant'],
    'req_amount' => $params['amount'],
    'req_words' => $words,
    'req_trans_id_merchant' => $_POST['trans_id'],
    'req_purchase_amount' => $params['amount'],
    'req_request_date_time' => date('YmdHis'),
    'req_session_id' => sha1(date('YmdHis')),
    'req_email' => $customer['data_email'],
    'req_name' => $customer['name'],
    'req_basket' => 'sayur,10000.00,1,10000.00;',
    'req_address' => 'Plaza Asia Office Park Unit 3 Kav 59',
    'req_mobile_phone' => '081987987999'
    'req_expiry_time' => '60'
);

$response = Doku_Api::doGeneratePaycode($dataPayment);

if($response->res_response_code == '0000'){
    echo 'GENERATE SUCCESS -- ';
}else{
    echo 'GENERATE FAILED -- ';
}
?>
```

The parameter ‘reg_expiry_time’ refers to the custom expiry window for the payment to be made. Exceeding this time limit will render the payment code invalid. You may set the time limit however you like, in *minute format*. If you do not set the expiry time parameter, DOKU will set it at the default time of 360 minutes (6 hours).

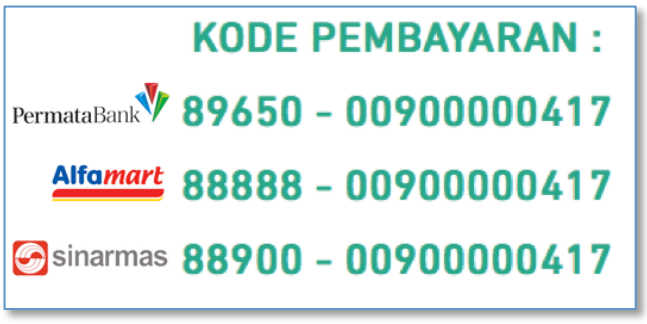
The DOKU server responds in JSON, like this:

```
{
   "res_pay_code":"62700000003",
   "res_pairing_code":"290316110837531987",
   "res_response_msg":"SUCCESS",
   "res_response_code":"0000"
}
```

2. Display the result in your app however you wish. If you choose to display the 11 digits only, letting the customer choose their payment method (remember to add a “How To” in the instructions), the result can be display like this:



Alternatively, you can display all three options like this:



3. Once the customer has made a payment, DOKU will send a payment notification containing the payment parameters to your server. The notification sent from DOKU will look something like this:

```
PAYMENTDATETIME=20160329110948
PURCHASECURRENCY=360
PAYMENTCHANNEL=05
AMOUNT=10000.00
PAYMENTCODE=00100000029
WORDS=01d9b362d3c1b80ff9196c6a565c49e5d9b03b8a
RESULTMSG=SUCCESS
TRANSIDMERCHANT=ZA912172
BANK=PERMATA
STATUSTYPE=P
APPROVALCODE=068992
RESPONSECODE=0000
SESSIONID=7b6647973dd13211a7fcf42eba79acea68aa69a1
```

4. Notify the DOKU server that you have received the payment notification, using the following example script:

```
<?php

$PAYMENTDATETIME = $_POST['PAYMENTDATETIME'];
$PURCHASECURRENCY = $_POST['PURCHASECURRENCY'];
$PAYMENTCHANNEL = $_POST['PAYMENTCHANNEL'];
$AMOUNT = $_POST['AMOUNT'];
$PAYMENTCODE = $_POST['PAYMENTCODE'];
$WORDS = $_POST['WORDS'];
$RESULTMSG = $_POST['RESULTMSG'];
$TRANSIDMERCHANT = $_POST['TRANSIDMERCHANT'];
$BANK = $_POST['BANK'];
$STATUSTYPE = $_POST['STATUSTYPE'];
$APPROVALCODE = $POST['APPROVALCODE'];
$RESPONSECODE = $_POST['RESPONSECODE'];
$SESSIONID = $_POST[' SESSIONID']

$WORDS_GENERATED = <function to generate words>

if ( $WORDS == $WORDS_GENERATED )
{
        echo "CONTINUE";
}
else
{
        echo "WORDS NOT MATCH";
}

?>
```

5.0 Internet Banking

Each bank has its own flow and authentication process for Internet Banking Payments. The majority of Internet Banking is hosted on the respective banks' own webpages where the customer enters his/her credentials and completes the authentication. So even though the initial payment steps will occur on the your page, it is redirected to the bank page eventually. Out of the Internet Banking facilities that are supported by DOKU currently, only Mandiri Clickpay allows for a merchant hosted flow.

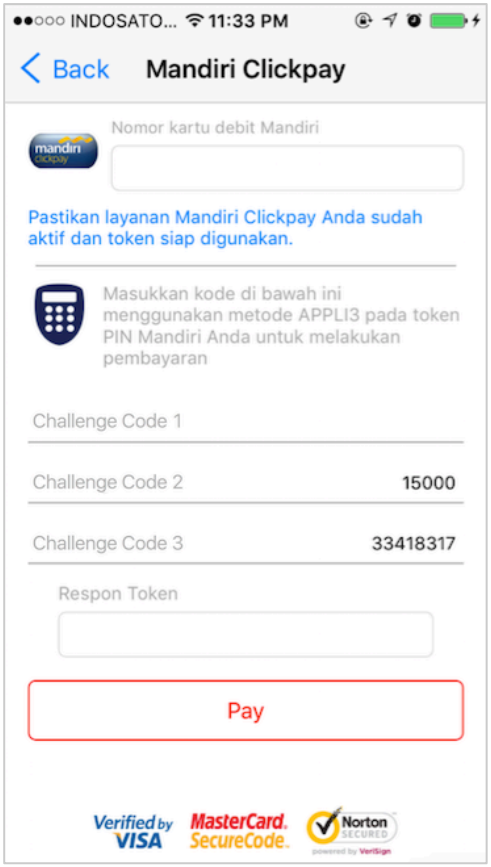
5.1 Mandiri Clickpay

Mandiri Clickpay integration comprises 3 easy steps:

1. Create payment form and send parameters to your server
2. Insert the challenge code formula
3. Send payment request

To get started on your integration, follow these steps one by one by pasting these scripts into your app.

1. Create the payment form with a field for debit card number input, like this:



Get the device ID from your customer by pasting the script into your app:

```
-(NSString*)getMyUUID
{
NSString *uuid = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
return [uuid stringByReplacingOccurrencesOfString:@"-" withString:@""];
}
```

2. As usual, set the payment channel in your request payment script into Mandiri ClickPay:

```
-(DKPaymentItem*)getPaymentItem
{
DKPaymentItem *paymentItem = [[DKPaymentItem alloc] init];
NSArray *basket =
@[@{@"name":@"sayur",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"},
@{@"name":@"buah",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"}];

paymentItem.dataAmount = @"15000.00";
paymentItem.dataBasket = basket;
paymentItem.dataImei = [self getMyUUID];
paymentItem.dataCurrency = @"360";
paymentItem.dataMerchantChain = @"NA";
paymentItem.dataSessionID = … session ID app merchant ... ;
paymentItem.dataTransactionID = … kode transaksi merchant ...;
paymentItem.isProduction = false;
paymentItem.dataMerchantCode = MerchantSharedMallID;
paymentItem.publicKey = MerchantPublicKey;
paymentItem.sharedKey = MerchantSharedKey;
paymentItem.dataWords = [paymentItem generateWords];
paymentItem.mobilePhone = @"08123123112";

[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeMandiriClickPay];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];


return paymentItem;
}
```

2. Once your customer has input their Mandiri Debit Card number, they will receive the first challenge code to enter into their Mandiri Token Device.

The challenge code formulas have different formats according to the table below:

Challenge Code 1	Last 10 digits from debit card
Challenge Code 2	Amount for charging
Challenge Code 3	8 digit random number

Obtain the challenge code and make a payment charge by copying the following script:

```
-(void)onDokuMandiriPaySuccess:(NSDictionary*)response
{
NSDictionary *dict = @{@"req_challenge_code_3": [response objectForKey:@"challenge3"],
@"req_response_token": [response objectForKey:@"responseValue"],
@"req_challenge_code_2": [response objectForKey:@"challenge2"],
@"req_challenge_code_1": [response objectForKey:@"challenge1"],
@"req_card_number": [response objectForKey:@"debitCard"],
@"req_transaction_id": … kode transaksi merchant ...,
@"req_payment_channel": @"02", @"req_device_id": … UUID ...};

NSURL *URL = [NSURL URLWithString:@"http://crm.doku.com/doku-library-staging/example-payment-
mobile/merchant-mandiri-example.php"];

NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:URL];
[request setValue:@"application/x-www-form-urlencoded" forHTTPHeaderField:@"Content-Type"];
[request setHTTPMethod:@"POST"];

NSString *params = [NSString stringWithFormat:@"data=%@", [self jsonStringWithPrettyPrint:NO
fromDictionary:dict]];

NSData *data = [params dataUsingEncoding:NSUTF8StringEncoding];
[request setHTTPBody:data];

AFHTTPSessionManager *manager = [AFHTTPSessionManager manager];
manager.responseSerializer.acceptableContentTypes = [NSSet setWithObject:@"text/html"];

NSURLSessionUploadTask *task = [manager uploadTaskWithStreamedRequest:request progress:nil
completionHandler:^(NSURLResponse * _Nonnull response, id  _Nullable responseObject, NSError *
_Nullable error){

[SVProgressHUD dismiss];

NSLog(@"mandiri charging : %@", dict);

[self popError:error withReponse:responseObject];
}];

[SVProgressHUD setDefaultMaskType:SVProgressHUDMaskTypeGradient];
[SVProgressHUD showWithStatus:@"Mohon Tunggu..."];
[task resume];
}
```

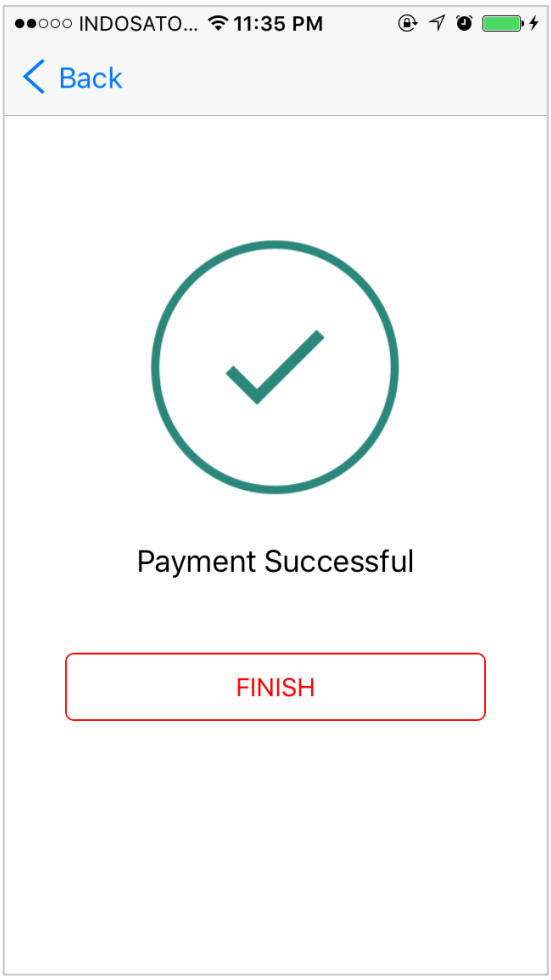
Once the customer has completed the Challenge Codes and clicked the “Process Payment” button, your app will send the data to your server. Then your server will pass through the data to the DOKU server, using our library. See example:

```
<?php
require_once('../Doku.php');

Doku_Initiate::$sharedKey = '<Put Your Shared Key Here>';
Doku_Initiate::$mallId = '<Put Your Merchant Code Here>';

$params = array(
    'amount' => '100000.00',
    'invoice' => $_POST['invoice_no']
);

$cc = str_replace(" - ", "", $_POST['cc_number']);
$words = Doku_Library::doCreateWords($params);

$customer = array(
    'name' => 'TEST NAME',
    'data_phone' => '08121111111',
    'data_email' => 'test@test.com',
    'data_address' => 'bojong gede #1 08/01'
);

$basket[] = array(
    'name' => 'sayur',
    'amount' => '10000.00',
    'quantity' => '1',
    'subtotal' => '10000.00'
);

$basket[] = array(
    'name' => 'buah',
    'amount' => '10000.00',
    'quantity' => '1',
    'subtotal' => '10000.00'
);

$dataPayment = array(
    'req_mall_id' => '1',
    'req_chain_merchant' => 'NA',
    'req_amount' => $params['amount'],
    'req_words' => $words,
    'req_purchase_amount' => $params['amount'],
    'req_trans_id_merchant' => $_POST['invoice_no'],
    'req_request_date_time' => date('YmdHis'),
    'req_currency' => '360',
    'req_purchase_currency' => '360',
    'req_session_id' => sha1(date('YmdHis')),
    'req_name' => $customer['name'],
    'req_payment_channel' => '02',
    'req_email' => $customer['data_email'],
    'req_card_number' => $cc,
    'req_basket' => $basket,
    'req_challenge_code_1' => $_POST['CHALLENGE_CODE_1'],
    'req_challenge_code_2' => $_POST['CHALLENGE_CODE_2'],
    'req_challenge_code_3' => $_POST['CHALLENGE_CODE_3'],
    'req_response_token' => $_POST['response_token'],
    'req_mobile_phone' => $customer['data_phone'],
    'req_address' => $customer['data_address']
);

$response = Doku_Api::doDirectPayment($dataPayment);


if($response->res_response_code == '0000'){
    echo 'PAYMENT SUCCESS -- ';
}else{
    echo 'PAYMENT FAILED -- ';
}
```

A success response from Mandiri Clickpay will look like this:

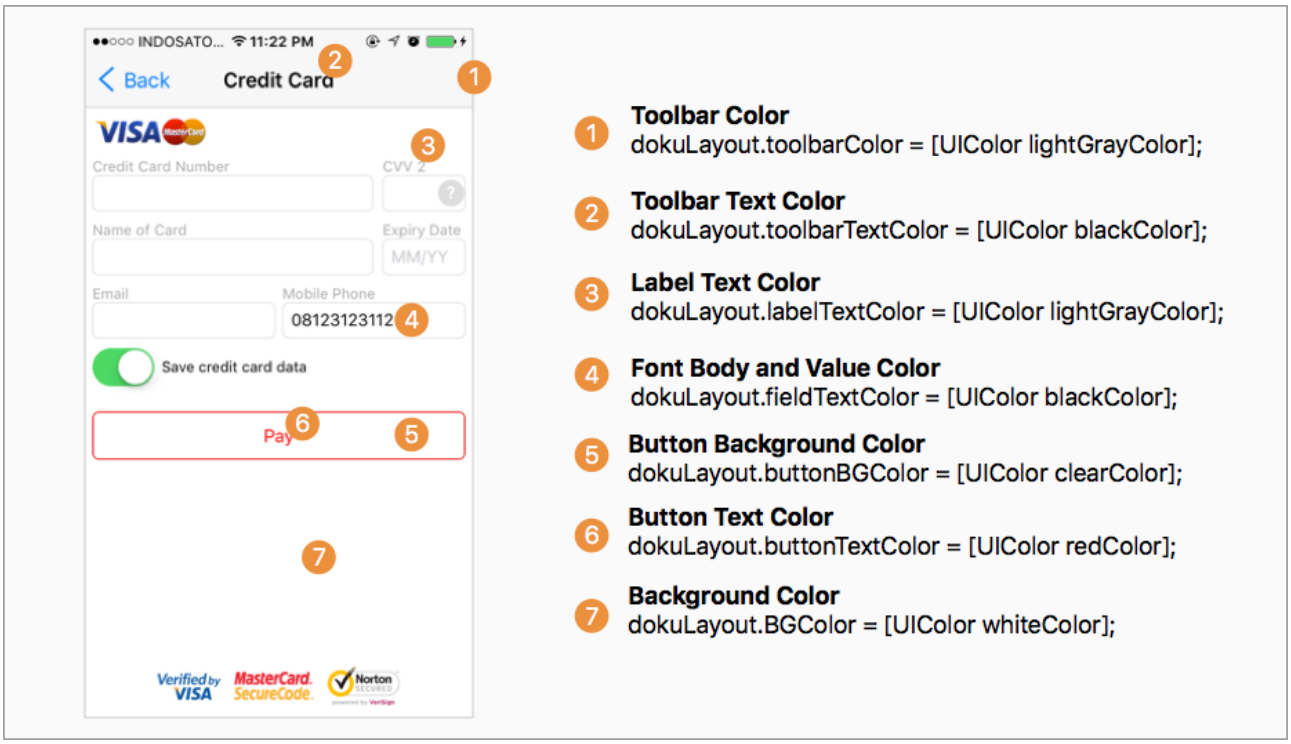

```
{
"res_amount": "10000.00",
"res_approval_code": "3862000000",
"res_bank": "Mandiri Click Pay",
"res_mcn": "411111******1111",
"res_message": "PAYMENT APPROVED",
"res_payment_date": "20160614111626",
"res_request_code": "1406160416263743250",
"res_response_code": "0000",
"res_response_msg": "SUCCES",
"res_session_id": "6a5b6a576e23037b3aba634156fd8124a04bf5c2",
"res_trans_id_merchant": "invoice_1465852586",
"res_transaction_code":
"4756b9f939a5c1557bf55d0e3bc8f47b34ca81e4"
}
```


After this is done, you may display the result on your app for the customer to see.



6.0 Customization

In order to customize your payment page, you need to set the layout settings every time you call the library.
If you do not customize your page, the layout will have the default theme set by DOKU, and will look like this:



Here is a sample script for how to set up your custom layout by initializing the DKLayout class:

```
dokuLayout = [[DKLayout alloc] init];
dokuLayout.toolbarColor = [UIColor redColor];
dokuLayout.toolbarTextColor = [UIColor greenColor];
dokuLayout.toolbarTintColor = [UIColor whiteColor];
dokuLayout.fieldTextColor = [UIColor blueColor];
dokuLayout.labelTextColor = [UIColor orangeColor];
dokuLayout.BGColor = [UIColor lightGrayColor];
dokuLayout.buttonTextColor = [UIColor yellowColor];
dokuLayout.buttonBGColor = [UIColor darkGrayColor];
```

After initialization, set it to the SDK class as shown below:

```
DKPaymentItem *paymentItem = [self getPaymentItem];
[[DokuPay sharedInstance] setLayout:dokuLayout];
[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeCC];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];
```

Then your code will look like this :

```
-(DKPaymentItem*)getPaymentItem
{
DKPaymentItem *paymentItem = [[DKPaymentItem alloc] init];
NSArray *basket =
@[@{@"name":@"sayur",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"},
@{@"name":@"buah",@"amount":@"10000.00",@"quantity":@"1",@"subtotal":@"10000.00"}];

paymentItem.dataAmount = @"15000.00";
paymentItem.dataBasket = basket;
paymentItem.dataImei = [self getMyUUID];
paymentItem.dataCurrency = @"360";
paymentItem.dataMerchantChain = @"NA";
paymentItem.dataSessionID = … session ID app merchant ... ;
paymentItem.dataTransactionID = … kode transaksi merchant ...;
paymentItem.isProduction = false;
paymentItem.dataMerchantCode = MerchantSharedMallID;
paymentItem.publicKey = MerchantPublicKey;
paymentItem.sharedKey = MerchantSharedKey;
paymentItem.dataWords = [paymentItem generateWords];
paymentItem.mobilePhone = @"08123123112";

dokuLayout = [[DKLayout alloc] init];
dokuLayout.toolbarColor = [UIColor redColor];
dokuLayout.toolbarTextColor = [UIColor greenColor];
dokuLayout.toolbarTintColor = [UIColor whiteColor];
dokuLayout.fieldTextColor = [UIColor blueColor];
dokuLayout.labelTextColor = [UIColor orangeColor];
dokuLayout.BGColor = [UIColor lightGrayColor];
dokuLayout.buttonTextColor = [UIColor yellowColor];
dokuLayout.buttonBGColor = [UIColor darkGrayColor];

[[DokuPay sharedInstance] setPaymentItem:paymentItem];
[[DokuPay sharedInstance] setLayout:dokuLayout];
[[DokuPay sharedInstance] setPaymentChannel:DokuPaymentChannelTypeCC];
[[DokuPay sharedInstance] setDelegate:self];
[[DokuPay sharedInstance] presentPayment];


return paymentItem;
}
```

7.0 Appendix

7.1 Payment Methods

Payment Type	Description
Credit Card	• Visa and Mastercard for Overseas Partner. JCB upon request • Direct API available • Features (acquirer dependant): 3D and non 3D Secure, BIN filtering, 1-Click Payment, 2-Click Payment
Internet Banking	• Available: Mandiri Clickpay, BCA Klikpay, BRI e-Pay, Danamon, Muamalat, Permata • Each bank has different authentication process through OTP or token • Direct API only available for Mandiri Clickpay. The rest is re-direct only
DOKU Wallet	• E-wallet product issued by DOKU • Source of fund: cash balance or linked credit card • Max. transaction value is Rp1,000,000 for non-KYC and Rp5,000,000 for KYC users • Authenticate with email, password and static PIN that is pre-set by the user • Direct API available
Convenience Store	• Accessible in almost 10,000 Alfa group stores (Alfa Express, Alfa Midi, Alfa Mart, Lawson and DAN+DAN) • Generate 16 digit payment code at checkout, user goes to nearest store and makes payment over the counter with cash or non-cash • Max. transaction value of Rp2,000,000 • Merchant can set payment code expiry time for every transaction • Direct API available
Bank Transfer	• Virtual account housed in Bank Permata but payable from any bank that is connected to ATM Bersama, Prima or Alto networks (over 120 banks in Indonesia) • Generate 16 digit payment code at checkout, user makes payment via ATM or Internet/mobile banking that is connected to 1 of the 3 networks • Merchant can set payment code expiry time for every transaction • Direct API available

7.2 Payment Request Parameters

No	Name	Type	Description
1.	Merchant Code	String number	Merchant Code from DOKU
2.	Words	String	Data encrypted from Merchant (amount + mall_id + shared_key + transaction_id + currency + imei device)
3.	Transaction ID	String	Invoice number
4.	Amount	String number	Amount to purchase
5.	Currency	String number	Currency that are used
6.	Chain Merchant	String	Chain merchant number
7.	Basket	String	Order items
8.	Session ID	String	Unique key from merchant
9.	Imei Device	Strinng	Device Imei id from merchant
10.	Mobile Phone	String number	Customer mobile phone number
11.	Merchant Status	String	Status merchant 'TRUE': production, 'FALSE': staging'
12.	Public Key	String	Public key from DOKU
Extra Parameters for Credit Card Tokenization			
13.	Customer ID	String number	Customer ID set from merchant
14.	Token Payment	String	Token from DOKU for save CC

7.3 DOKU Response Codes

In this section of the Appendix, you will find the list of response codes and their description for the different payment methods.

7.3.1 General response codes

The response codes listed in this section include both prepayment and payment response codes, and mostly apply to all payment methods. These are the most common response codes you will receive from DOKU.

Error Code	Description
0000	Successful approval
5555	Undefined error
5501	Payment channel not registered
5502	Merchant is disabled
5503	Maximum attempt 3 times
5504	Words not match
5505	Invalid parameter
5506	Notify failed
5507	Invalid parameter detected / Customer click cancel process
5508	Re-enter transaction
5509	Payment code already expired
5510	Cancel by Customer
5511	Not an error, payment code has not been paid by Customer
5512	Insufficient Parameter
5514	Reject by Fraud System
5515	Duplicate PNR
5516	Transaction Not Found
5517	Error in Authorization process
5518	Error parsing XML
5519	Customer stop at 3D Secure page
5520	Transaction Failed via scheduler
5521	Invalid Merchant
5522	Rates were not found
5523	Failed to get Transaction status
5524	Failed to void transaction
5525	Transaction can not be process
5526	Transaction is voided because timeout to wallet
5527	Transaction will be process as Off Us Instalment
5529	Invalid Merchant
5530	Internal server error
5531	Pairing Code does not exist
5532	Invalid Payment Channel
5533	Failed to inquiry list of fund
5534	Invalid Pairing Code
5535	Invalid Token
5536	Time Out

5537	Invalid Currency
5538	Invalid Purchase Currency
5539	3D Secure Enrolment check failed
5540	3D Secure Authentication failed
5541	Form Type is not valid
5542	Duplicate Transaction ID
5543	Please check 3D Secure result
5544	Failed to delete token
5545	Failed to Void
5547	BIN are not allowed in promo
5548	Invalid Parameter
5553	Failed to tokenize

7.3.2 Credit Card

The response codes in this section only apply to credit card transactions.

Error Code	VISA	MASTERCARD	ORIGIN	ACTIONS
0001	Refer to card issuer	Refer to card issuer	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0002	Refer to card issuer, special condition	-	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0003	Invalid merchant or service provider	Invalid Merchant	VISA/MASTER	Contact DOKU or acquiring bank
0004	Pickup card	Capture card	VISA/MASTER	Should consider blocking the card temporarily or Block login ID
0005	Do Not Honor	Do Not Honor	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0006	Error	-	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0007	Pickup card, special condition (other than lost/stolen card)	-	VISA/MASTER	Should consider blocking the card
0008	-	Honor with ID	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0010	Partial Approval - Private label	-	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0011	VIP Approval	-	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0012	Invalid Transaction	Invalid Transaction	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0013	Invalid amount (currency conversion field overflow. Visa Cash - Invalid load mount)	Invalid Amount	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0014	Invalid account number (no such number)	Invalid Card Number	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0015	No such issuer	Invalid issuer	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0019	Re-enter transaction	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0021	No Action taken (unable to back out prior transaction)	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0025	Unable to locate record in file, or account number is missing from inquiry	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0028	**File** is temporarily unavailable	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0030	-	Format error	VISA/MASTER	Contact DOKU or ACQUIRING BANK

0041	Pickup card {lost card)	Lost Card	VISA/MASTER	Should consider blocking the card temporarily or Block login ID
0043	Pickup card [stolen card)	Stolen Card	VISA/MASTER	Should consider blocking the card temporarily or Block login ID
0051	Insufficient funds	Insufficient Funds/Over Credit limit	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0052	No checking account	-	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0053	non savings account	-	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0054	Expired card	Expired Card	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0055	Incorrect PIN (Visa cash - invalid or missing SI signature)	Invalid PIN	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0057	Transaction not permitted to cardholder [Visa cash - incorrect routing, not a load request)	Transaction not permitted to issuer/cardholder	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0058	Transaction not allowed at terminal	Transaction not permitted to acquirer/terminal	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0061	Activity amount limit exceeded	Exceeds withdrawal amount limit	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0062	**Restricted** card (for example in country exclusion table)	Restricted Card	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0063	Security violation	Security Violation	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0065	Activity count limit exceeded	Exceeds withdrawal count limit	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0075	Allowable number of PIN-entry tries exceeded	Allowable number of PIN tries exceeded	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0076	Unable to locate previous message (no match on Retrieval Reference number)	Invalid/nonexistent "To Account" specified	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0077	Previous message located for a repeat or reversal, but repeat or reversal data are inconsistent with original message	Invalid/nonexistent "From account" specified	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0078	-	Invalid/nonexistent account specified (general)	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0080	invalid date (For use in private label card transactions and check acceptance transactions)	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK

0081	PIN Cryptographic error found (error found by VIC security module during PIN decryption)	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0082	Incorrect CW/1CW	-	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0083	Unable to verify PIN	-	VISA/MASTER	Tell Customer to contact the Bank Issuer of the card used.
0084	-	Invalid Authorization Life Cycle	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0085	No reason to decline a request for account number verification or address verification	Not Decline Valid for AVS only, balance inquiry, or SET Cardholder certificate requests [VISA Only)	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0091	Issuer unavailable or switch inoperative (STIP not applicable or available for this transaction)	Authorization System or issuer system inoperative	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0092	Destination cannot be found for routing	Unable to route transaction	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0093	Transaction cannot be completed; violation of law	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0094	-	Duplicate transmission detected	VISA/MASTER	Contact DOKU or ACQUIRING BANK
0096	System malfunction / System malfunction or certain field error conditions	System Error	VISA/MASTER	Contact DOKU or ACQUIRING BANK
00NO	Force STIP	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
00N3	Cash service not available	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
00N4	Cash request exceeds issuer limit	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
00N7	Decline for CW2 failure	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
00P2	Invalid biller information	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
00P5	PIN Change/Unblock request declined	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
00P6	Unsafe PIN	-	VISA/MASTER	Contact DOKU or ACQUIRING BANK
00TO	Timeout / Transaction's response exceed time limit	Timeout / Transaction's response exceed time limit	DOKU	Contact DOKU or ACQUIRING BANK
00UE	Unknown Exception / PosServer not responding	Unknown Exception / PosServer not responding	DOKU	Contact DOKU or ACQUIRING BANK

7.3.3 DOKU Wallet

The response codes in this section only apply to DOKU Wallet transactions.

Error Code	Description
0E01	FAILED GET MERCHANT
0E02	MASTER MERCHANT INACTIVE
0E03	INVALID WORDS FROM MERCHANT
0E04	INVALID MERCHANT
0E05	FAILED TO PROCESS PAYMENT
0E06	PAYMENT METHOD NOT DEFINE
0E07	FAILED EXECUTE PRE AUTH PLUGINS
0E08	FAILED EXECUTE POST AUTH PLUGINS
0E09	INVALID PAY ID
0E10	ERROR PAY ID
0E11	FAILED EXECUTE PRE TRANS MIP PLUGINS
0E12	VERIFY RESPONSE STOP FROM MERCHANT
0E13	FAILED VERIFY TO MERCHANT
0E14	FAILED SEND PAYMENT CASH WALLET
0E15	NOTIFY RESPONSE STOP FROM MERCHANT
0E16	FAILED NOTIFY TO MERCHANT
0E18	FAILED EXECUTE POST TRANS MIP PLUGINS
0E19	NOT ENOUGH CASH BALANCE AND DON'T HAVE CREDIT CARD
0E20	SPENDER NO HAVE LINK TO CREDIT CARD
0E21	ERROR CHECK 3D SECURE CREDIT CARD
0E22	PIN/OTP IS NOT VALID
0E23	PLEASE INPUT CVV2
0E24	INVALID SESSION
0E25	FAILED SEND LINK AUTHENTICATION TO CARD HOLDER
0E26	INSUFFICIENT PARAMS
0E27	FAILED EXECUTE PRE TRANS CIP PLUGINS
0E28	FAILED EXECUTE POST TRANS CIP PLUGINS
0E29	FAILED SEND PAYMENT MIP CREDIT CARD
0E30	YOU DO NOT HAVE PIN
0E31	DUPLICATE INVOICE NO
0E32	URL NOT FOUND
0E33	CUSTOMER NOT FOUND
0E34	VOID PROCESS FAILED
0E35	Failed Send ONE TIME PIN to your email
0E36	Failed Send Link for create PIN to your email
0E37	THIS SPENDER CAN'T TRANSACT IN THIS MERCHANT
0E38	You have reach your DOKU ID Transaction Limit
0E39	Process MIP Transaction Failed
0E99	ERROR SYSTEM

7.3.4 Virtual Account

The response codes in this section only apply to Convenience Store and Bank Transfer transactions.

Error Code	Description
0001	Decline (internal error)
0013	Invalid amount
0014	Bill not found
0066	Decline
0088	Bill already paid

7.3.5 Mandiri Clickpay

The response codes in this section only apply to Mandiri Clickpay transactions.

Error Code	Description
0001	Internal system error: cannot parse message
0002	Internal system error: unmatched signature hash
0003	Internal system error: Cannot process message
0004	Internal system error: Error on field
0005	Internal system error: Transaction not found
0006	Internal system error: Create VPA response error
0101	Internal system error: Create velis-authenticator message
0102	Internal system error: Runtime try/catch error when creating VTCPStream
0103	Internal system error: Cannot connect to velis-authenticator
0104	Internal system error: Send request to velis-authenticator failed
0105	Internal system error: Waiting response from velis-authenticator failed
0106	Internal system error: Read response from velis-authenticator failed
0107	Internal system error: Parse response from velis-authenticator failed
0108	Internal system error: Signature key from velis-authenticator is invalid
1101	User not registered: Channel not register in database (not found)
1102	User not registered: User not active
1103	User not registered: User has deleted
1104	User not registered: User not found
1105	User not registered: Channel for User not active
1106	User not registered: Channel for User has deleted - no access
1107	User not registered: Channel for User not register / not found
1108	User has blocked: User has disabled
1109	User has blocked
1110	User has blocked: Channel for User has disabled
1111	User has blocked: Channel for User has blocked
1112	User already activated: User has invalid status (or already active)
1113	User already activated: Channel for User has invalid status (or already active)
1114	Invalid token: Token of User not active
1115	Invalid token: Token of User has disable
1116	Invalid token: Token of User has deleted
1117	Invalid token: Token of User not found
1118	Invalid token: Method CR not allowed for Token of User
1119	Invalid token: Method RO not allowed for Token of User
1120	Invalid token: Method SG not allowed for Token of User
1121	Invalid token: Device Token Type not valid (only support VS = VASCO Token)
1122	Invalid token response: Code Not Verified
1123	Invalid token response: Code Replay Attempt
1124	Invalid token response: Challenge Too Small
1125	Invalid token response: Challenge Too Long

1126	Invalid token response: Challenge Check Digit Wrong (Host Check Challenge Mode)
1127	Invalid token response: Challenge Character Not Decimal
1128	Invalid token response: Challenge Corrupt (Host Check Challenge Mode)
1129	Invalid token response: Response Length Out of Bounds
1130	Invalid token response: Response Too Small
1131	Invalid token response: Response Too Long
1126	Invalid token response: Challenge Check Digit Wrong (Host Check Challenge Mode)
1127	Invalid token response: Challenge Character Not Decimal
1128	Invalid token response: Challenge Corrupt (Host Check Challenge Mode)
1129	Invalid token response: Response Length Out of Bounds
1130	Invalid token response: Response Too Small
1131	Invalid token response: Response Too Long
1132	Invalid token response: Response Check Digit Wrong
1133	Invalid token response: Response Character Not Decimal
1134	Invalid token response: Response Character Not Hexadecimal
1135	Invalid token response: Token Authentication Failed
1199	Receive error response from VA
0201	Internal system error: Create DSP-ISO message failed
0202	Internal system error: No active DSPSession
0203	Internal system error: Cannot send request to DSP-Silverlake
0204	Internal system error: Waiting response from DSP-Silverlake
0205	Internal system error: Read response from DSP-Silverlake without bit 39
0206	Internal system error: Read response from DSP-Silverlake without bit126
0207	Invalid card number: Card number not belong to this CIF
2101	Invalid card number: Card not found
2102	Not enough balance
2103	Invalid customer account
2104	DSP-Silverlake system error
2199	Receive error response from DSP-Silverlake
0301	Internal system error: Cannot connect to VAM
3101	Invalid XML request: Invalid data XML (tc)
3102	Invalid XML request: Invalid data XML (userid)
3103	Invalid XML request: Invalid data XML (trace number)
3104	Invalid XML request: Invalid data XML (reference number)
3105	Invalid XML request: Invalid data XML (datetime)
3106	Invalid XML request: Invalid data XML (merchantid)
3107	Invalid XML request: Invalid data XML (bankid)
3108	Invalid XML request: Invalid data XML (item detail)
3109	Invalid XML request: Invalid data XML (amount)
3110	Invalid XML request: Invalid data XML (challenge)
3111	Invalid XML request: Invalid data XML (authentication)
3112	Invalid XML request: Invalid data XML (signature)
3113	Invalid XML request: Invalid data XML (aggregator)

3114	Invalid XML request: Error parse XML
3115	Invalid XML request: XML data is null
3116	Invalid XML request: Unmatched signature request
3117	Invalid XML request: Cannot find Aggregator
3118	User already registered: Duplicate UserID
3119	Customer account not found: Cannot find customer account
3120	Not registered UserID
3121	Daily transaction limit is reached
3122	Maximum transaction limit is reached
3123	Transaction payment rejected: Invalid limit configuration
3124	Transaction payment rejected: Cannot find Merchant ID
3125	Transaction payment rejected: Inactive merchant
3126	Transaction payment rejected: Cannot find Bank Commission
3127	Transaction payment rejected: Cannot find Bank Commission Tearing
3128	Transaction payment rejected: Cannot find Aggregator Commission
3129	Transaction payment rejected: Cannot find Aggregator Commission Tearing
3130	Transaction payment rejected: Duplicate Transaction request
3131	Reversal rejected: Cannot find original data for reversal
3132	Reversal rejected: Cannot find merchant account for reversal
3133	Registration failed: Failed add customer channel
3134	Unregistered failed: Failed remove customer channel
3135	Merchant registration failed: Duplicate Merchant
3201	Error init database
3202	Error write to database
4000	No connection to Aggregator
9000	Other error
9013	Unable to send request to bank

7.4 Check Payment Status API

DOKU provides an API for merchants to check the status of a specific transaction. The implementation of this API is optional and can be used if the merchant wants to re-confirm the status of a particular transaction. This API can be accessed by the HTTP POST Method.

HTTP action URL : **https://pay.doku.com/Suite/CheckStatus**

To use this API, you should send the below parameters to the above URL:

No	Name	Type	Length	Comments
1	MALLID	N		Given by DOKU
2	CHAINMERCHANT	N		Given by DOKU
3	TRANSIDMERCHANT	AN	…30	Transaction ID from merchant
4	SESSIONID	AN	…48	
5	WORDS	AN	…200	Hashed key combination encryption (use SHA1 method). The hashed key is generated from combining the parameter values in this order : MALLID+ <shared key>+TRANSIDMERCHANT. For transaction with currency other than 360 (IDR), use : MALLID+ <shared key>+TRANSIDMERCHANT + CURRENCY

Response Status
Once the API is executed, DOKU will respond with the payment status in XML format as per below. You can check the Response Code from the table above in this Appendix.

```
<?xml version="1.0"?>
<PAYMENT_STATUS>
        <AMOUNT></AMOUNT>
        <TRANSIDMERCHANT></TRANSIDMERCHANT>
        <WORDS></WORDS>
        <RESPONSECODE></RESPONSECODE>
        <APPROVALCODE></APPROVALCODE>
        <RESULTMSG></RESULTMSG>
        <PAYMENTCHANNEL></PAYMENTCHANNEL>
        <PAYMENTCODE></PAYMENTCODE>
        <SESSIONID></SESSIONID>
        <BANK></BANK>
        <MCN></MCN>
        <PAYMENTDATETIME></PAYMENTDATETIME>
        <VERIFYID></VERIFYID>
        <VERIFYSCORE></VERIFYSCORE>
        <VERIFYSTATUS></VERIFYSTATUS>
        <CURRENCY></CURRENCY>
        <PURCHASECURRENCY></PURCHASECURRENCY>
        <BRAND></BRAND>
        <CHNAME></CHNAME>
        <THREEDSECURESTATUS></THREEDSECURESTATUS>
        <LIABILITY></LIABILITY>
        <EDUSTATUS></EDUSTATUS>
</PAYMENT_STATUS>
```