Prepared (Subject resp)		No.		
Mohamed Maalim				
Approved (Document resp)	Checked	Date	Rev	Reference
		2022-01-14	PA9	

Ericsson Wallet Platform Open API Solution Documentation

Contents

Prepared (Subject resp)		No.		
Mohamed Maalim				
Approved (Document resp)	Checked	Date	Rev	Reference
		2022-01-14	PA9	

Revision History

Revision	Date	Author	Remarks
PA1	27th July 2018	Mats Hagman	Initial Draft
PA2	08 Aug 2018	Mohamed Maalim	Added Getting Started
PA3	11th Oct 2018	Mohamed Maalim	Added Sandbox Provisioning.
PA4	12th Oct 2018	Mohamed Maalim	Added Production API Key provisioning.
PA5	15th Oct 2018	Mohamed Maalim	Added Error Codes
PA6	6 Nov 2018	Mohamed Maalim	Reviewed Diagrams, URLs & Spell checks
PA7	4 March 2021	Avik Chatterjee	Updated for Open API phase 2 and phase one pictures changed for maintaining same styles.
PA8	23r March 2021	Avik Chatterjee	Added merchant payment with consent flow as per request from Byron.
PA9	27th Sep 2021	Arunmozhi R	Added Shoprite API's RequesttoWithdraw, Refund and Deposit

Prepared (Subject resp)		No.		
Mohamed Maalim				
Approved (Document resp)	Checked	Date	Rev	Reference
		2022-01-14	PA9	

1 Introduction

The Open API is as JSON REST API that is used by Partner systems to access services in Wallet platform. The Open API exposes services that are used by e.g. online merchants for managing payments and other financial services

This document gives an overview of the structure of the API.

2 Getting Started

Follow 9 simple steps to get Started

a) Signup for MTN MoMo API

b) Login then Click Subscribe to a product under the product Tab to obtain Subscription key.

c) Check and fetch your Subscription Keys under your profile.

d) Generate API User and API Key using the provisioning API described in **3.2.2**

e) Generate a Token using your newly created API User and API key as described in **3.2.3**

f) Navigate to the desired API and use your Subscription Key and Token to connect to API Endpoints

g) Try out the APIs on the Portal

h) Download the testcases here

i) Use the testing MSISDNs mentioned in section **5.4**

3 API description

3.1 Header

3.2 Authentication

There are two credentials used in the Open API.

- Subscription Key
- API User and API Key.

The subscription key is used to give access to APIs in the API Manager portal. A user is assigned a subscription Key as and when the user subscribes to products in the API Manager Portal.

The API User and API Key are used to grant access to the wallet system in a specific country. API user and Key is wholly managed by the merchant through Partner Portal.

Merchants is allowed to generate/revoke API Keys through the Partner Portal.

However, on Sandbox Environment a Provisioning API is exposed to enable developers generate own API User and API Key for testing purposes only.

3.2.1 Subscription key

The subscription key is part of the header of all request sent to the API Manager. The subscription key can be found under user profile in the API Manager Portal.

The subscription key is assigned to the ***Ocp-Apim-Subscription-Key*** parameter the header.

3.2.2 API User And API Key Management.

The API user and API key are provisioned differently in the sandbox and production environment.

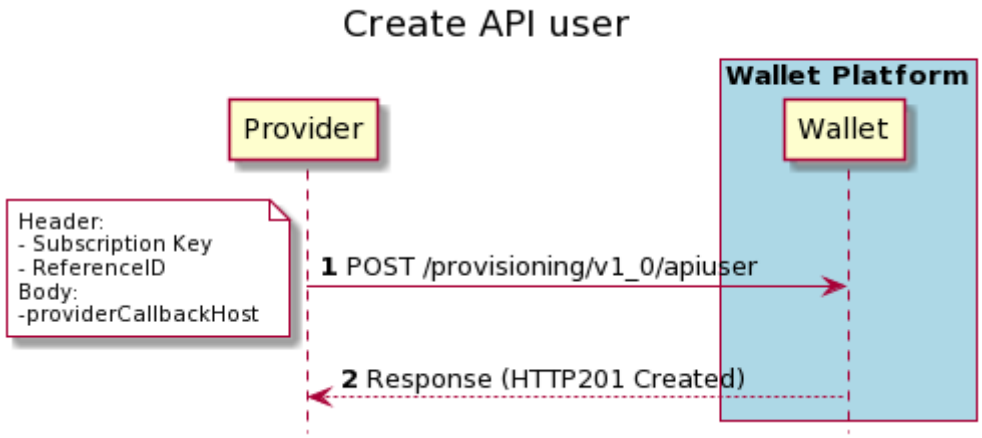
In the Sandbox a provisioning API is used to create the API User and API Key, where as in the production environment the provisioning is done through the Merchant Portal.

The sections below describe the different steps required in creating API User and API key in Sandbox and Production Environments.

3.2.2.1 Sandbox Provisioning

The Steps below describes

3.2.2.1.1 Create API User



1. The Provider sends a POST */provisioning/v1_0/apiuser* request to Wallet platform.
2. The Provider specifies the UUID Reference ID in the request Header and the subscription Key.
3. Reference ID will be used as the User ID for the API user to be created.
4. Wallet Platform creates the User and responds with 201.

Example:

Request:

POST https:{baseURL}/provisioning/v1_0/apiuser HTTP/1.1

Host: *{Sandbox URL}*

X-Reference-Id*: c72025f5-5cd1-4630-99e4-8ba4722fad56*

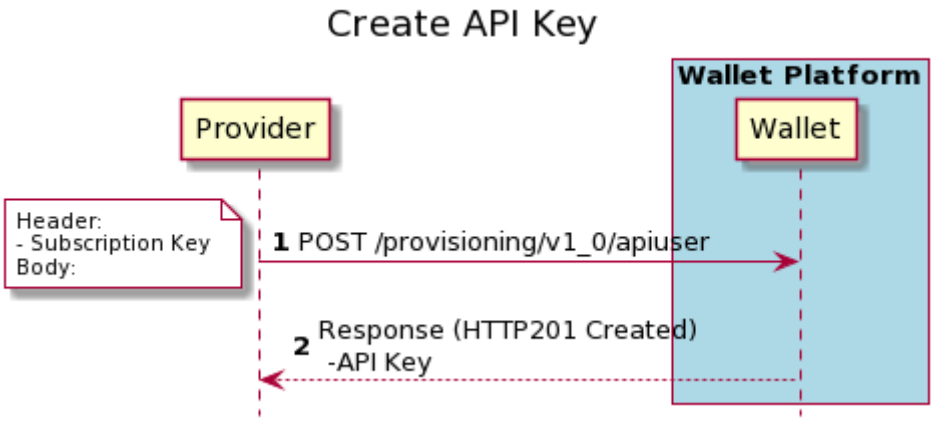
Ocp-Apim-Subscription-Key: *d484a1f0d34f4301916d0f2c9e9106a2*

{"providerCallbackHost": "Myapp.com"}

Response

201 Created

3.2.2.1.2 Create API Key



1. The Provider sends a POST

   https:{baseURL}/provisioning/v1_0/apiuser/{APIUser}/apikey request to Wallet platform.

2. The Provider specifies the API User in the URL and subscription Key in the header.

3. Wallet Platform creates the API Key and responds with 201 Created with the newly Created API Key in the Body.

4. Provider now has both API User and API Key created.

   Example:

   Request

   POST *https:{baseURL}/*provisioning/v1_0/apiuser/c72025f5-5cd1-4630-99e4-8ba4722fad56/apikey HTTP/1.1

   Host: {Sandbox}

   Ocp-Apim-Subscription-Key: *d484a1f0d34f4301916d0f2c9e9106a2*

   Response

   HTTP/1.1 201 Created

   date: Wed, 10 Oct 2018 09:16:15 GMT

   content-type: application/json;charset=utf-8

   content-length: 45

   {

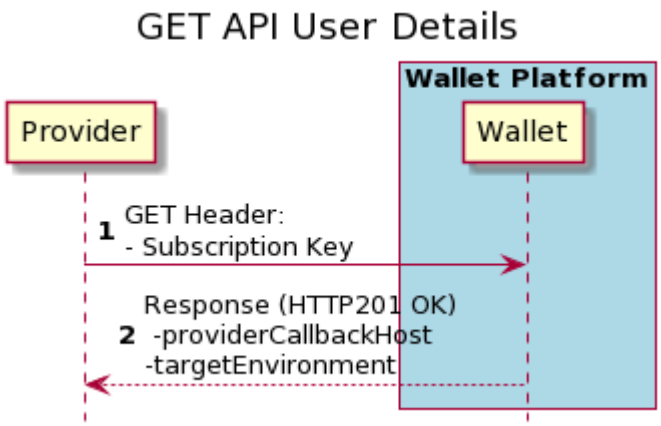
   "apiKey": "f1db798c98df4bcf83b538175893bbf0"

   }

3.2.2.1.3 GET API User details

It's possible to fetch API user details such as Call Back Host. However, its not possible to fetch the API key.

Provider shall be required to generate a new Key should they lose the existing one.



1. The Provider sends a GET *https:{baseURL}/provisioning/v1_0/apiuser/{APIUser}* request to Wallet platform.
2. The Provider specifies the API User in the URL and subscription Key in the header.
3. Wallet Platform responds with 200 Ok and details of the user.
4. TargetEnvironment is preconfigured to sandbox in the Sandbox environment, therefore Providers will not have the option of setting it to a different parameter.

Example

Request

GET https:{baseURL}/provisioning/v1_0/apiuser/ c72025f5-5cd1-4630-99e4-8ba4722fad56

***Host**: {Sandbox}*

Ocp-Apim-Subscription-Key: d484a1f0d34f4301916d0f2c9e9106a2

Response:

HTTP/1.1 200 Accepted

date: Wed, 10 Oct 2018 09:16:15 GMT

```
{
    "providerCallbackHost": "Myapp.com",
    "targetEnvironment": "sandbox"
}
```

3.2.2.2 Production Provisioning

Production API User and API Key are provisioned and managed on the Partner Portal

Partner Portal is the wallet portal granted to partners after Go-Live. The credentials for this portal are shared with partners after Go-live.

The steps below describe the provisioning process

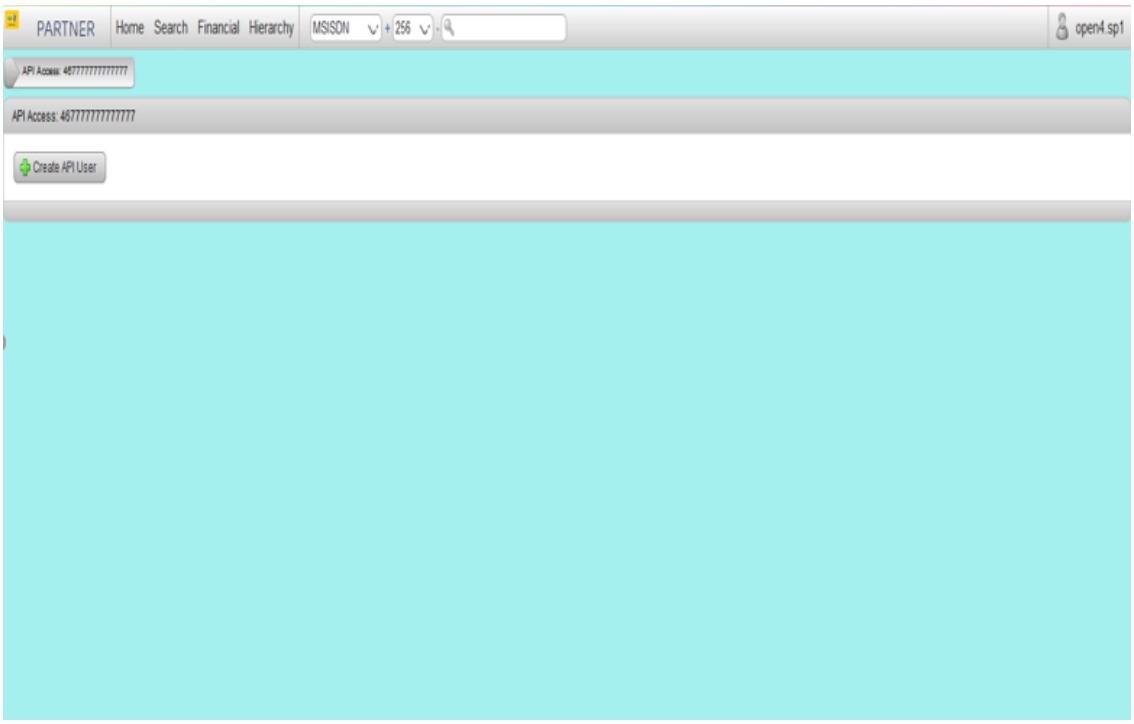
3.2.2.2.1 Log on to Partner Portal

The URL and Credentials for partner portal are to be obtained after Go-live

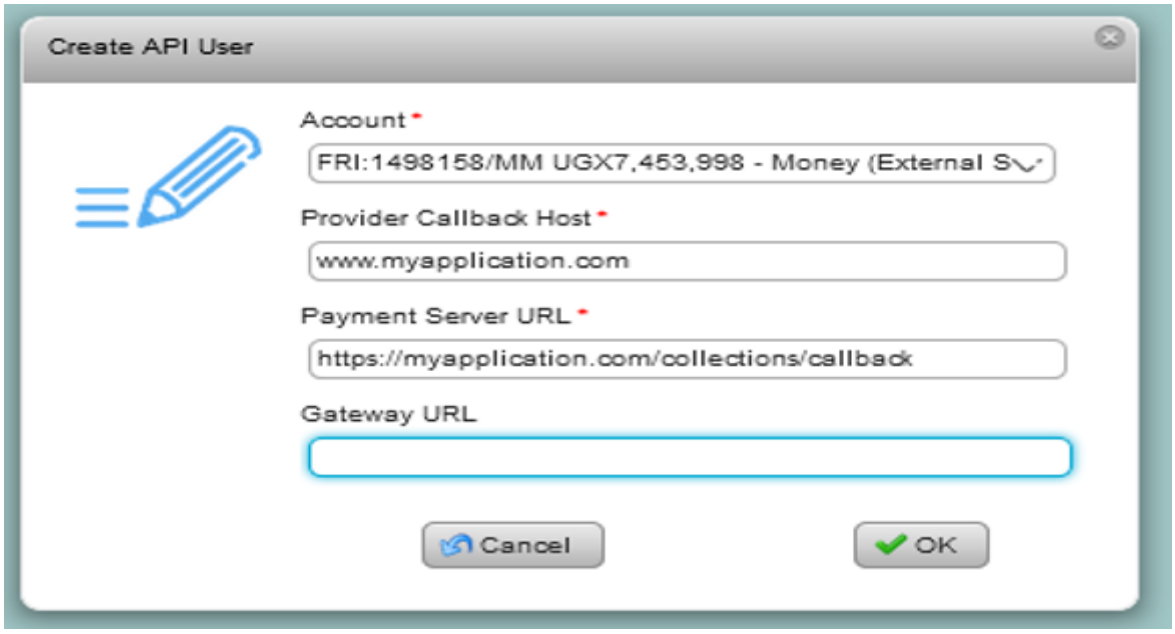
3.2.2.2.2 Create API user

Partner shall Click on the Top right under user profile to access the option.

Partner shall click on the Create API user Option shown below.



Partner shall fill the in the callback URL and select a transaction wallet.

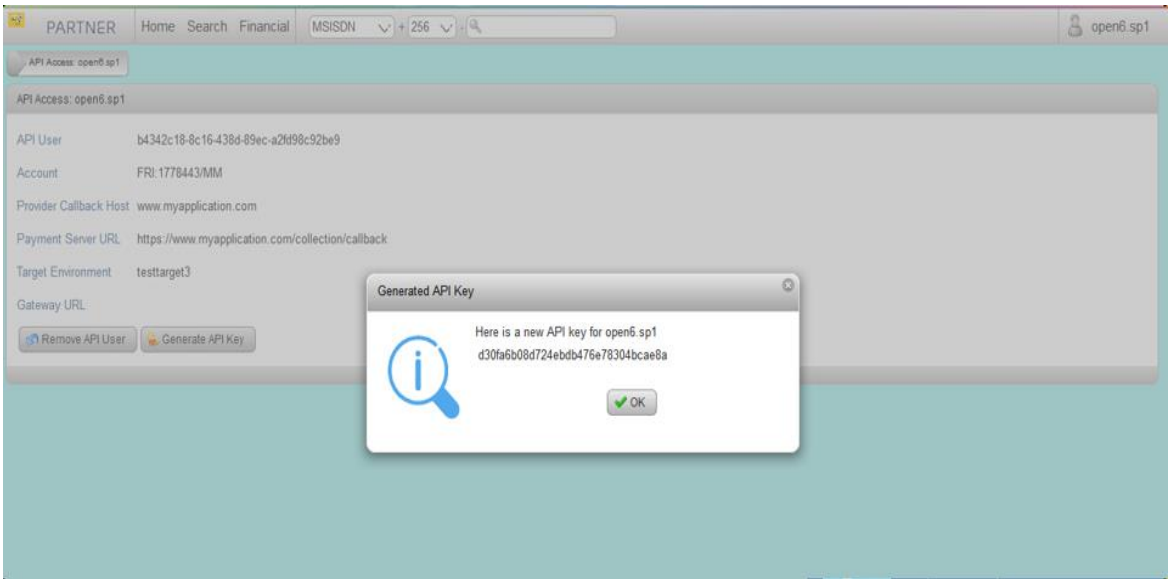


The table below describes the different fields required when creating an API User

Field	Optionality	Description
Account	**Mandatory**	Drop down option with available wallet. Partner shall choose main Transaction Wallet
Provider Callback Host	**Mandatory**	Partner Subdomain. Example www.myapplication.com
Payment Server URL	**Mandatory**	This shall be the callback url Example: https://myapplication.com/collections/callback
Gateway URL	**Optional**	This should be left empty.

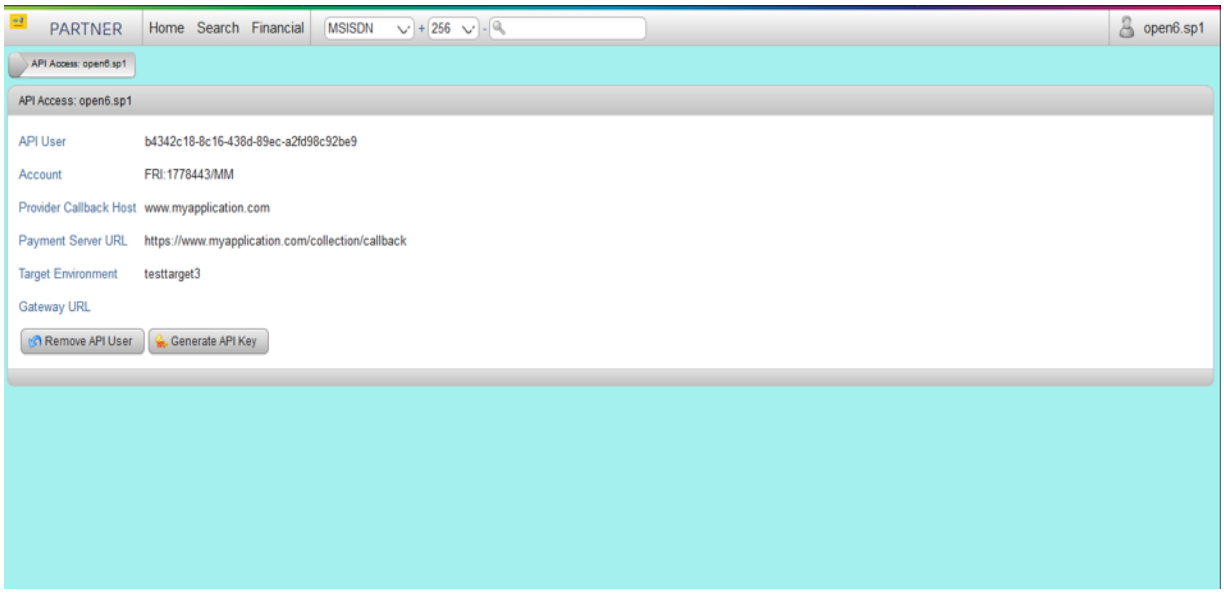
Partner shall click Ok after filling all the mandatory fields.

Upon submission ECW creates the API User and API key. API Key is displayed as a flash message as shown below.



Partner shall be required to note the API Key as it will not be possible to retrieve it.

Should Partner lose the API Key It's possible to regenerate a new Key for the Portal as shown below



Its also possible for partner to delete and re-create an API User.

3.2.3 Oauth 2.0

The Open API is using Oauth 2.0 token for authentication of request. Client will request an access token using Client Credential Grant according to RFC 6749. The token received is according to RFC 6750 Bearer Token.

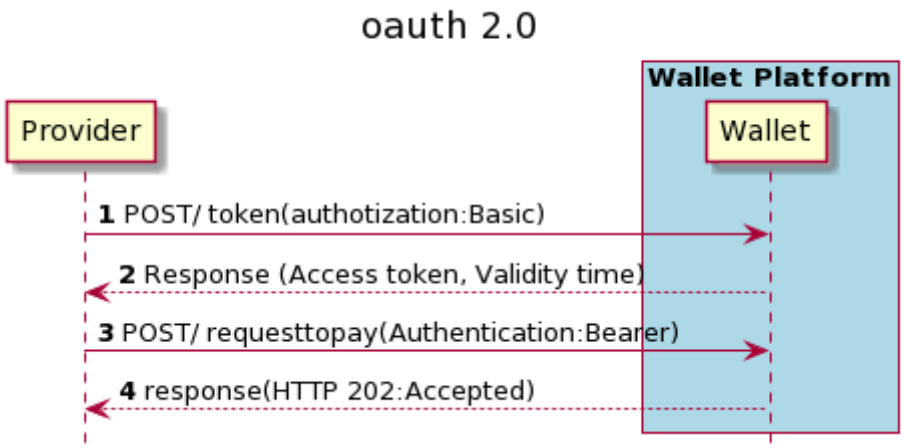
The API user and API key are used in the basic authentication header when requesting the access token. The API user and key are managed in the Partner GUI for the country where the account is located. The Partner can create and manage API user and key from the Partner GUI.

In case of sandbox the API Key and API User are managed through a Provisioning API as described on **3.2.2**

The received token have an expiry time. The token same token can be used used for transactions until it is expired. A new token is requested by using the POST /token service in the same way as for the initial token. The new token can be requested before the previous have expired to avoid authentication failure due to expired token.

Important: The token must be treated as a credential and kept secret. The party that have access to the token will be authenticated as the user that requested the token.

The below sequence describes the flow for requesting a token and using the token in a request.



1. Provider system request an access token using the API Key and API user as authentication.

2. Wallet platform authenticates credentials and respond with the access token

3. Provider system will use the access token for any request that is sent to Wallet Platform, e.g. POST /requesttopay.

Note: The same token shall be used if it is not expired.

3.3 API Methods

The API is using POST, GET, PUT methods. This section gives an overview of the interaction sequence used in the API and the usage of the methods. The Complete API definitions and Schema is found in the Swagger Definition found under API Sandbox.

3.3.1 POST

POST method is used for creating a resource in Wallet Platform. The request includes a reference id which is used to uniquely identify the specific resource that are created by the POST request. If a POST is using a reference id that is already used, then a duplication error response will be sent to the client.

Example: POST /requesttopay

The POST is an asynchronous method. The Wallet Platform will validate the request to ensure that it correct according to the API specification and then answer with HTTP 202 Accepted. The created resource will get status PENDING. Once the request has been processed the status will be updated to SUCCESSFUL or FAILED. The requester may then be notified of the final status through callback as described in **4.1**

3.3.2 GET

GET is used for requesting information about a specific resource. The URL in the GET shall include the reference of the resource. If a resource was created with POST then the reference id that was provided in the request is used a the identity of the resource.

Example:

POST /requesttopay request is sent with X-Reference-Id = 11377cbe-374c-43f6-a019-4fb70e57b617.

GET /requesttopay/11377cbe-374c-43f6-a019-4fb70e57b617 will return the status of the request.

3.3.3 PUT

The PUT method is used by the Open API when sending callbacks. Callback is sent if a callback URL is included in the POST request. The Wallet Platform will only send the callback once. There is no retry on the callback if the Partner system does not respond.

If the callback is not received, then the Partner system can use GET to validate the status.

4 Use cases

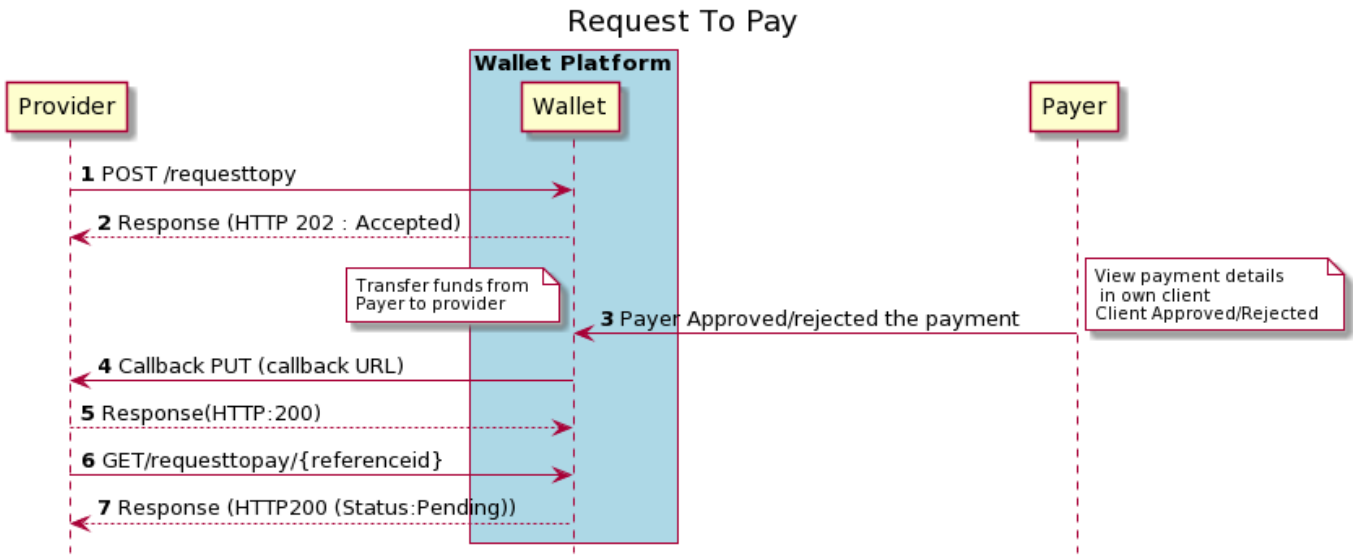
This section describes the services in the Open API.

Note: It's expected that the developer has an access token as described in 2.2.2

4.1 Request to Pay

Request to Pay service is used for requesting a payment from a customer. This can be used for example, an e-commerce website requesting a payment from a customer. The customer is requested to approve the transaction on the customer client.

The below sequence describes how the requesttopay service is used.



1. Customer (Payer) have selected product(s) in the merchant web shop and decided to check out. Customer select to pay with Mobile Money.

2. The provider system collects the account information for the customer e.g. mobile number and calculate the total amount of the products.

3. The provider system sends a request to pay (POST /requesttopay) operation to Wallet Platform. This request includes the amount and customer (Payer) account holder number.

4. Wallet Platform will respond with HTTP 202 Accepted to the provider system

5. Provider shall inform the customer that a payment needs to be approved, by giving information on the merchant web page. For example, the merchant could show information that payment is being processed and that customer needs to approve using the own client, e.g. USSD, mobile app.

6. Wallet Platform will process the request so that the customer can approve the payment. The request to pay will be in PENDING state until the customer have approved/Rejected the payment.

7. The Customer (Payer) will use his/her own client to review the payment. Customer can approve or reject the payment.

8. Wallet platform will transfer the funds if the customer approves the payment. Status of the payment is updated to SUCCESSFUL or FAILED.

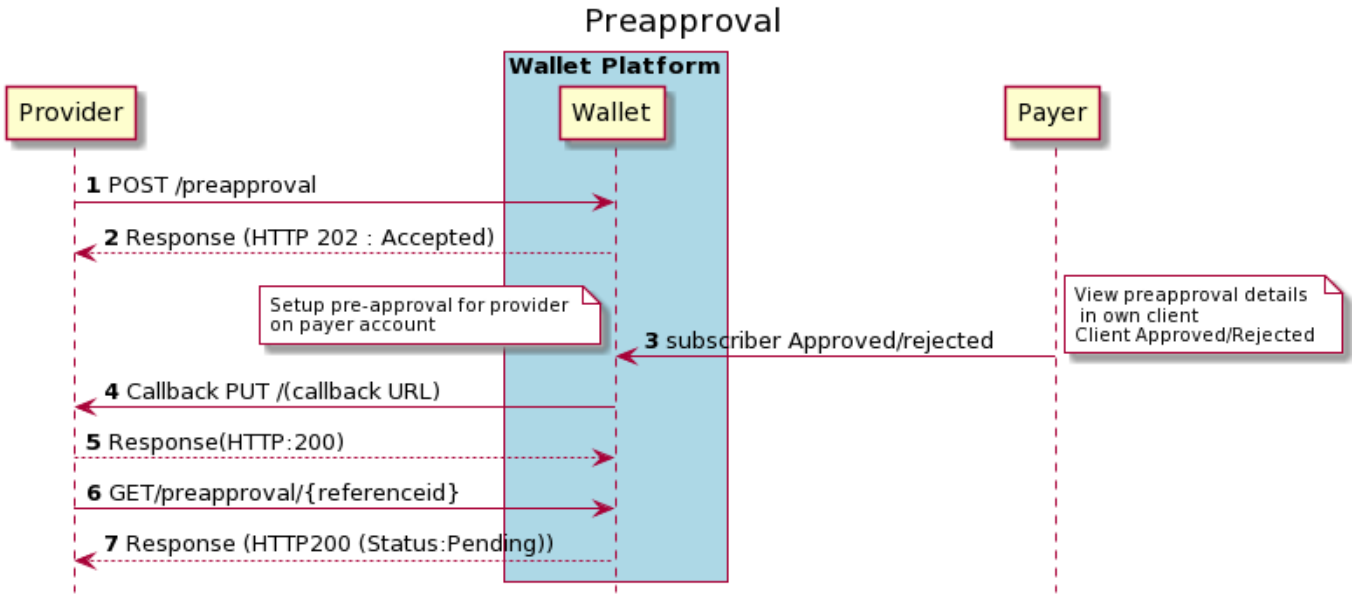
9. If a callback URL was provided in the POST /requesttopay then a callback will be sent once the request to pay have reached a final state (SUCCESSFUL, FAILED). Note the callback will only be sent once. There is no retry.

10. GET request can be used for validating the status of the transaction. GET is used if the partner system has not requested a callback by providing a callback URL or if the callback was not received.

4.2 Pre-Approval

Pre-approval is used to setup an auto debit towards a customer. The Partner can request a pre-approval from the customer. Once the customer has approved then the partner can debit the customer account without authorization from the customer.

The call flow for setting up a pre-approval is like the request to pay use case. The following picture describes the sequence for pre-approval.

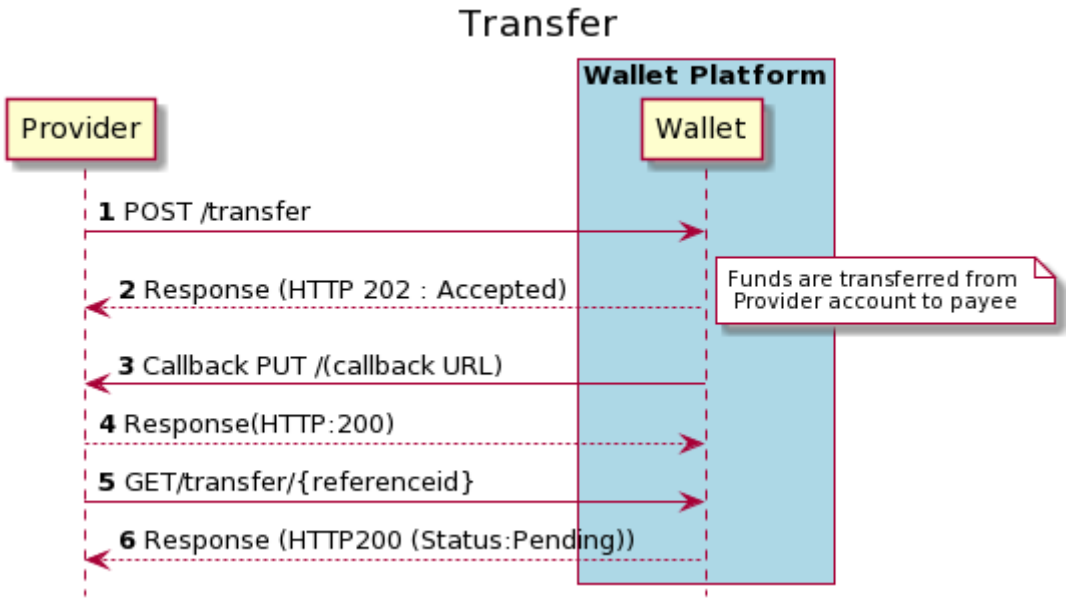


1. The Provider sends a POST /preapproval request to Wallet platform.
2. Provider shall inform the customer that pre-approval needs to be approved.
3. Customer (Payer) will use the own client to view the pre-approval request. Customer can approve or reject the request.
4. Callback will be sent if a callback URL was provided in the POST request. The callback is sent when the request has reach a final state (Successful, Failed).
5. The Provider can use the GET request to validate the status of the pre-approval.

4.3 Transfer

Transfer is used for transferring money from the provider account to a customer.

The below sequence gives an overview of the flow of the transfer use case.



5. The Provider sends a POST /transfer request to Wallet platform.

6. Wallet platform will directly respond to indicate that the request is received and will be processed.

7. Wallet platform will authorize the request to ensure that the transfer is allowed. The funds will be transferred from the provider account to the Payee account provided in the transfer request.

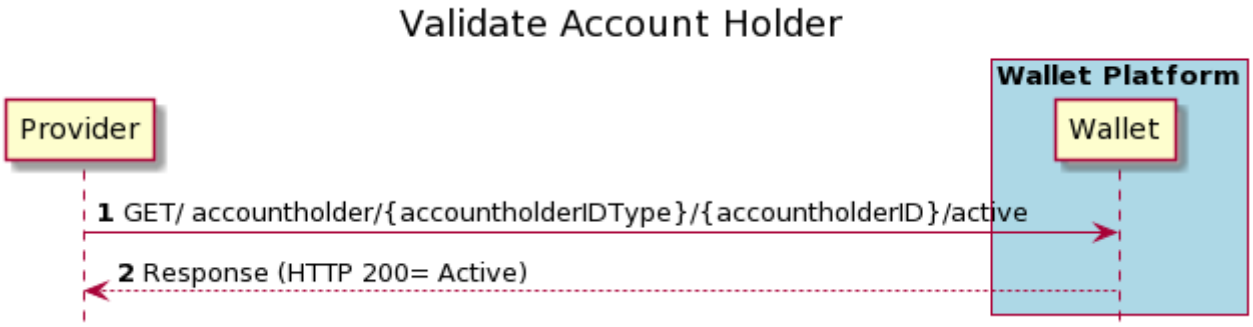
8. Callback will be sent if a callback URL was provided in the POST request. The callback is sent when the request has reach a final state (SUCCESSFUL, FAILED).

9. The Provider can use the GET request to validate the status of the transfer.

4.4 Validate Account Holder

Validate account holder can be used to do a validation if a customer is active and is able to receive funds. The use case will only validate that the customer is available and active. It does not validate that a specific amount can be received.

The sequence for the validate account holder is described below.

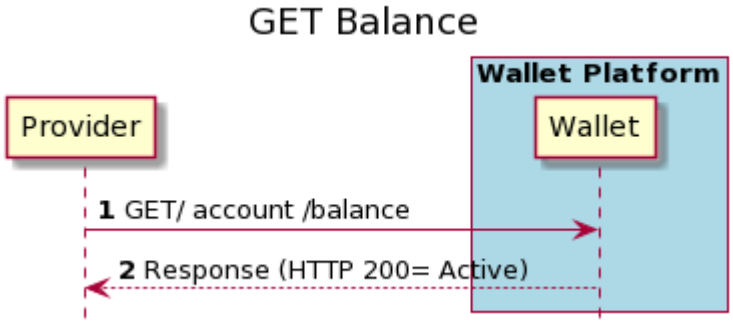


1. The Partner can send a GET /accountholder request to validate is a customer is active. The Partner provides the id of that customer as part of the URL

2. Wallet platform will respond with HTTP 200 if the account holder is active.

4.5 Get Balance

Get balance request is used to check the balance on the default account connected to the API User. The following is the sequence flow for get balance use case.



1. The partner will send a GET /account/balance request

2. Wallet platform will respond with the available balance on the API user account.

4.6 Get Consumer information with Consent

Authentication as a service can be used to validate/authenticate customer information, by way of a service provider sending a request to the wallet platform, with or without customer consent. The consumer will authenticate via authorization service, and if required, the consumer will be required to provide consent. The scope is to validate or retrieve customer information.

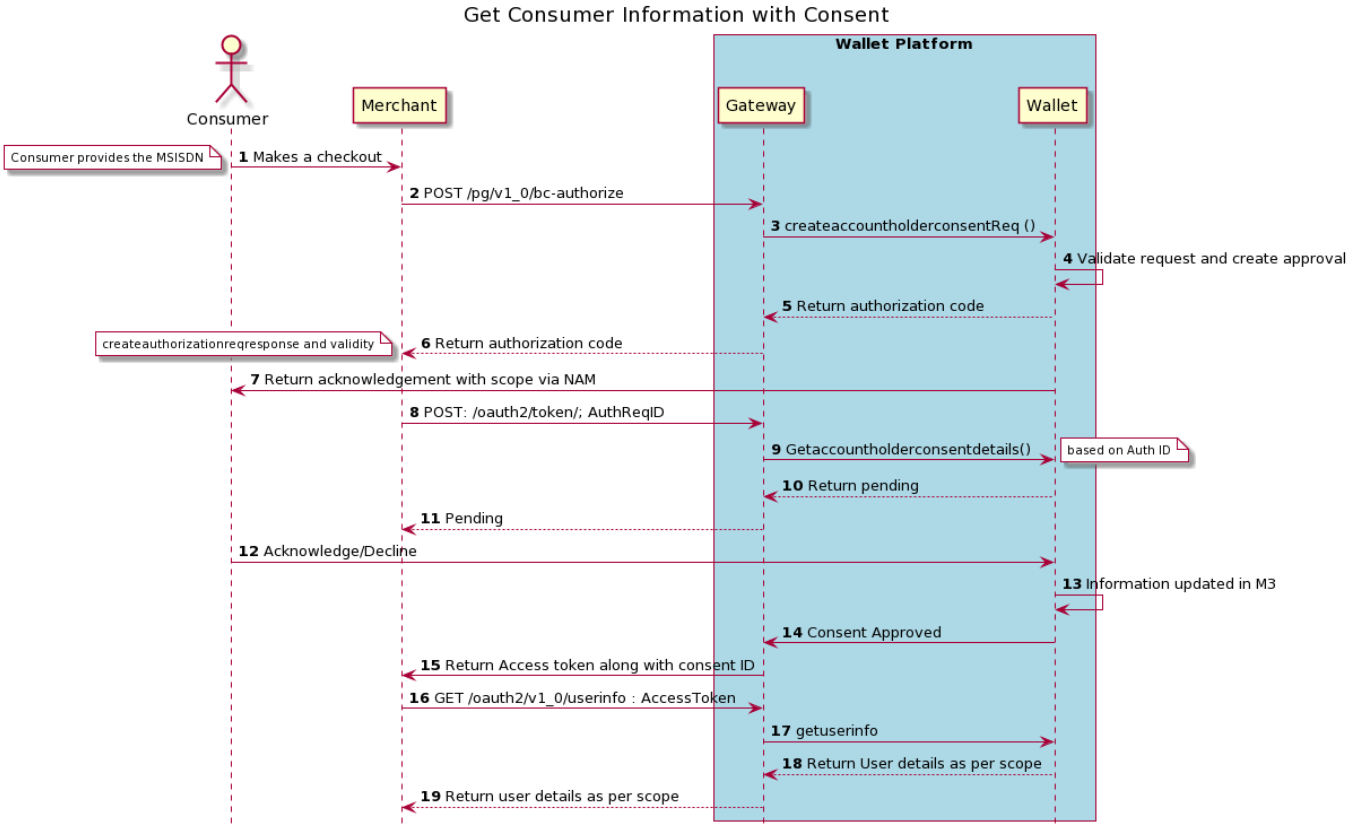
The authentication as a service capability will also enable a 3rd party provider to obtain consent, or digitally accept terms and conditions. This can be seen as a “digital signature” (where a partner requires).

Partners will be able to fetch limited user info without need for additional consent from customers. This will also enable partners to record and review that a customer has accepted the terms of a contract (amongst other use cases).

This will be used as a base when any token based “as a service “ is required to use.

API: bc-authorize

It provides the Partner the ability to fetch detailed information
about a Consumer after the Consumer has given the Partner consent for the retrieval of said information:



4.6.1 Token based API authorization

Token based API authorization is required to support consent management for different services. The Partner will request consent from the consumer for certain financial and non-financial transactions, and this will remain valid until the account holder revokes the consent.
The functionalities that can be enhanced by using consent will be

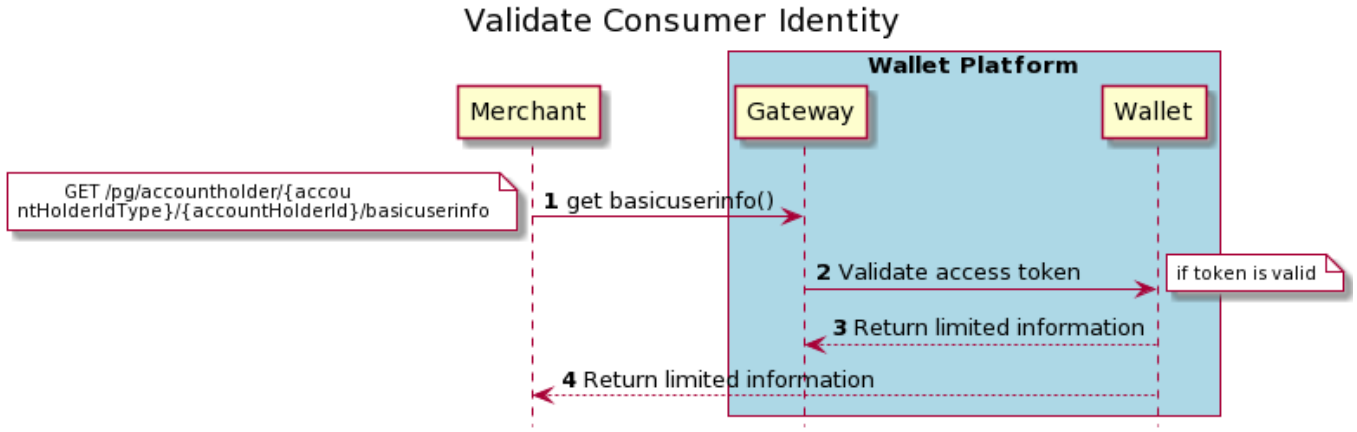
1. Transfer
2. Payment
3. Merchant payment
4. Transfer to any bank account

For partners this will allow to view the status of the transactions for which consent was give using related parameters.
For account holders, they can revoke the consent from any service at their will.
Business benefit of consent can be
Improved customer experience. Consent can be configured for a specific time and there will be no need to get consent for each transaction.
Providing better control on what services are being allowed by the account holder for consent etc.

4.7 Validate consumer identity

KYC as a service will enable a partner to retrieve (limited) customer KYC information with their consent. Upon request by the service provider, the customer will authorize, authenticate and provide consent to get detailed information.
The Partner will receive a short-lived token to fetch detailed information of the customer from the wallet platform..
The basic information that can be retrieved by default is
Profile: Name , Gender, Date of birth, locale etc.

API: GetBasicUserinfo can be used to either validate for example, Consumer's age or name.
The use case will return limited information about the Consumer that can be used to pre-populate fields, validate the Consumer's age or use it for remittance or sanctions screening purposes.

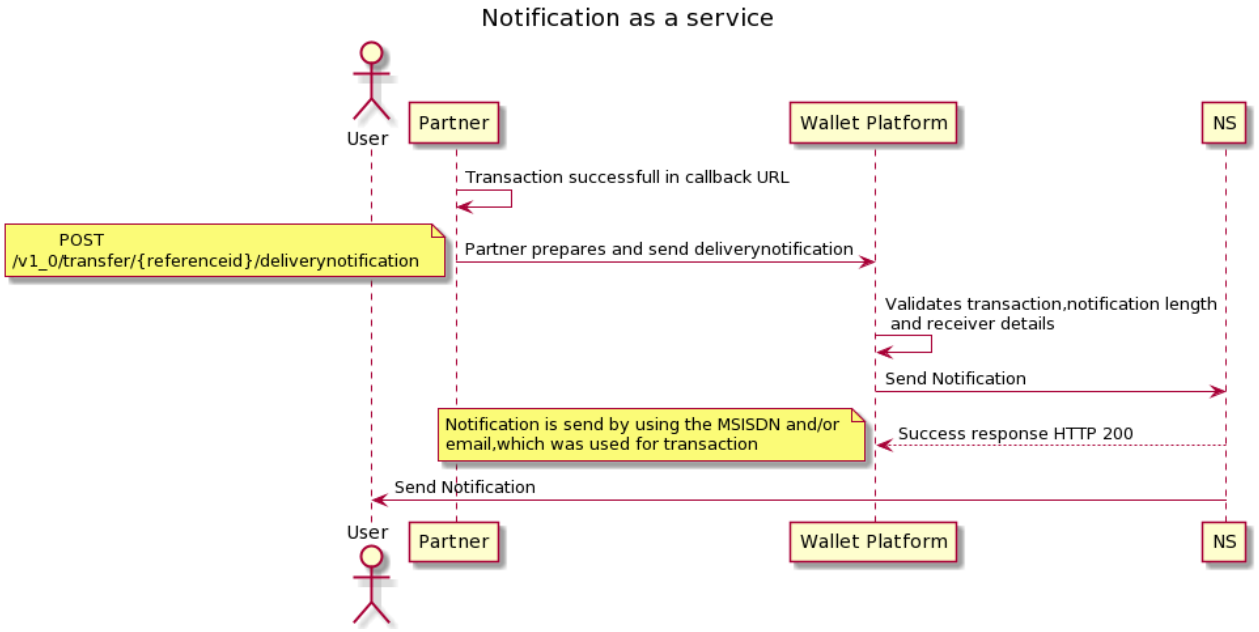


4.8 Delivery Notification

This service is intended to provide additional notification to a customer after the completion of a successful financial transaction, by SMS or by email.
Merchants and Service Providers using PGW to interact with EWP receive possibility to send extra information to their customers after completed transaction.
Extra information is sent via SMS is free text that contain information that a partner wants to communicate for example, delivery notification reference number, a lottery number, a booking number, ticket id etc.
The channel used to send the notification will be determined based on what identity was used to initiate original transaction/payment.
The time window during which partner can use additional notification is configured in Partner Gateway.

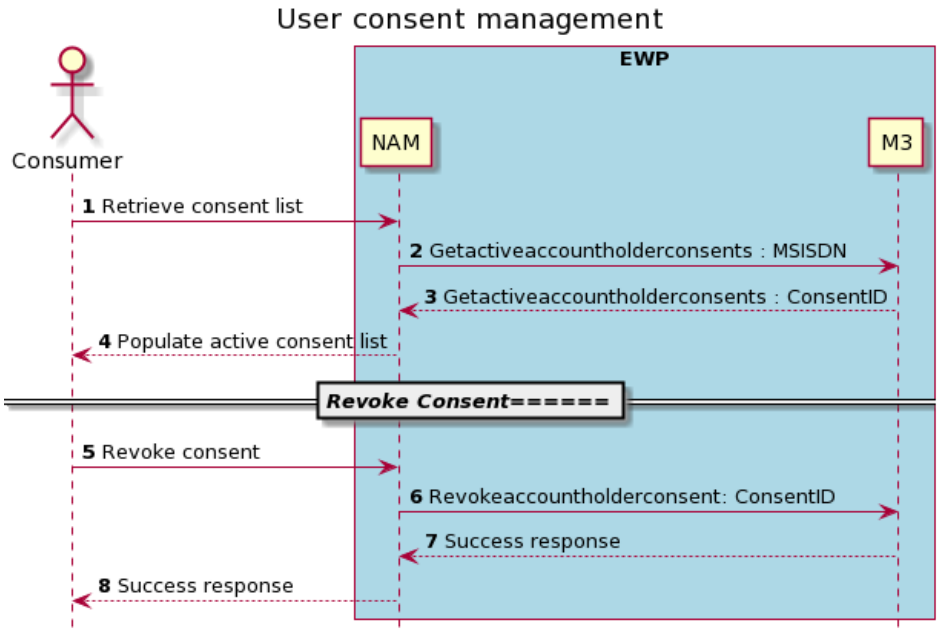
Sequence Flow
Actors
— Merchant (partner)
Prerequisites
— Merchant (partner) has an API user and key.
— A financial transaction has been completed.
Service flow
1. A financial transaction is completed by the Consumer.
2. The Partner sends a request for an additional delivery notification containing the ID of the completed transaction and a free text.

3. Partner Gateway validates the request by:
— Investigating when the financial transaction was made.
— Investigating the free text length. Transaction ids will be there by default so there will be 140 character free text area where merchant can configure the message content
— Finding the notification receiver.
4. Partner Gateway will then forward the free text message towards the end user via the Notification Service in the operator network where the account holder resides.
Partner Gateway uses MSISDN or email of the account holder to the sent message depending on if a MSISDN or email was used to identify the account holder during the transaction.
5. The delivery notification is completed.

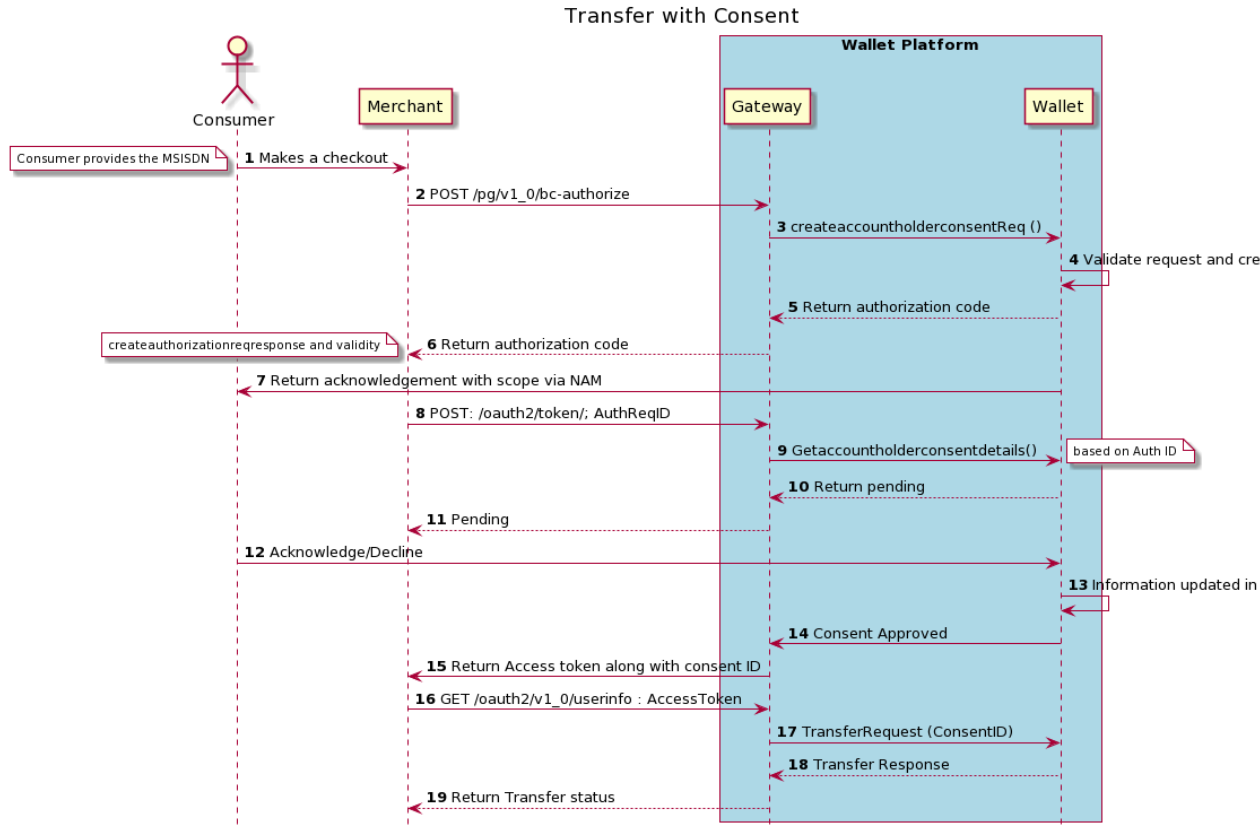


4.9 User consent View and Revoke

The End user (customer) may view the active consents and can revoke the active consents at any time.
User dials USSD code to retrieve the active consent
EWP returns the active consent list
User selects the consent to be revoked and sends the request back to EWP
EWP revokes the consent and returns success message to consumer.

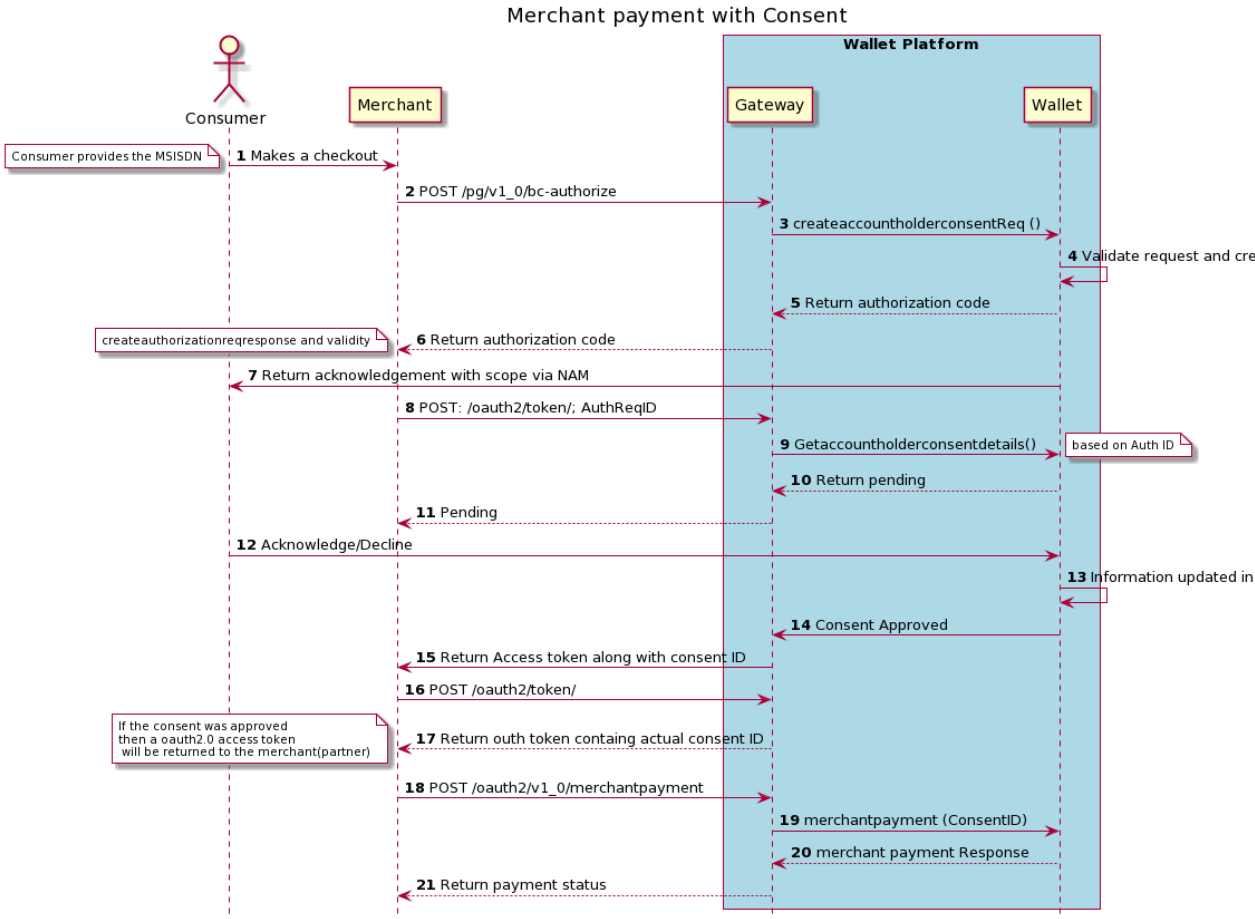


4.10 Transfer with consent sample sequence flow



Note: Bank transfer with consent flow will be similar to transfer with consent

4.11 Merchant payment with consent sequence flow

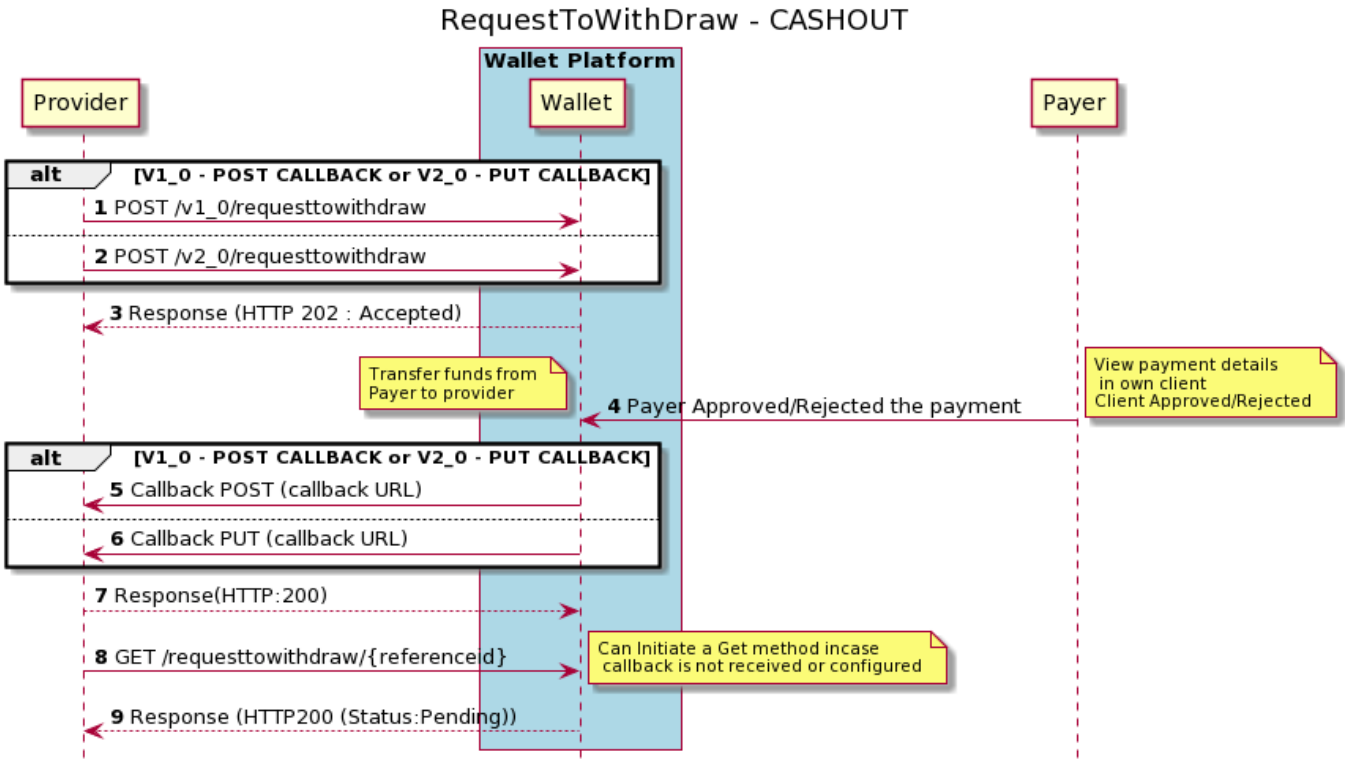


Note: Payment with consent will be similar to merchantpayment with consent

4.12 Request to Withdraw - CASHOUT

Request to Withdraw service is used for requesting a payment from a customer. This can be used for example, an e-commerce website requesting a payment from a customer. The customer is requested to approve the transaction on the customer client.

The below sequence describes how the requesttowithdraw service is used.



1. Agent (Partner) selects to request a withdrawal from a Consumer, specifies the MSISDN and the amount to withdraw.
2. The Partner system collects the account information for the Consumer such as mobile number and the amount to withdraw.
3. The Partner System sends a request to withdraw, POST /v1_0/requesttowithdraw (deprecated) or POST /v2_0/requesttowithdraw, to Wallet Platform. This request includes the amount and Consumer (Payer) account holder number.
4. Partner Gateway checks if the Agent (partner) is authorized to perform the request to withdraw operation.
5. Partner Gateway checks if the Agent (partner) has a valid access token and not expired token. If access token has expired, Partner system needs to create a new access token
6. Since request to withdraw operation is asynchronous, Partner Gateway creates the transaction and responds with HTTP 202 Accepted.
7. Partner Gateway sends initiatetransfer request on behalf of the Agent towards the Wallet Platform.
8. Wallet Platform checks if the Partner Gateway has authorization to perform the initiatetransfer request on behalf of the Agent.
9. Partner system informs the Consumer that a withdrawal (cash-out) needs to be approved.
10. Wallet Platform will process the request so that the consumer can approve the payment. The request to withdraw will be in PENDING state until the consumer have approved/rejected the withdrawal.
11. The Consumer (Payer) uses own client to review the request to withdraw. Consumer can approve or reject the request to withdraw.

12. Wallet platform transfers the funds if the consumer approves the payment. Status of the payment is updated to SUCCESSFUL or FAILED.
13. If a callback URL was provided in the request to pay then the callback PUT <callback-url> or POST <callback-url> (depending on version used in the request) will be sent once the request to pay have reached a final state (SUCCESSFUL, FAILED). Note the callback will only be sent once. There is no retry.

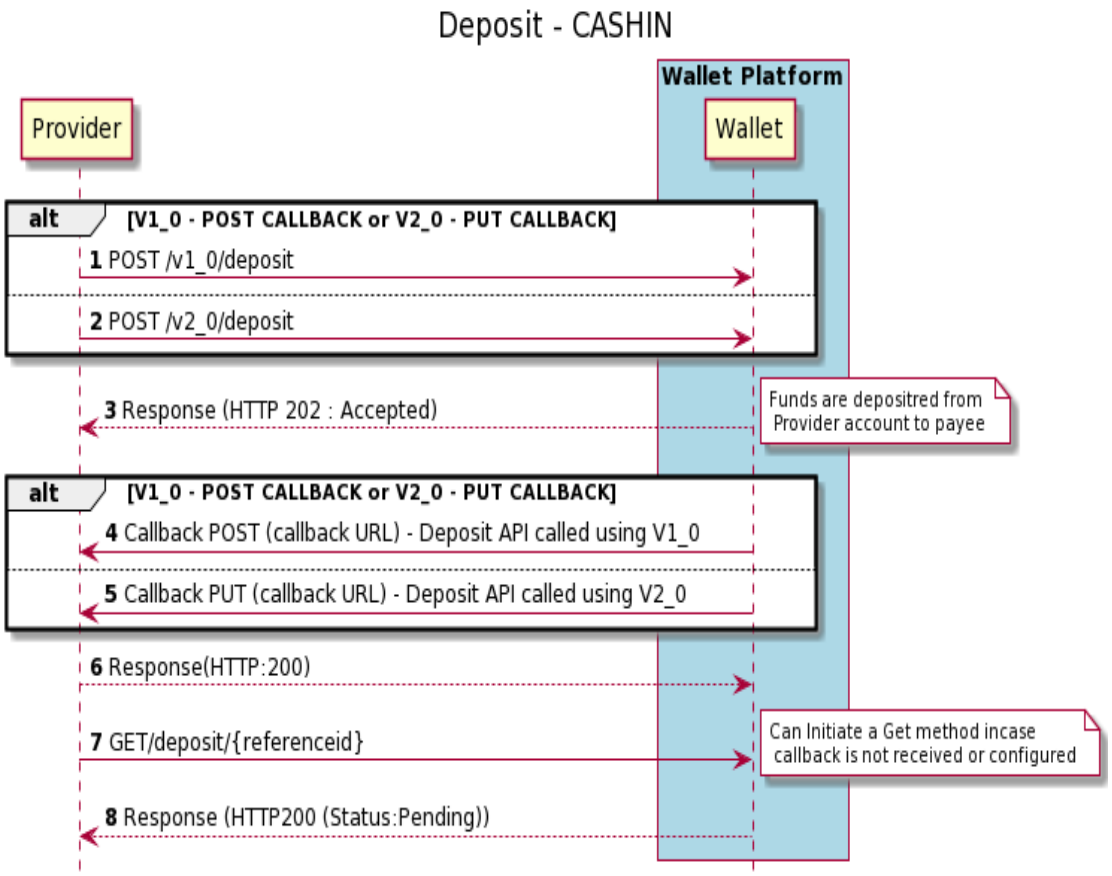
Alternatively the request GET /requesttowithdraw/{referenceId} can be used for validating the status of the transaction. The request should be used if the partner system has not requested a callback (no callback URL was provided), or if the callback was not received

14. Consumer receives the result. Notification will be sent towards consumer and

4.13 Deposit - CASHIN

Transfer is used for transferring money from the provider account to a customer.

The below sequence gives an overview of the flow of the transfer use case.



1. Agent (Partner) selects to perform a deposit (cash-in) to a Consumer, specifies the MSISDN and the amount to deposit.
2. The Partner system collects the account information for the Consumer such as mobile number and the amount to deposit.
3. Partner system sends a POST /v1_0/deposit (deprecated) or POST /v2_0/deposit
4. Partner Gateway checks if the Agent (partner) is authorized to perform the deposit operation.
5. Partner Gateway checks if the Agent's access token and not expired token. If access token has expired, Partner system needs to create a new access token
6. Since deposit operation is asynchronous, Partner Gateway creates the transaction and responds with HTTP 202 Accepted.
7. Partner Gateway sends a cashin request on behalf of the Agent towards the Wallet Platform.
8. Wallet Platform checks if the Partner Gateway has authorization to perform the cashin request on behalf of the Merchant.
9. Wallet Platform will process the request and transfer the funds. Status of the transfer is updated to SUCCESSFUL or FAILED.
10. If a callback URL was provided in the transfer request then the callback PUT <callback-url> or POST <callback-url> (depending on version used to create the transaction) will be sent once the request to pay have reached a final state (SUCCESSFUL, FAILED). Note the callback will only be sent once. There is no retry.

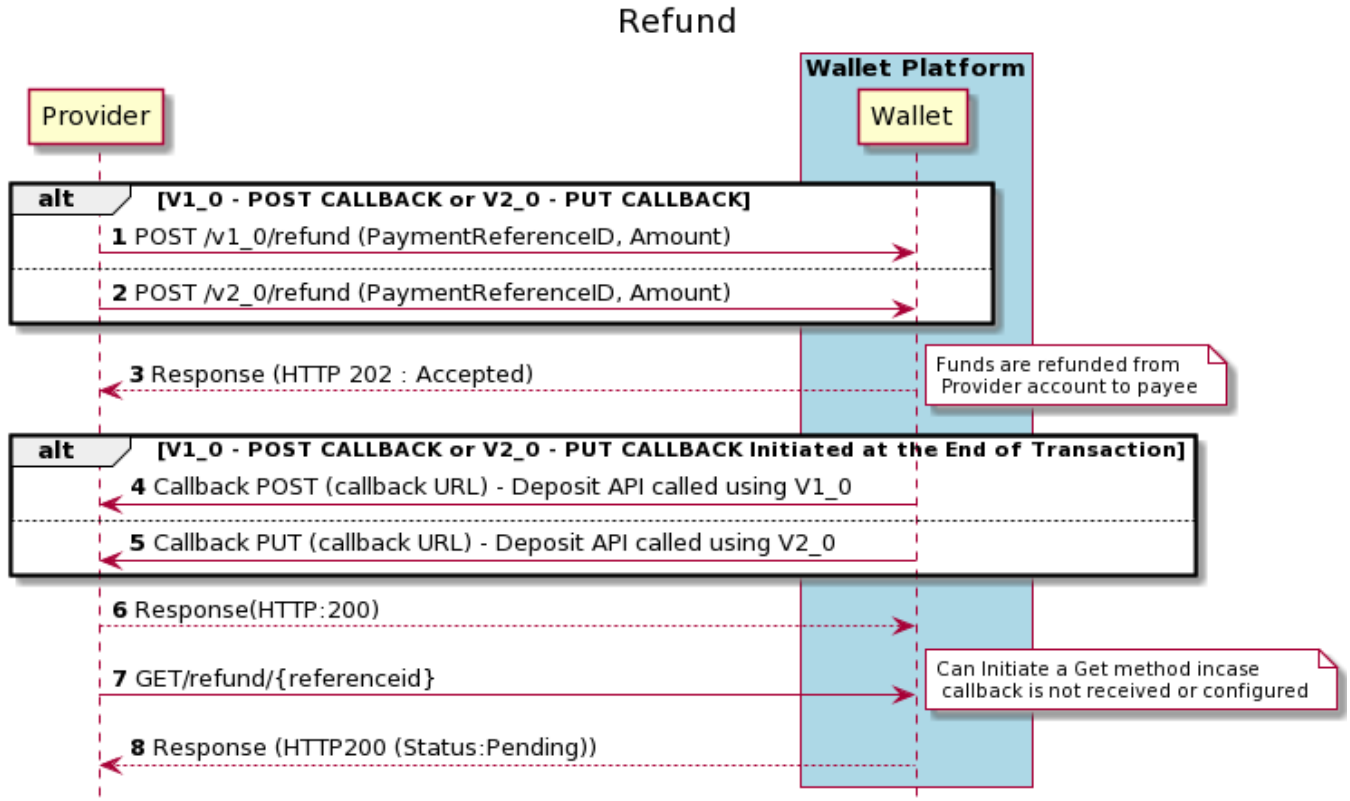
    Alternatively the request GET /deposit/{referenceid} can be used for validating the status of the transaction. The request should be used if the partner system has not requested a callback (no callback URL was provided), or if the callback was not received

11. Agent receives the result. Notification will be sent towards consumer and agent over the configured notification channel.

4.14 Refund

Refund is used for transferring money from the provider account to a customer incase a refund needs to be processed.

The below sequence gives an overview of the flow of the refund use case.



1. Agent (Partner) selects to perform a deposit (cash-in) to a Consumer, specifies the MSISDN and the amount to deposit.
2. The Partner system collects the account information for the Consumer such as mobile number and the amount to deposit.
3. Partner system sends a POST /v1_0/deposit (deprecated) or POST /v2_0/deposit
4. Partner Gateway checks if the Agent (partner) is authorized to perform the deposit operation.
5. Partner Gateway checks if the Agent's access token and not expired token. If access token has expired, Partner system needs to create a new access token
6. Since deposit operation is asynchronous, Partner Gateway creates the transaction and responds with HTTP 202 Accepted.
7. Partner Gateway sends a cashin request on behalf of the Agent towards the Wallet Platform.
8. Wallet Platform checks if the Partner Gateway has authorization to perform the cashin request on behalf of the Merchant.
9. Wallet Platform will process the request and transfer the funds. Status of the transfer is updated to SUCCESSFUL or FAILED.
10. If a callback URL was provided in the transfer request then the callback PUT <callback-url> or POST <callback-url> (depending on version used to create the transaction) will be sent once the request to pay have reached a final state (SUCCESSFUL, FAILED). Note the callback will only be sent once. There is no retry.

    Alternatively the request GET /deposit/{referenceid} can be used for validating the status of the transaction. The request should be used if the partner system has not requested a callback (no callback URL was provided), or if the callback was not received

11. Agent receives the result. Notification will be sent towards consumer and agent over the configured notification channel.

5 Common Error Codes

The complete definitions of error codes are found in the swagger documentation. Below is the list of error codes available.

5.1 Generic Error Codes

HTTP Code	Error Response Code	Description
409	N/A	Duplicated Reference Id. Cannot create new recourse
404	N/A	Reference Id not found. Requested resource does not exist.
400	N/A	Bad request. Request does not follow the specification.
401	N/A	Authentication failed. Credentials not valid
500	NOT_ALLOWED	Authorization failed. User does not have permission.
500	NOT_ALLOWED_TARGET_ENVIRONMENT	Not allowed target environment
500	INVALID_CALLBACK_URL_HOST	Callback URL with different host name then configured for API User
500	INVALID_CURRENCY	Currency not supported on the requested account
500	INTERNAL_PROCESSING_ERROR	Default error code used when there is no specific error mapping.
503	SERVICE_UNAVAILABLE	Service temporary unavailable, try again later

5.2 Preapproval Error Code

HTTP Code	Error Response Code	Description
500	PAYER_NOT_FOUND	Payer not found

5.3 RequestToPay Error Codes

Error Code	Error Response Code	Description
500	PAYER_NOT_FOUND	Payer not found. Account holder is not registered.
500	PAYEE_NOT_ALLOWED_TO_RECEIVE	Payee cannot receive funds due to e.g. transfer limit.

5.4 Transfer Error Codes

Error Code	Error Response Code	Description
500	NOT_ENOUGH_FUNDS	Not enough funds on payer account
500	PAYER_LIMIT_REACHED	Not allowed to end due to Payer limit reached
500	PAYEE_NOT_FOUND	Payee not found. Account holder is not registered

5.5 Validate Account Holder Error Codes

Error Code	Error Response Code	Description
404	N/A	Account holder is not found

5.6 Delivery Notification Error Codes

Error Code	Error Response Code	Description
400	N/A	Bad request (delivery message is invalid, invalid length)
410	N/A	Gone (DELIVERY_NOTIFICATION_EXPIRED)

409	N/A	The transaction is not successfully complete

6 Testing

To facilitate testing a set of predefined users and Test accounts are provided.

These users and accounts have a predefined test scenario.

The Sandbox URL is: *https://momodeveloper.mtn.com/*

Developer is required to Sign up and Subscribe to a Product before accessing any of the APIs.

6.1 Oauth Token

Oauth Token is generated from the merchants' API Key and Secret. API Key and Secret can be obtained through the provisioning API in Sandbox as described in **3.2.2**

6.2 Target Environment

The Target Environment used in Testing is **"sandbox"**

6.3 Test Currency

The currency used in Sandbox is EUR

6.4 Test Numbers

The following Numbers are predefined with respective response for all Testcases.

- 46733123450": // failed
- 46733123451": // rejected
- 46733123452": // Timeout
- 46733123453": // ongoing (will answer pending first and if requested again after 30 seconds it will respond success)
- 46733123454": // pending
- Any other Number results in Success