PayU

PayU | Nigeria

Introduction: Enterprise API

Introduction

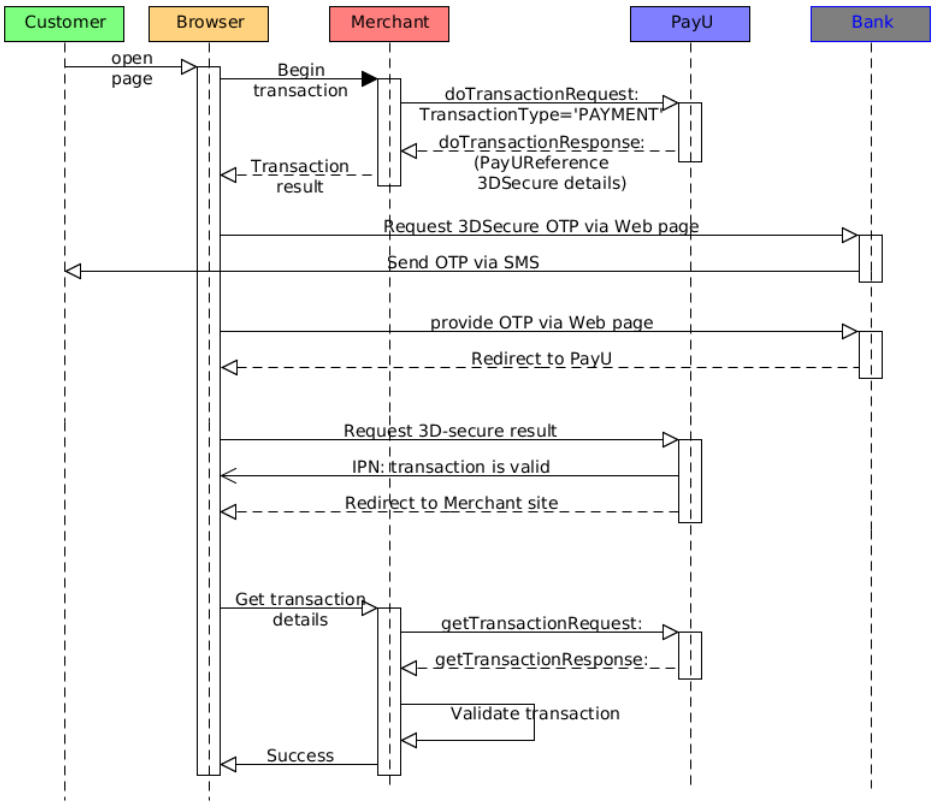
The PayU Enterprise API provides merchants a seamless integration into the PayU service to process transaction requests using their own UI. All request calls for the Enterprise API are done via service side calls to PayU with SOAP web services.

Integration overview (Enterprise API)

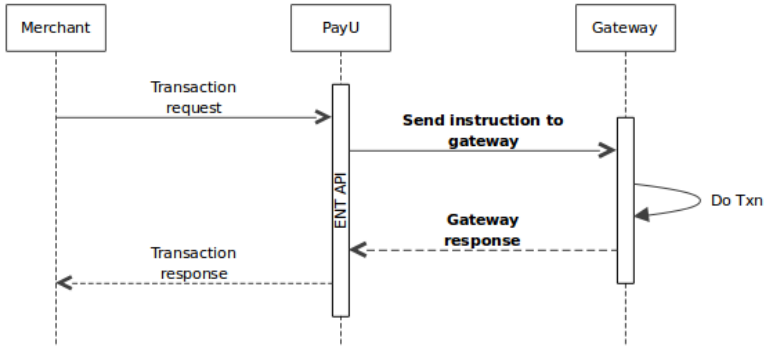
Integration into PayU's Enterprise API is accomplished server side SOAP API calls as this provides an extra layer of security.
In order to complete a successful integration, it is recommended to first do a test integration against the PayU staging environment with details given in the test overview section, before moving to the production environment. Once the test integration process is complete, the staging details will need to be substituted with the production detail. This detail will be provided by PayU once confirmation of a successful test integration has been received.

API call overview (Enterprise API)

The following is an exmaple of a credit card payment flow with 3D Secure flow. For more specific information around 3D secure, please refer to the 3D Secure section for Enterprise API.



The following is an example of a non 3D Secure integration.



The payment sequence set out above, is only for a golden path integration and more test cases as well as recommendations are given in the test overview section. This summarised payment sequence can change and is dependent on the transaction type used in the doTransaction call.

SOAP header (Enterprise API)

Each server to server SOAP call to PayU's API will require a SOAP header with a username and password defined. The SOAP header provides an extra layer of security for the merchant's transaction. Following is a detailed overview on using SOAP Headers in the PayU API calls

An example of a PayU SOAP header XML is:

```
<wsse:Security SOAP-ENV:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
   <wsse:UsernameToken wsu:Id="UsernameToken-9" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      <wsse:Username>PayU provided SOAP username</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">PayU provided SOAP password</wsse:Password>
   </wsse:UsernameToken>
</wsse:Security>
```

doTransaction (Enterprise API)

A doTransaction can be done to initiate any of the following processes:

- *RESERVE*
- *RESERVE_CANCEL*
- *PAYMENT*
- *FINALIZE*
- *CREDIT*

Please note: No special characters are allowed in any of the API call parameters as downstream systems are unable to process them.

Request Parameters

Parameter Name	Description	Data type
DoTransactionRequest		
Api	Version of the API. *Current Version: 1.0*	VarChar(4)
Safekey	PayU Merchant Identifier. *Provided to merchant upon integration*	Char(38)
TransactionType	The type of transaction being performed *RESERVE* *RESERVE_CANCEL* *PAYMENT* *FINALIZE* *CREDIT*	String
AuthenticationType	The method of authentication used if a payment is being made that requires a user's permission NA HANDSHAKE TOKEN * (2nd call and subsequent calls)	String
storePaymentMethod	Indicates if PayU must save this card and provide the merchant with a token to use for subsequent payments for the customer - True or False * (**Only required for the first call, do not resend on subsequent calls for the customer**)	
Additional Information		
MerchantReference	Unique merchant identifier for transaction	VarChar (64)
PayUReference	PayU unique reference number for transaction	VarChar(24)
DemoMode	To determine if the API request should be handled as a demo transaction *Values: True/False*	Boolean
Secure3d	To determine if the API request should process the transaction with 3DSecure *Values: True/False*	Boolean
NotificationURL	URL where the merchant would like to be notified of transaction result via IPN (Instant Payment Notification).	String
Basket		
Description	Basket description that will show on PayU	VarChar (64)
AmountInCents	Total amount of the basket in cents. *Format as 1000 for NGN10.00*	Integer
CurrencyCode	Currency code defined by PayU. *Example: NGN*	VarChar(2)

Customer		
FirstName	Customer first name	VarChar (64)
LastName	Customer last name	VarChar (64)
RegionalId	Regional identifier for customer such as RSA ID	String
Mobile	Customer Mobile Number in international format without +(eg 27827891248)	String
Email	Customer Email Address	VarChar (64)
MerchantUserId	*Unique Customer ID in merchant system (**Required for Token payment)**	VarChar (64)
DOB	Customer Date of Birth. *Format 1970-07-30*	VarChar (10)
Payment Methods – Credit Card (No handshake)		
CardNumber	The credit card number that appears on the credit card	VarChar(24)
CardExpiry	The expiry month and year of the credit card. Format required: 042002 where the first two digits represents the month and the last four represents the year in the following format MMYYYY	VarChar(6)
CVV	The credit card CVV number. Maximum length is 4 and the numbers must be positive integer	VarChar(4)
NameOnCard	The full names that appears on the credit card, e.g. John Doe	VarChar(64)
BudgetPeriod	The budget term of payment. This indicates whether buyer wishes to pay once off, or spread the period of payment over several months. The period of payment starts at 0 (once off payment), 6 months, 12 months etc. Maximum 60 months	Integer
AmountInCents	Amount in cents for payment method. *Format as 1000 for NGN10.00*	Integer
Payment Methods – Credit Card (with handshake/token authentication)		
pmID	The time-expired unique reference for the payment method of the user	VarChar (64)
CVV	The credit card CVV number. Maximum length is 4 and the numbers must be positive integer	VarChar(4)
AmountInCents	Amount in cents for payment method. *Format as 1000 for NGN10.00*	Integer

Required Fields for Request

	RESERVE	*RESERVE_CANCEL*	*PAYMENT*	*FINALIZE*	*CREDIT*
DoTransaction Request					
Api	Y	Y	Y	Y	Y
Safekey	Y	Y	Y	Y	Y
TransactionType	Y	Y	Y	Y	Y
AuthenticationType	N(3)	N	N(3)	N	N
storePaymentMethod	N	N	N(4)	N	N
Additional Information					
MerchantReference	Y	Y	Y	Y	Y
PayUReference	NA	Y	NA	Y	Y
DemoMode	N	NA	N	NA	NA
NotificationURL	N	N	N	N	N
Basket					
Description	Y	NA	Y	NA	NA
AmountInCents	Y (1)	Y	Y(1)	Y (1,2)	Y (1,2)
CurrencyCode	Y	Y	Y	Y	Y
Customer					
FirstName	N	N	N	N	N
LastName	N	N	N	N	N
RegionalId	N	N	N	N	N
Mobile	N	N	N	N	N
Email	N	N	N	N	N
MerchantUserId	N	N	N(4)	N	N
DOB	N	N	N	N	N
Payment Methods – Loyalty					
Information	NA	NA	NA	Y	NA

MermbershipNumber	NA	NA	NA	Y	NA
AmountInCents	NA	NA	NA	Y	NA
Payment Methods - Credit Card					
CardNumber	Y	NA	Y	NA	NA
CardExpiry	Y	NA	Y	NA	NA
CVV	Y	NA	Y	NA	NA
NameOnCard	Y	NA	Y	NA	NA
BudgetPeriod	N	NA	N	NA	NA
AmountInCents	Y	Y	Y	Y	Y
Custom Fields					
Key	N	N	N	N	N
Value	N	N	N	N	N
Y - Mandatory					
N - Optional					
NA - Not Allowed					
(1) Basket Amount must equal sum of PaymentMethod Amount's					
(2) Amount must equal to or less than the original transaction amount **(3) Field is required to be set to HANDSHAKE if payment is being made with PayU wallet or stored credit card** **(4) Field is required when doing a transaction with a TOKEN or to generate a TOKEN**					

getTransaction (Enterprise API)

Request Parameters

Parameter Name	Mandatory	Description	Values
GetTransactionRequest			
Api	Y	Version of the API. *Current Version: 1.0*	VarChar(4)
Safekey	Y	PayU Merchant Identifier. *Provided to merchant*	Char(38)

		upon integration	
PayUReference	C	PayU unique reference number for transaction	VarChar (64)
MerchantReference	C	MerchantReference supplied for previous transaction	VarChar (64)

Either *PayUReference* or *MerchantReference* should be supplied for the call. If both are supplied the call will fail.

Example:

```
<GetTransactionRequest Api="1.0" Safekey="{34D0CF84-CEFF-4E32-95FC-1FB9168FD130}" >
   <PayUReference> 34604892772</PayUReference>
</GetTransactionRequest>
```

Transaction State

A transaction in PayU can be in one of 4 states: NEW, PROCESSING, SUCCESSFUL and FAILED. SUCCESSFUL and FAILED are both final states that cannot be changed.

In the case of a ***DoTransaction*** for a FINALISE. The transaction will immediately go from, “NEW” to “PROCESSING” state when PayU performs the payment. Upon receiving the response from the payment gateway the state is changed to SUCCESSFUL or FAILED.

Response Parameters

Parameter Name	Description	Values
GetTransactionResponse		
Staged	Whether transaction was staged or not	Boolean
MerchantReference	Merchant identifier for transaction.	VarChar (64)
TransactionType	TransactionType on which data is being passed back on: RESERVE FINALIZE PAYMENT EFFECT_STAGING CREDIT RESERVE_CANCEL REGISTER_LINK TOPUP	String
TransactionState	The current state of the transaction being referenced. NEW PROCESSING SUCCESSFUL FAILED	String

PointOfFailure	Indicated on failed transactions where the point of failure was, blank if successful transaction	VarChar (64)
ResultCode	Result code returned by PayU for transaction	Integer
ResultMessage	Result message relating to result code for transaction	String
DisplayMessage	Customer friendly message to display in a browser to the customer	String
Basket		
Description	Basket description that will show on PayU	VarChar (64)
AmountInCents	Total amount of the basket in cents. *Format as 1000 for NGN10.00*	Integer
CurrencyCode	International Currency Code. *Example: NGN*	String
PaymentMethodsUsed		
Information	Payment method information (example: loyalty program name)	VarChar (64)
AmountInCents	Amount in cents for payment method. *Format as 1000 for NGN10.00*	Integer
NameOnCard	If credit card used, this will return cardholder name of the card	String
CardNumber	If credit card used, this will return a masked credit card number	VarChar (64)
MembershipNumber	Loyalty membership number	VarChar (64)

Instant Payment Notification (Enterprise API)

Instant Payment Notifications (IPNs) are sent back to the *NotificationURL* in (but not limited to) the following scenarios their:

1. When a payment on the PayU redirect either fails or is successful (3D secure redirect)
2. When transaction pending review for fraud is either approved or rejected by case managers. In the case of approval the IPN will fire after the payment has been attempted and will return that result.

For a detailed overview of the process see below:

The Instant Payment Notification (IPN) is an asynchronous notification mechanism where PayU sends transaction results/information to a specified URL without any dependency on a customer's browser. The merchant's system can consume the IPN and validate the outcome of the transaction. **The standard server response for a successfully posted IPN, is an 'HTTP 200 OK' status code from the specified URL's server.**

Instant Payment Notifications (IPN)'s will be fired in the following cases when the merchant has provided a ***NotificationURL*** in their ***SetTransaction*** or ***DoTransaction*** API call:

1. When a payment on the PayU redirect either fails or is successful
2. When a user session times out on the PayU redirect with no chance for the user to finish payment
3. When transaction pending review for fraud is either approved or rejected by case managers. In the case of approval the IPN will fire after the payment has been attempted and will return that result.

The *NotificationURL* can be set to an HTTP or HTTPS endpoint.
If unable to deliver an IPN on the first attempt, PayU will retry a further 2 times (3 attempts in total). If however on the 3rd attempt it could still not be successfully delivered, the IPN data will automatically be sent to the store's configured merchant email address.

Warning:

The receiving system might take longer to process the IPN than PayU can wait for the response. Once PayU reaches its timeout it will mark the message as unsuccessful and it will be sent again. This can lead to a situation where the receiver processes the **same IPN multiple times**. To stop this from happening the receiver **MUST** pay close attention to the **ResponseHash**. Each discrete IPN has a unique ResponseHash. When an IPN is resent, it will be exactly the same message. This includes the ResponseHash. Thus if the receiving system is cognisant of the fact that it is receiving the **same IPN again based on ResponseHash**, it can effectively **stop duplicate transactions**.

Example:

- Merchant sends payment instruction to PayU.
- PayU processes and sends IPN response informing merchant that transaction was successful.
- Merchant decides to credit customer and instructs PayU to credit the previous transaction.
- PayU does the credit and sends an IPN notifying the merchant that it was successful.
- Merchant receives the IPN and starts to credit the customer, *but does not respond in time*.
- PayU marks the IPN as not successful and resends.
- Merchant receives *same IPN again.*
- At this point *the merchant checks the ResponseHash* and figures out that it has already processed this IPN and ignores it, but returns a success to PayU in order to stop further resending.

IPN transaction result validation

Interpretation of IPN responses to validate the result of a transaction should ideally be done on (but not limited to) a combination of the following fields:

- resultCode
- successful
- transactionState
- transactionType

Please refer to the transaction states and types pages which sets out all of the variations found for possible values. Please refer to the error message page for all possible PayU errors that can be recieved in the IPNs.

Transaction states

A transaction in PayU can be in one of 4 states:

- NEW
- PROCESSING
- SUCCESSFUL
- FAILED

It is important to note that SUCCESSFUL and FAILED are both final states that cannot be changed.

Once a setTransaction has been completed the transaction will be in a NEW state. When the user has redirected to PayU to complete the transaction the state is changed to PROCESSING. After the payment or stage has been completed the state will change to SUCCESSFUL.

In the case of a doTransaction for a FINALIZE, the transaction will go almost immediately from the NEW state to PROCESSING when PayU go off to the external payment gateway to perform the payment. Upon receiving the response from the external payment gateway the state is changed to SUCCESSFUL or FAILED.

3D Secure (Enterprise API)

For credit cards, PayU will allow you to set up your transactions to go through an additional level of authentication with Visa or MasterCard. The process is referred to as 3D Secure and involves the customer entering an additional verification code to prove that the card is theirs. This helps prevent fraud and helps reduce the risk associated with accepting credit card payments online.

doTransaction amendments for 3D Secure

Merchants will need to make the following amendments to the doTransaction API call in order to do 3D secure transactions on the PayU platform.
AdditionalInformation.supportedPaymentMethods = CREDITCARD
AdditionalInformation.secure3d = true
AdditionalInformation.returnUrl =<valid full url e.g. http://example.com/return/>
AdditionalInformation.notificationUrl =<valid full url e.g. http://example.com/notification/ >

Integration payment sequence/flow

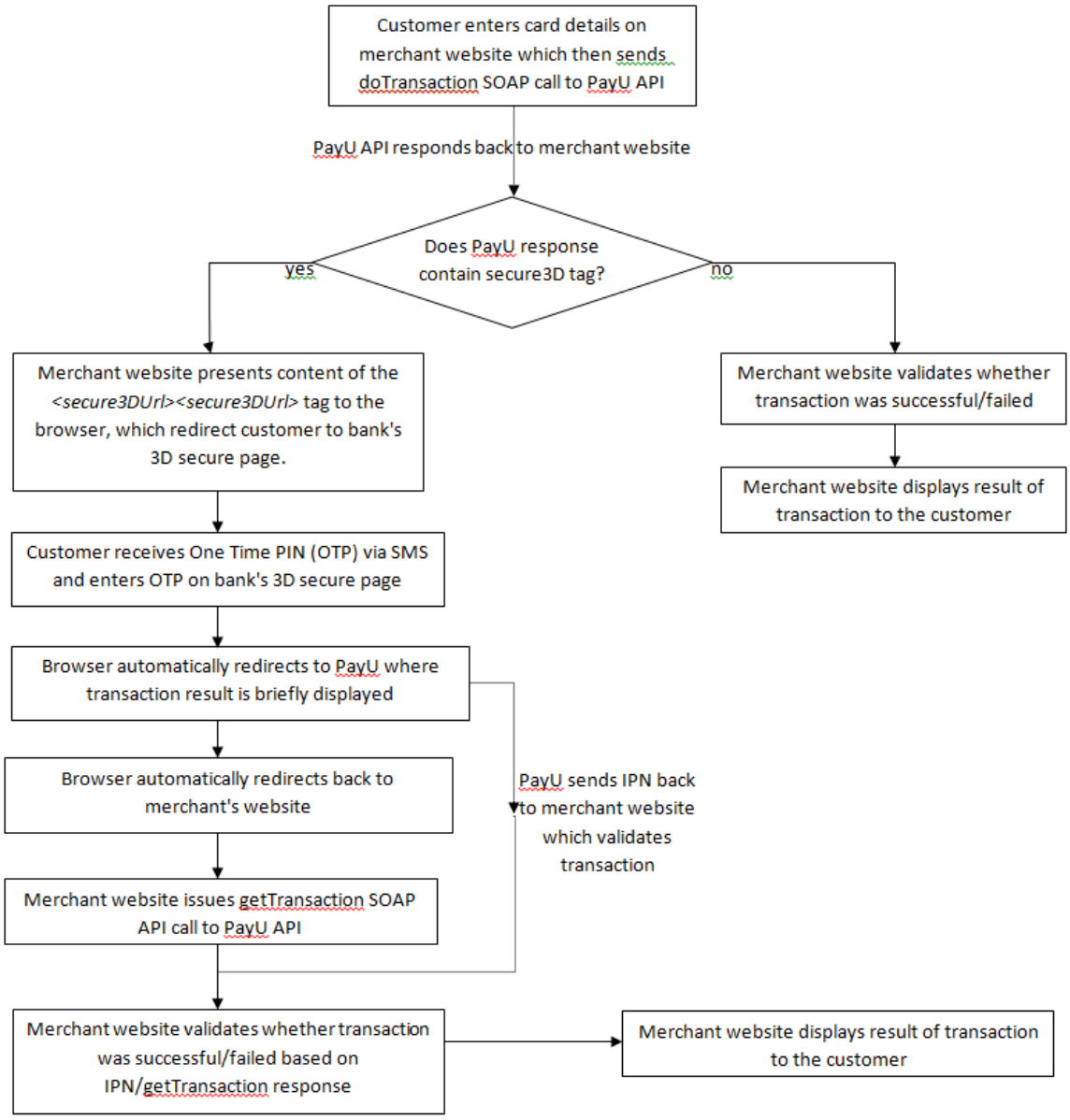
When developing a solution with PayU's Enterprise API you will need to cater for a Card enrolled and Card not enrolled scenario. When the <secure3D> tag is present in the PayU API response it indicates that card is enrolled.

Simple payment sequence integration:

Merchant's website issues a doTransaction SOAP call against PayU's API
PayU's API reponds with the result of the transaction:
2.1. If <secure3D> is not present in the response - indicating that the card is not enrolled:
    2.1.1. Validate whether transaction was failed or successful against the API response
    2.1.2. No further processing required and website indicates payment result to the customer.
2.2. If <secure3D> is present in the response - indicating that the card is enrolled:
    2.2.1. Present the <secure3DUrl></secure3DUrl> tag content to the browser which will redirect the customer to the bank's 3D Secure page
    2.2.2. Customer receives OTP via SMS
    2.2.3. Customer enters the received OTP on bank's 3D Secure page
Browser automatically redirects customer back to PayU where a message is briefly displayed.
Browser automatically redirects customer back to merchant's website (url specified within returnUrl parameter).
Merchant website issues a getTransaction SOAP call against PayU's API from merchant's website and validates payment result e.g. payment successful or declined as per Transaction result validation. The merchant will also recieve an
IPN to the url specified within notificationUrl parameter with the result of the transaction.

It should be noted that the sequence set out in this document (as well as above and flowchart below), is only for a golden path integration.

The diagram below describes the 3D secure process



Error Codes

Point of Failure	Results Code	Result Message
PAYU	P001	Invalid Safekey
	P003	General system error
	P004	IP not allowed
	P005	Invalid PayUReference
	P006	Invalid MerchantReference
	P007	Invalid TransactionType
	P011	PayU request timeout
	P014	Security error
	P015	User cancelled the transaction
	P018	Staged transaction has expired
	P019	Transaction amount must be equal to less than original transaction amount
	P022	Transaction is not in the correct state - last transaction: <TransactionType> state: <TransactionState>
	P023	Transaction in progress
	P025	Lookup Transaction timeout
	P026	Lookup Transaction pm id invalid
	P027	Customer wallet not found
	P028	Requested transaction fails security rules
	P029	User failed to preapprove payments to merchant
	P030	User cancelled
	P031	Registration failed - mobile number already exists
	P032	Registration failed - email already exists
	P033	Registration failed - NationalID/Passport already exists
	P034	Login failed - invalid credentials
	P035	Login failed - wallet locked
	P036	Validation failed - [fieldname]
	P037	Merchant store deactivated
	P038	New PIN does not match format requirements
	P039	New Password does not match format requirements
	P040	Add credit card failed - credit card already exists
	P041	Invalid MerchantID
	P042	Decryption failed
	P043	Invalid VASID
	P044	Rejected for fraud
	P045	Rejected - Fraud system is unavailable
	P046	Payment failed
	P047	Payment method is not configured or allowed
	P048	Payment pending fraud case management
	P049	Payment failed following case management

	P050	Multi-tender not supported with provided payment methods
	P051	Basket amount does not match sum of payment methods
	P052	Wallet topup failed
MIGS	0	Transaction Successful
	1	Transaction could not be processed
	2	Transaction Declined - Contact Issuing Bank
	3	Transaction Declined- No reply from Bank
	4	Transaction Declined - Expired Card
	5	Transaction Declined - Insufficient credit
	6	Transaction Declined - Bank system error
	7	Payment Server Processing Error
	8	Transaction Declined - Transaction Type Not Supported
	9	Bank Declined Transaction (Do not contact Bank)
	A	Transaction Aborted
	B	Transaction Blocked
	C	Transaction Cancelled
	D	Deferred Transaction
	E	Transaction Declined - Refer to card issuer
	F	3D Secure Authentication Failed
	I	Card Security Code Failed
	L	Shopping Transaction Locked
	N	Cardholder is not enrolled in 3D Secure
	P	Transaction is Pending
	R	Retry Limits Exceeded, Transaction Not Processed
	T	Address Verification Failed
	U	Card Security Code Failed
	V	Address Verification and Card Security Code Failed