

# Deposits

Deposits allow you to request payment from a customer. Funds will be moved from the customer's mobile money wallet to your account in pawaPay. In this guide, we'll go through step by step on how to approach building a high quality deposit flow. If you haven't already, check out the following information to set you up for success with this guide.

## **What you should know**

Understand some consideration to take into account when working with mobile money.

## **How to start**

Sort out API tokens and callbacks.

Let's start by hard-coding everything. Throughout this guide we will be making the flow more dynamic and improving the customer experience incrementally.

1

### Initiate the deposit

Let's send this payload to the `initiate deposit` endpoint.

Copy  
Ask AI

```
POST https://api.sandbox.pawapay.io/v2/deposits
```

```
{  
  "depositId": "afb57b93-7849-49aa-babb-4c3ccbfe3d79",  
  "amount": "100",
```

```

    "currency": "RWF",
    "payer": {
      "type": "MMO",
      "accountDetails": {
        "phoneNumber": "250783456789",
        "provider": "MTN_MOMO_RWA"
      }
    }
  }
}

```

Let's see what's in this payload. We ask you to generate a UUIDv4 `depositId` to uniquely identify this deposit. This is so that you always have a reference to the deposit you are initiating, even if you do not receive a response from us due to network errors. This allows you to always **reconcile all payments** between your system and pawaPay. You should store this `depositId` in your system **before** initiating the deposit with pawaPay. The `amount` and `currency` should be relatively self-explanatory. This is the `amount` that will be collected from the customer. Any fees will be deducted from that amount after the collection has completed. Then the `provider` and `phoneNumber` specify exactly who is paying. Mobile money wallets are identified by the phone number, like bank accounts are by their account number. The `provider` specifies which mobile money operator this wallet is registered at. For example MTN Ghana or MPesa Kenya. You will receive a response for the initiation.

Copy  
Ask AI

```

{
  "depositId": "afb57b93-7849-49aa-babb-4c3ccbfe3d79",
  "status": "ACCEPTED",
  "nextStep": "FINAL_STATUS",
  "created": "2025-05-15T07:38:56Z"
}

```

The `status` shows whether this deposit was `ACCEPTED` for processing or not. We will go through failure modes later in the guide. We will also take a look later how `nextStep` can be used, but for now it's safe to ignore it.

2

The customer will approve the payment

Now that the deposit has been initiated, the customer will receive a PIN prompt on their phone to authorise the payment.

Tele2 EE



Local

Fri, May 16

10



Home

Fri, May 16

10



Enter PIN

This step only happens when initiating live payments on your production account. On your sandbox account, this step is skipped!

3

Get the final status of the payment

Now that the customer has approved the payment, you will receive a `callback` from us to your configured callback URL.

Copy  
Ask AI

```
{
  "depositId": "afb57b93-7849-49aa-babb-4c3ccbfe3d79",
  "status": "COMPLETED",
  "amount": "100.00",
  "currency": "RWF",
  "country": "RWA",
  "payer": {
    "type": "MMO",
    "accountDetails": {
      "phoneNumber": "250783456789",
      "provider": "MTN_MOMO_RWA"
    }
  },
  "customerMessage": "DEMO",
  "created": "2025-05-15T07:38:56Z",
  "providerTransactionId": "df0e9405-fb17-42c2-a264-440c239f67ed"
}
```

The main thing to focus on here is the `status` which is `COMPLETED` indicating that the deposit has been processed successfully and the funds have been collected to your wallet on your pawaPay account.

If you have not configured callbacks, you can always poll the `check deposit status` endpoint for the final status of the payment.

4

And done!

Congratulations! You have now made your very first deposit with pawaPay. We made a call to pawaPay to initiate the deposit. And we found out the final status

of that payment. But there's a little more to a high quality integration. In the next sections, we will go into more details on:

- How to make this easy to expand to new markets and providers
- How to handle failures so that discrepancies between your system and pawaPay don't happen
- How to cover countries with providers that use different authorisations methods for payments
- And more...

In the first part we hardcoded everything. Now let's take a look at how to ask that information from the customer who is paying. We are focusing on the use case where you know the amount to be paid already. Later in the guide, we will also cover the case where the customer can choose the amount to pay.

Some providers do not support decimals in amount, so amounts like "100.50" might not be possible. You can find how providers support decimals in the [Providers section](#).

1

## Showing the providers

The [active configuration](#) endpoint has useful and important information including which providers and markets have been configured on your pawaPay account. In most cases you already know from which country the customer is from. And since we are implementing deposits, we can fetch the configuration for deposits only.

Copy  
Ask AI

```
GET https://api.sandbox.pawapay.io/v2/active-  
conf?country=RWA&operationType=DEPOSIT
```

```
{
  "companyName": "Demo",
  "countries": [
    {
      "country": "RWA",
      "prefix": "250",
      "flag": "https://static-content.pawapay.io/country_flags/rwa.svg",
      "displayName": {
        "en": "Rwanda",
        "fr": "Rwanda"
      },
      "providers": [
        {
          "provider": "AIRTEL_RWA",
          "displayName": "Airtel",
          "logo": "https://static-content.pawapay.io/company_logos/airtel.png",
          "currencies": [
            {
              "currency": "RWF",
              "displayName": "R₣",
              "operationTypes": ...
            }
          ]
        }
      ],
      {
        "provider": "MTN_MOMO_RWA",
        "displayName": "MTN",
        "logo": "https://static-content.pawapay.io/company_logos/mtn.png",
        "currencies": [
          {
            "currency": "RWF",
            "displayName": "R₣",
            "operationTypes": ...
          }
        ]
      }
    ]
  ]
}
```

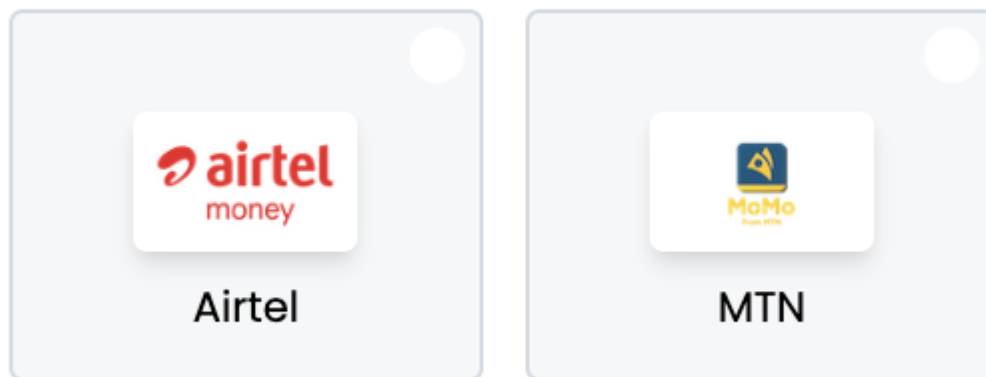
```

    }
    ]
    ...
}

```

As you can see, in `providers` we have a list of all the providers that have been configured on your account for initiating deposits. The `displayName` contains the name of the provider. You can show those as options to your customer. To enable new payment providers being automatically added to your deposit flows, we've also added the logo of the provider as the `logo`. This way it is possible to implement the deposit flow dynamically, so that when new providers are enabled on your pawaPay account, they become available for your customers without additional effort. This might look something like this:

## Operator



Do not have a default provider. For example, as the first selection in a dropdown. This often gets missed by customers causing payments to fail due to being sent to the wrong provider.

2

Ask for the phone number

In the `active configuration` endpoint there is already the `prefix` within the country object. This is the country calling code for that country. We recommend showing that to the customer in front of the input box for entering the phone number. This makes it clear that the number they enter should not include the country code. We have also added `flag` to the country object to make it look a little nicer. This might look something like this:

Amount

100.00 FRw

Phone number



+250

Enter your mobile number

3

Now let's validate the number and predict the provider

To make sure the number entered by the customer is a valid phone number, let's use the `predict provider` endpoint. First, concatenate the `prefix` to what was entered by the customer as the phone number. You can now send it to validation. We will handle basic things like whitespace, characters etc. We also make sure the number of digits is correct for the country and handle leading zeros.

Copy  
Ask AI

```
POST https://api.sandbox.pawapay.io/v2/predict-provider
```

```
{  
  "phoneNumber": "25007 834-56789a"  
}
```

If the phone number is not valid, a failure will be returned. You can show the customer a validation error. Otherwise, you will get the following response:

Copy  
Ask AI

```
{  
  "country": "RWA",  
  "provider": "MTN_MOMO_RWA",  
  "phoneNumber": "250783456789"  
}
```



The `phoneNumber` is the sanitized MSISDN format of the phone number that you can use to initiate the deposit. We strongly recommend using this endpoint for validating numbers especially due to some countries not strictly following ITU E.164. Also, in the response, you will find the `provider`. This is the predicted provider for this `phoneNumber`. We recommend preselecting the predicted provider for the customer. In most countries we see very high accuracy for predictions removing another step from the payment experience. We recommend allowing the customer to override the prediction as the accuracy is not 100%. You should now have a payment page similar to this:



Payment to  
**DEMO**

For  
**Example payment**

Amount

100.00 FRw

Phone number



+250

791675881

Operator



Airtel



MTN

Cancel

Pay

## Initiate the deposit

Now all information is either coming from the [active configuration](#) endpoint, the customer and your system (the amount to pay). We can call the [initiate deposit](#) endpoint with that information.

Copy  
Ask AI

```
POST https://api.sandbox.pawapay.io/v2/deposits
```

```
{
  "depositId": "afb57b93-7849-49aa-babb-4c3ccbfe3d79",
  "amount": "100",
  "currency": "RWF",
  "payer": {
    "type": "MMO",
    "accountDetails": {
      "phoneNumber": "250783456789",
      "provider": "MTN_MOMO_RWA"
    }
  }
}
```

The response and callback with the final status are the same as in the first part of the guide.

Please keep in mind that not all providers support amounts with decimals. You can find which providers do and which ones don't from the [Providers](#) section.

## Done again!

We now made a real deposit flow collecting information from different sources and avoiding any hard-coded data. This way, enabling new providers will not cause additional changes for you. Next, let's take a look at what to do while we are waiting for the customer to authorise the payment.

After you initiate a deposit, the customer will need to authorise it. While there are a couple of different ways providers handle payment authorisation, the most common one involves a PIN prompt popping up on the customers phone. In this step, we will focus on that type of authorisation flow. Later in the guide, we will cover the rest as well.

1

## Finding the payment authorisation type

We already used the [active configuration](#) endpoint to find the providers configured on your pawaPay account. The type of authorisation used by the provider is also available from that endpoint.

Copy  
Ask AI

```
GET https://api.sandbox.pawapay.io/v2/active-  
conf?country=SLE&operationType=DEPOSIT
```

```
{  
  "companyName": "Demo",  
  "countries": [  
    {  
      ...  
    },  
    "providers": [  
      {  
        "provider": "ORANGE_SLE",  
        "displayName": "Airtel",  
        "nameDisplayedToCustomer": "SL  
LIMITED",  
        "logo": "https://static-  
content.pawapay.io/company_logos/orange.png",  
        "currencies": [  
          {  
            "currency": "SLE",  
            "displayName": "Le",  
            "operationTypes": {
```

```

        "DEPOSIT": {
            ...
            "authType":
"PROVIDER_AUTH",
            "pinPrompt": "MANUAL",
            "pinPromptRevivable":
true
            ...
        }
    ]
}
    ]
}
    ]
    ...
}

```

There are three properties here that we will focus on. The `authType` shows the type of authorisation that this provider uses. We are focusing on `PROVIDER_AUTH` currently. Other `authTypes` will be covered later in the guide. The `pinPrompt` property is only applicable for providers with `authType` of `PROVIDER_AUTH`. It indicates whether or not the PIN prompt that is part of the authorisation will pop up on the customers phone automatically (`AUTOMATIC`) or if they need to take some action for that (`MANUAL`). The `pinPromptRevivable` shows whether it's possible for the customer to get the PIN prompt to pop up again in case it fails on the first attempt. Now that we know the specifics of the authorisation flow of the provider, let's implement the screen to show the customer after having initiated the deposit.

2

## Customer experience for AUTOMATIC `pinPrompt`

For `pinPrompt` value of `AUTOMATIC` we can show the customer a screen indicating that they should authorise the payment by entering their mobile money PIN on their phone. On the PIN prompt, there is usually a reference to the payment provider (pawaPay or one of its local subsidiaries). You should show that to the customer as part of the message to assure that it's a valid payment authorisation request. You can find it from the response as `nameDisplayedToCustomer`.

Copy  
Ask AI

GET <https://api.sandbox.pawapay.io/v2/active-conf?country=SLE&operationType=DEPOSIT>

```
{
  "companyName": "Demo",
  "countries": [
    {
      ...
    },
    "providers": [
      {
        "provider": "ORANGE_SLE",
        "displayName": "Airtel",
        "nameDisplayedToCustomer": "SL
LIMITED",
        ...
      }
    ]
  }
}
```

Here's an example of what you might show to the customer while waiting for the callback with the final status of this payment.



Please authorise the payment of 100.00  
LE to SL LIMITED by entering your PIN  
code on your phone

Powered by  pawaPay

3

What if the PIN prompt is revivable?

If `pinPromptRevivable` is true for the provider, it means that the customer can revive the PIN prompt in case something happens on the first attempt. For example, this might happen when they can't find their phone before the PIN prompt times out, they happen to get a call at the same time or there is a network

issue delivering the PIN prompt. In this case, we recommend waiting about 10-15 seconds after initiation and showing the customer instructions on how they can revive the PIN prompt to try again. The specific instructions are in the response from `active configuration` endpoint as 'pinPromptInstructions'.

Copy  
Ask AI

```
...
"pinPromptInstructions": {
  "channels": [
    {
      "type": "USSD",
      "displayName": {
        "en": "Did not get a PIN prompt?",
        "fr": "N'avez-vous pas reçu la demande de
code PIN?"
      },
      "instructions": {
        "en": [
          {
            "text": "Dial *115#",
            "template": "Dial {{shortCode}}",
            "variables": {
              "shortCode": "*115#"
            }
          },
          {
            "text": "Enter PIN",
            "template": "Enter PIN",
            "variables": {
              "shortCode": "*115#"
            }
          }
        ],
        "fr": [
          {
            "text": "Composez *115#",
            "template": "Composez
{{shortCode}}",
            "variables": {
              "shortCode": "*115#"
            }
          }
        ]
      }
    }
  ]
}
```



```

    }
  },
  {
    "text": "Entrez le code PIN",
    "template": "Entrez le code PIN",
    "variables": {
      "shortCode": "*115#"
    }
  }
]
},
...

```

The `channel` for reviving the PIN prompt indicates where the customer is able to revive the PIN prompt. This is predominantly `USSD` indicating they need to dial a USSD code on their phone to revive the PIN prompt. The `displayName` includes a message on the intent of the instructions. You are free to use it or come up with your own message. The `quickLink` (if available) includes the href that you can use to predial the USSD shortcode. You should use it as the `href` of an `<a />` tag or button.

This way, when the customer is using the same phone to visit your site that they are paying with, they can press to predial the USSD code. The `instructions` include the exact steps they should take to revive the PIN. You can iterate over them to show the instructions for reviving the PIN prompt. You can use the `text` of each instruction directly or if you want to emphasise key properties, you can use the `template` and `variables`. Just surround the `variables` of an instruction with the emphasis you need and replace them in the template.

4

## Customer experience for MANUAL `pinPrompt`

In case of `pinPrompt` value of `MANUAL`, the customer needs to take some action to get to the PIN prompt. While most customers know what to do and they will get an SMS with instructions anyway, we've seen improved success rates with clear instructions as part of payment experience. You can get the instructions to show for this exactly the same way as for PIN prompt revival.

Copy

## Ask AI

```
"pinPromptInstructions": {
  "channels": [
    {
      "type": "USSD",
      "displayName": {
        "en": "Follow the instructions to
authorise the payment",
        "fr": "Veuillez suivre les instructions
afin d'autoriser le paiement."
      },
      "instructions": {
        "en": [
          {
            "text": "Dial *555*6#",
            "template": "Dial {{shortCode}}",
            "variables": {
              "shortCode": "*555*6#"
            }
          },
          {
            "text": "Enter PIN",
            "template": "Enter PIN",
            "variables": {
              "shortCode": "*555*6#"
            }
          }
        ],
        "fr": [
          {
            "text": "Composez *555*6#",
            "template": "Composez
{{shortCode}}",
            "variables": {
              "shortCode": "*555*6#"
            }
          },
          {
            "text": "Entrez le code PIN",
            "template": "Entrez le code PIN",
```

```
        "variables": {
          "shortCode": "*555*6#"
        }
      }
    ]
  },
}
```

5

And done!

Now we have made sure that the customer always knows what they need to do to authorise a payment. Next let's take a look now how to handle failures.

Payment initiation can fail for various reasons. Let's see how we can best handle those.

1

Handling provider downtime

Providers may have downtime. We monitor providers performance and availability 24/7. For your operational and support teams, we have a [status page](#). From there they can [subscribe](#) to get updates in email or slack for all the providers they are interested in. To avoid failed payment attempts, we've also exposed this information over API from both the [provider availability](#) and [active configuration](#) endpoints. This way you can be up front with the customer before they attempt to pay.

Copy  
Ask AI

GET https://api.sandbox.pawapay.io/v2/active-conf?country=BEN&operationType=DEPOSIT

```
{
  "companyName": "Demo",
  "countries": [
    {
      "country": "BEN",
      "displayName": {
        "fr": "Bénin",
        "en": "Benin"
      },
      "prefix": "229",
      "providers": [
        {
          "provider": "MOOV_BEN",
          "nameDisplayedToCustomer": "PAWAPAY",
          "currencies": [
            {
              "currency": "XOF",
              "displayName": "CFA",
              "operationTypes": {
                "DEPOSIT": {
                  ...
                }
              }
            }
          ],
          "status":
"OPERATIONAL",
          ...
        }
      ]
    },
    {
      "provider": "MTN_MOMO_BEN",
      "nameDisplayedToCustomer": "PAWAPAY",
      "currencies": [
        {
          "currency": "XOF",
          "displayName": "CFA",
          "operationTypes": {
```

```

        "DEPOSIT": {
            ...
            "status": "CLOSED",
            ...
        }
    }
}
]
}
]
}
]
}
...
}

```

Based on whether the `status` of the specific provider is `OPERATIONAL` or `CLOSED` you can inform the customer up front that this provider is currently not available and they can attempt again later.

2

## Handling decimals in amount and transaction limits

Every time you [initiate a deposit](#) you should confirm that the `status` in the response is `ACCEPTED`. If the status is `REJECTED` the `failureReason` will contain both the `failureCode` and `failureMessage` indicating what has happened. Most of those failures are avoidable if handled beforehand.

The `failureMessage` from pawaPay API is meant for you and your support and operations teams. You are free to decide what message to show to the customer.

Copy  
Ask AI

```

//Incorrect amount of decimals in amount
{
    "depositId": "f4401bd2-1568-4140-bf2d-eb77d2b2b639",
    "status": "REJECTED",
    "failureReason": {
        "failureCode": "INVALID_AMOUNT",
        "failureMessage": "The provider MOOV_BEN only
supports up to '2' decimal places in amount."
    }
}

```

```

//Amount larger than transaction limits allow
{
  "depositId": "f4401bd2-1568-4140-bf2d-eb77d2b2b639",
  "status": "REJECTED",
  "failureReason": {
    "failureCode": "AMOUNT_OUT_OF_BOUNDS",
    "failureMessage": "The amount needs to be more
than '100' and less than '2000000' for provider 'MOOV_BEN'."
  }
}

//Invalid currency
{
  "depositId": "f4401bd2-1568-4140-bf2d-eb77d2b2b639",
  "status": "REJECTED",
  "failureReason": {
    "failureCode": "INVALID_CURRENCY",
    "failureMessage": "The currency 'USD' is not
supported with provider 'MOOV_BEN'. Please consult API docs
for supported currencies."
  }
}

```

In the above example, where the provider does not support decimal places in amount, you need to preempt that by rounding the amounts. Depending on your use case, you might do that preemptively by adjusting local pricing or dynamically before payment. Second thing to consider is that customers mobile money wallets have limits on how big the payments can be. Customers are able to get those limits increased on their mobile money wallets by contacting their provider and going through extended KYC. By default, we have set the transaction limits on your pawaPay account based on the most common wallet limits. Regarding currencies, the only country available through pawaPay that supports multiple currencies is [DRC](#). These are exposed in the [active configuration](#) endpoint as an array of [currencies](#). You can access both the decimal places supported and transaction limits from the [active configuration](#) endpoint.

Copy  
Ask AI

```

GET https://api.sandbox.pawapay.io/v2/active-
conf?country=BEN&operationType=DEPOSIT

```

```

{

```

```

    "companyName": "Demo",
    "countries": [
      {
        "country": "BEN",
        "displayName": {
          "fr": "Bénin",
          "en": "Benin"
        },
        "prefix": "229",
        "providers": [
          {
            "provider": "MOOV_BEN",
            "nameDisplayedToCustomer": "PAWAPAY",
            "currencies": [
              {
                "currency": "XOF",
                "displayName": "CFA",
                "operationTypes": {
                  "DEPOSIT": {
                    ...
                  }
                }
              }
            ],
            "decimalsInAmount":
              "NONE",
            "minAmount": "100",
            "maxAmount":
              "2000000",
            ...
          }
        ]
      },
      {
        "provider": "MTN_MOMO_BEN",
        "nameDisplayedToCustomer": "PAWAPAY",
        "currencies": [
          {
            "currency": "XOF",
            "displayName": "CFA",
            "operationTypes": {
              "DEPOSIT": {

```





```
"phoneNumber": "250783456789"
```

```
}
```

The `phoneNumber` in the response will be in the correct format to [initiate the deposit](#).

4

## One initiation per depositId

All payments initiated with pawaPay are idempotent based on their ID. You must always generate a new UUIDv4 for the `depositId`. When you attempt to use the same `depositId` more than once the payment will be rejected.

Copy  
Ask AI

```
{
```

```
  "depositId": "f4401bd2-1568-4140-bf2d-eb77d2b2b639",
```

```
  "status": "DUPLICATE_IGNORED"
```

```
}
```

5

## Handling provider availability

We already discussed how to show to the customer which payment methods are currently not available due to provider downtime. There is usually some time between when you fetch and show this information to the customer and when they initiate the payment. We still need to ensure accurate information is shown in case the provider goes down during that time. If a payment is initiated during provider downtime, it will be rejected.

Copy  
Ask AI

```
{
```

```
  "depositId": "f4401bd2-1568-4140-bf2d-eb77d2b2b639",
```

```
  "status": "REJECTED",
```

```
  "failureReason": {
```

```
    "failureCode": "PROVIDER_TEMPORARILY_UNAVAILABLE",
```

```
    "failureMessage": "The provider 'MTN_MOMO_BEN' is  
currently not able to process payments. Please consult our  
status page for downtime information on all providers.
```

```
Programmatic access is also available, please consult our API  
docs."
```

```
  }
```

```
}
```

Many customers will have mobile money wallets with different providers. It's reasonable to ask them to pay with another provider or to attempt again later.

6

## Handling HTTP 500 with failureCode UNKNOWN\_ERROR

The `UNKNOWN_ERROR` failureCode indicates that something unexpected has gone wrong when processing the payment. There is no `status` in the response in this case. It is **not safe** to assume the initiation has failed for this deposit. You should verify the status of the deposit using the `check deposit status` endpoint. Only if the deposit is `NOT_FOUND` should it be considered `FAILED`. We will take a look later in the guide, how to ensure consistency of payment statuses between your system and pawaPay.

7

And done!

We have now handled different failures that might happen during initiation. Next, let's take a look at payment failures that can happen during processing.

1

## Handling failures during processing

As the pawaPay API is asynchronous, you will get a `deposit callback` with the final status of the deposit. If the `status` of the deposit is `FAILED` you can find further information about the failure from `failureReason`. It includes the `failureCode` and the `failureMessage` indicating what has gone wrong. The `failureMessage` from pawaPay API is meant for you and your support and operations teams. You are free to decide what message to show to the customer.

Find all the `failure codes` and implement handling as you choose. Operation specific processing failures are also documented in the API reference:

- `Deposit callback`
- `Payout callback`
- `Refund callback`

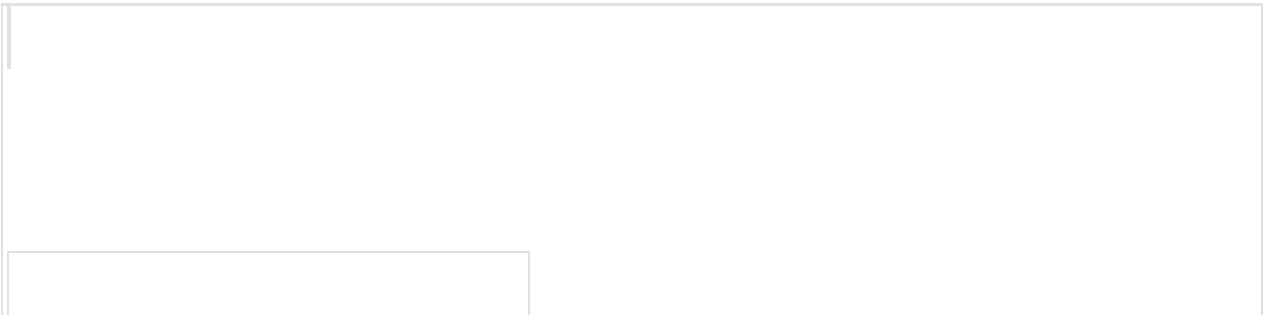
We recommend allowing easy retries for customers by taking the customer back to the payment information collection screen and showing the failure reason on that page. This way they can quickly try again.

We have standardised the numerous different failure codes and scenarios with all the different providers. The specificity of the failure codes varies by provider. The `UNSPECIFIED_FAILURE` code indicates that the provider indicated a failure with the payment, but did not provide any more specifics on the reason of the failure. In case there is a general failure, the `UNKNOWN_ERROR` failureCode will be returned.

2

And done!

We have now also taken care of failures that can happen during payment processing. This way the customer knows what has happened and can take appropriate action to try again. Now let's take a look at some markets where the payment authorisation flow is a little different than usual.



When working with financial APIs there are some considerations to take to ensure that you never think a payment is failed, when it is actually successful or vice versa. It is essential to keep systems in sync on the statuses of payments. Let's take a look at some considerations and pseudocode to ensure consistency.

1

## Defensive status handling

All statuses should be checked defensively without assumptions.

Copy  
Ask AI

```
if( status == "COMPLETED" ) {  
    myInvoice.setPaymentStatus(COMPLETED);  
} else if ( status == "FAILED" ) {  
    myInvoice.setPaymentStatus(FAILED);  
} else if ( status == "PROCESSING" ) {  
    handleRedirectionAuth();  
} else {  
    //It is unclear what might have failed. Escalate for  
    further investigation.  
    myInvoice.setPaymentStatus(NEEDS_ATTENTION);  
}
```

2

## Handling network errors and system crashes

The key reason we require you to provide a `depositId` for each payment is to ensure that you can always ask us what is the status of a payment, even if you never get a response from us. You should always store this `depositId` in your system *before initiating a deposit*.

Copy  
Ask AI

```
var depositId = new UUIDv4();  
  
//Let's store the depositId we will use to ensure we  
always have it available even if something dramatic happens  
myInvoice.setExternalPaymentId(depositId).save();  
myInvoice.setPaymentStatus(PENDING);  
  
try {  
    var initiationResponse =  
pawaPay.initiateDeposit(depositId, ...)  
} catch (InterruptedException e) {  
    var checkResult =  
pawaPay.checkDepositStatus(depositId);
```

```

        if ( result.status == "FOUND" ) {
            //The payment reached pawaPay. Check the status of
it from the response.
        } else if ( result.status == "NOT_FOUND" ) {
            //The payment did not reach pawaPay. Safe to mark
it as failed.
            myInvoice.setPaymentStatus(FAILED);
        } else {
            //Unable to determine the status. Leave the
payment as pending.
            //We will create a status recheck cycle later for
such cases.

            //In case of a system crash, we should also leave
the payment in pending status to be handled in the status
recheck cycle.
        }
    }
}

```

The important thing to notice here is that we only mark a payment as FAILED when there is a clear indication of it's failure. We use the `check deposit status` endpoint when in doubt whether the payment was `ACCEPTED` by pawaPay.

3

## Implementing an automated reconciliation cycle

Implementing the considerations listed above avoids almost all discrepancies of payment statuses between your system and pawaPay. When using callbacks to receive the final statuses of payments, issues like network connectivity, system downtime, and configuration errors might cause the callback not to be received by your system. To avoid keeping your customers waiting, we strongly recommend implementing a status recheck cycle. This might look something like the following.

Copy  
Ask AI

```

    //Run the job every few minutes.

    var pendingInvoices =
invoices.getAllPendingForLongerThan15Minutes();

    for ( invoice in pendingInvoices ) {

```

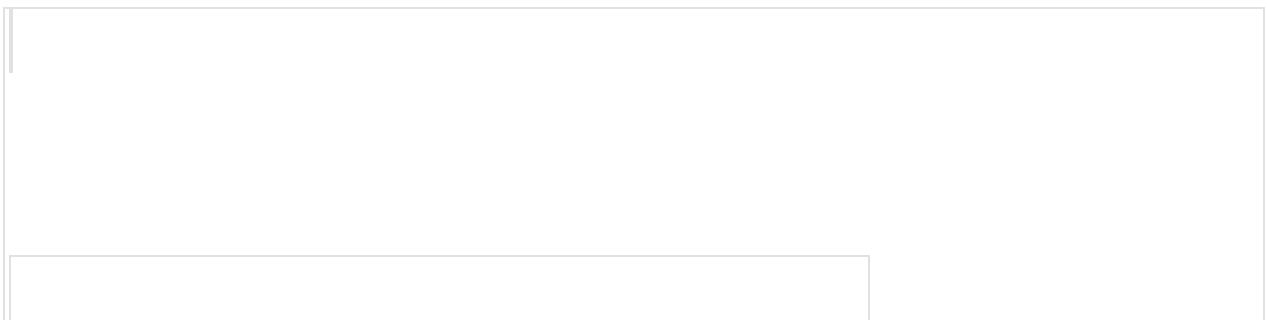
```

        var checkResult =
pawaPay.checkDepositStatus(invoice.getExternalPaymentId);

        if ( checkResult.status == "FOUND" ) {
            //Determine if the payment is in a final status
            and handle accordingly
            handleInvoiceStatus(checkResult.data);
        } else if (checkResult.status == "NOT_FOUND" ) {
            //The payment has never reached pawaPay. Can be
            failed safely.
            invoice.setPaymentStatus(FAILED);
        } else {
            //Something must have gone wrong. Leave for next
            cycle.
        }
    }
}

```

Having followed the rest of the guide, with this simple reconciliation cycle, you should not have any inconsistencies between your system and pawaPay. Having these checks automated will take a load off your operations and support teams as well.



Overwhelmingly providers use a PIN prompt to authorise payments. There are some markets that need support for alternative authorisation flows:

- Burkina Faso
- Ivory Coast
- Senegal

If you are not looking to go live in these markets, you can skip this section of the guide.

In this case, we recommend filtering out all providers that do not have `authType` of `PROVIDER_AUTH` from your calls to `active configuration`.

Let's implement support for these authorisation flows as well to ensure that all providers can be used.

## Preauthorised flows

1

Let's add support for preauthorised payments

Currently, the only provider using preauthorised payments is Orange in Burkina Faso. Preauthorisation means that the customer needs to authorise the payment *before* initiation the payment. For Orange in Burkina Faso the customer needs to generate an OTP through Oranges USSD menu. Let's make sure they know how to do that by showing them step-by-step instructions for that. Let's get those instructions from the `active configuration` endpoint.

Copy  
Ask AI

```
"authTokenInstructions": {
  "channels": [
    {
      "type": "USSD",
      "displayName": {
        "en": "Preauthorization instructions",
        "fr": "Instructions de préautorisation"
      },
      "instructions": {
        "en": [
          {
            "text": "Dial *144*4*6#",
            "template": "Dial {{shortCode}}",
            "variables": {
              "shortCode": "*144*4*6#"
            }
          }
        ]
      }
    }
  ],
}
```

```

        {
            "text": "Enter amount",
            "template": "Enter amount",
            "variables": {
                "shortCode": "*144*4*6#"
            }
        },
        {
            "text": "Enter PIN",
            "template": "Enter PIN",
            "variables": {
                "shortCode": "*144*4*6#"
            }
        },
        {
            "text": "Use OTP",
            "template": "Use OTP",
            "variables": {
                "shortCode": "*144*4*6#"
            }
        }
    ],
    "fr": [
        {
            "text": "Composez *144*4*6#",
            "template": "Composez
{{shortCode}}",
            "variables": {
                "shortCode": "*144*4*6#"
            }
        },
        {
            "text": "Entrez le montant",
            "template": "Entrez le montant",
            "variables": {
                "shortCode": "*144*4*6#"
            }
        },
        {
            "text": "Entrez le code PIN",

```



```

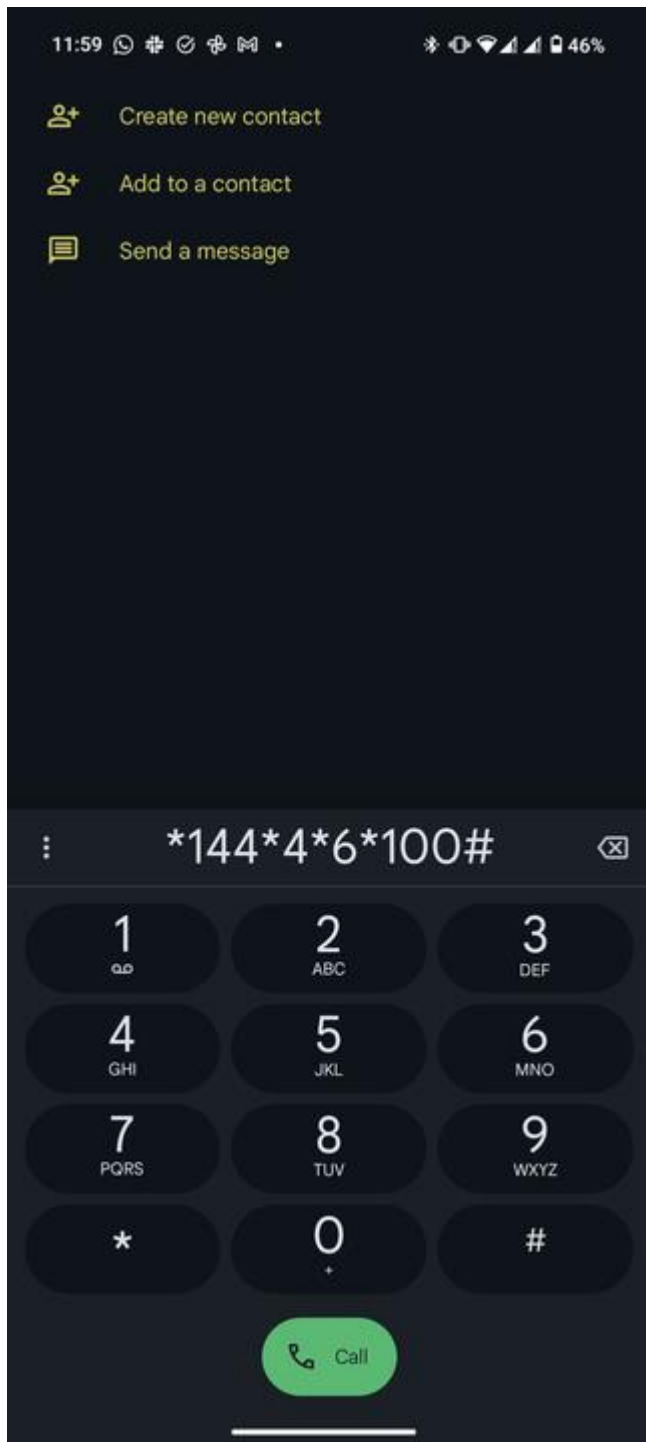
        "template": "Entrez le code PIN",
        "variables": {
            "shortCode": "*144*4*6#"
        }
    },
    {
        "text": "Entrez le code à usage
unique reçu par SMS",
        "template": "Entrez le code à
usage unique reçu par SMS",
        "variables": {
            "shortCode": "*144*4*6#"
        }
    }
]
}
},

```

The channel for reviving the PIN prompt indicates where the customer is able to generate the OTP. For Orange in Burkina Faso, this is USSD indicating they need to dial a USSD code on their phone to revive the PIN prompt. The displayName includes a message on the intent of the instructions. You are free to use it or come up with your own message. The quickLink includes the href that you can use to pre-dial the USSD shortcode. You should use it as the href of an a tag or button. This way, when the customer is using the same phone to visit your site that they are paying with, they can click to pre-dial the USSD code. The instructions include the exact steps they should take to generate the OTP. You can iterate over them to show the instructions to generate the OTP. You can use the text of each instruction directly or if you want to emphasise key properties, you can use the template and variables. Just surround the variables of an instruction with the emphasis you need and replace them in the template. You also need to provide the customer a place to provide the preauthentication OTP. The customer would then follow the instructions on their phone to generate the OTP.

2

Dial into the USSD menu of the provider



3

Enter their mobile money PIN to generate the OTP

Saisir le Code PIN pour  
générer un code OTP pour  
votre transaction en ligne  
100 FCFA

Appuyez sur \* pour le menu  
principal

---

CANCEL



# Operator message

OTP généré avec succès

**073394446**. Montant de

transaction: 100 FCFA

OTP: 367025

OK





Payment to  
**DEMO**

For  
**Example payment**

Amount

100.00 FCFA

Phone number



+226

73394446

OTP

367025

Dial **\*144\*4\*6\*100#** to get the OTP

Operator



Orange



Moov

6

You must then include this OTP as the `preAuthorisationCode` into the request to initiate deposit.

Copy  
Ask AI

```
{
  "depositId": "3bd57454-fc43-49ad-9949-54e03f173c85",
  "payer": {
    "type": "MMO",
    "accountDetails": {
      "phoneNumber": "22673394446",
      "provider": "ORANGE_BFA"
    }
  },
  "preAuthorisationCode": "367025",
  "amount": "100",
  "currency": "XOF"
}
```

You can now just tell the customer that the payment is processing and wait for a callback from pawaPay with the final status of the payment.

7

And done!

We can now get paid by customers using Orange in Burkina Faso. Now let's look at redirection based authorisation flows.

## Redirection based flows

Some providers use redirection based authorisation. This is slightly more involved as you will need to redirect the customer to a URL where they can authorise the payment. This is used by Wave in both Senegal and Ivory Coast. Let's look at how that might look like for a customer using Wave in Senegal.







Payment to  
**DEMO**

For  
**Example payment**

Amount

99.00 FCFA

Phone number



+221

773456789

Operator



Yas



WAVE



Orange

Cancel

Pay

## Initiate deposit

You can now **initiate the deposit**. After the authorisation is completed, the customer will be forwarded to...

- **successfulUrl** if the payment completed successfully.
- **failedUrl** if the payment failed to complete.

Copy  
Ask AI

```
{
  "depositId": "3bd57454-fc43-49ad-9949-54e03f173c85",
  "amount": "99",
  "currency": "XOF",
  "successfulUrl": "https://merchant.com/successfulUrl",
  "failedUrl": "https://merchant.com/failedUrl",
  "payer": {
    "type": "MMO",
    "accountDetails": {
      "phoneNumber": "221773456789",
      "provider": "WAVE_SEN"
    }
  }
}
```

You will receive a response for this with information on what to do next.

Copy  
Ask AI

```
{
  "depositId": "3bd57454-fc43-49ad-9949-54e03f173c85",
  "status": "ACCEPTED",
  "nextStep": "GET_AUTH_URL",
  "created": "2025-05-15T07:38:56Z"
}
```

The **nextStep** property indicates that the next thing to do is to retrieve the **authorizationUrl** to forward the customer to.

## Get the authorizationUrl

If you have configured `callbacks`, you will receive a callback with the `authorizationUrl`.

Copy  
Ask AI

```
{
  "depositId": "241cc3c4-74b8-4fbd-921d-340cf648e2a3",
  "status": "PROCESSING",
  "nextStep": "REDIRECT_TO_AUTH_URL",
  "amount": "99.00",
  "currency": "XOF",
  "country": "SEN",
  "payer": {
    "type": "MMO",
    "accountDetails": {
      "phoneNumber": "221773456789",
      "provider": "WAVE_SEN"
    }
  },
  "customerMessage": "DEMO",
  "created": "2025-05-15T07:38:56Z",
  "successfulUrl": "https://merchant.com/successfulUrl",
  "failedUrl": "https://merchant.com/failedUrl",
  "authorizationUrl":
    "https://provider.com/authorizationUrl"
}
```

If you do not have callbacks configured, you can always poll `check deposit status`. The response will keep returning `GET_AUTH_URL` as the value of `nextStep` until the provider makes the `authorizationUrl` available. Then the response will include the `authorizationUrl` and the value of `nextStep` will have changed to `REDIRECT_TO_AUTH_URL`.

Copy  
Ask AI

```
{
  "depositId": "241cc3c4-74b8-4fbd-921d-340cf648e2a3",
  "status": "PROCESSING",
  "nextStep": "REDIRECT_TO_AUTH_URL",
  "amount": "99.00",
  "currency": "XOF",
  "country": "SEN",
```

```

    "payer": {
      "type": "MMO",
      "accountDetails": {
        "phoneNumber": "221773456789",
        "provider": "WAVE_SEN"
      }
    },
    "customerMessage": "DEMO",
    "created": "2025-05-15T07:38:56Z",
    "successfulUrl": "https://merchant.com/successfulUrl",
    "failedUrl": "https://merchant.com/failedUrl",
    "authorizationUrl":
      "https://provider.com/authorizationUrl"
  }
}

```

This is usually very fast, but since the provider needs to return it, polling should be implemented.

4

Forward the customer to authorise the payment

You can now use the `authorizationUrl` that you retrieved in the previous step to forward the customer to it. If they are on desktop, they will see a QR code that they can scan with their phone that has the Wave app installed.



Payez avec  
**Wave**



**Scan the QR code to pay**

Download the Wave app on your phone



5

The customer can then authorise the payment

If the customer is using your site from the phone that has Wave app installed, they will be immediately redirected to the app to confirm the payment.

12:41

26%



310F



Scan



Send



Payments



Airtime



Bank







Rewards





...they will see a confirmation screen and then depending on the result of the payment, be forwarded back to either the `successfulUrl` or `failedUrl`.

15:02

    22%



-99F



Fee	0F
Status	Completed
Date & time	26 Jul 2024 12:48 pm
New balance	112F
Transaction ID	TNSKOLCULRWXTQWI

In partnership with Wave Digital Finance.



7

## Confirm the payment status

We will of course send you a callback with the final status of the payment, as usual. We do recommend validating the status of the deposit on your `successfulUrl` or `failedUrl` from the `check deposit status` endpoint.

8

## And done!

You have now built the payment experience allowing customers to pay you regardless of which provider they are using.

For some use cases, the customer should decide how much they are paying. Let's take a look at how to support that.

1

## Decimals support

Not all providers support decimals in amount - amounts like 100.50 will fail to process. This information is available for each provider in `active configuration` endpoint.

Copy  
Ask AI

```
GET https://api.sandbox.pawapay.io/v2/active-conf?country=BEN&operationType=DEPOSIT
```

```
{
  "companyName": "Demo",
  "countries": [
    {
```

```

        "country": "BEN",
        "displayName": {
            "fr": "Bénin",
            "en": "Benin"
        },
        "prefix": "229",
        "providers": [
            {
                "provider": "MOOV_BEN",
                "nameDisplayedToCustomer": "PAWAPAY",
                "currencies": [
                    {
                        "currency": "XOF",
                        "displayName": "CFA",
                        "operationTypes": {
                            "DEPOSIT": {
                                ...
                            }
                        }
                    }
                ],
                "decimalsInAmount":
                "NONE",
                ...
            }
        ]
    },
    {
        "provider": "MTN_MOMO_BEN",
        "nameDisplayedToCustomer": "PAWAPAY",
        "currencies": [
            {
                "currency": "XOF",
                "displayName": "CFA",
                "operationTypes": {
                    "DEPOSIT": {
                        ...
                    }
                }
            }
        ],
        "decimalsInAmount":
        "NONE",
        ...
    }
]

```

```

    }
  ]
}
]
}
]
}
]
...
}

```

The possible values are `NONE` and `TWO PLACES`. It's easy to provide dynamic validation now that is appropriate for the provider.

2

## Transaction limits

Customers' mobile money wallets have limits on how big the payments can be. Customers are able to get those limits increased on their mobile money wallets by contacting their provider and going through extended KYC. By default, we have set the transaction limits on your pawaPay account based on the most common wallet limits. The limits for the provider will be available from the [active configuration](#) endpoint.

Copy  
Ask AI

```
GET https://api.sandbox.pawapay.io/v2/active-
conf?country=BEN&operationType=DEPOSIT
```

```

{
  "companyName": "Demo",
  "countries": [
    {
      "country": "BEN",
      "displayName": {
        "fr": "Bénin",
        "en": "Benin"
      },
      "prefix": "229",
      "providers": [
        {
          "provider": "MOOV_BEN",
          "nameDisplayedToCustomer": "PAWAPAY",
          "currencies": [

```

```

    {
      "currency": "XOF",
      "displayName": "CFA",
      "operationTypes": {
        "DEPOSIT": {
          ...
          "minAmount": "100",
          "maxAmount":
"2000000",
          ...
        }
      }
    }
  ],
  },
  {
    "provider": "MTN_MOMO_BEN",
    "nameDisplayedToCustomer": "PAWAPAY",
    "currencies": [
      {
        "currency": "XOF",
        "displayName": "CFA",
        "operationTypes": {
          "DEPOSIT": {
            ...
            "minAmount": "100",
            "maxAmount":
"2000000",
            ...
          }
        }
      }
    ]
  }
]
...
}

```

It's easy to provide dynamic validation ensuring the amount is between `minAmount` and `maxAmount`.

3

And done!

Now customers can choose the amount to pay without making multiple attempts.

When using pawaPay, you might find that a payment status is `IN RECONCILIATION`. This means that there was a problem determining the correct final status of a payment. When using pawaPay all payments are reconciled by default and automatically - we validate all final statuses to ensure there are no discrepancies. When encountering payments that are `IN RECONCILIATION` you do not need to take any action. The payment has already been sent to our automatic reconciliation engine and it's final status will be determined soon. The reconciliation time varies by provider. Payments that turn out to be successful are reconciled faster.

We've made everything easy to test in our sandbox environment before going live.

**Test different failure scenarios**



We have different phone numbers that you can use to test various failure scenarios on your sandbox account.

## **Review failure codes**

Make sure all the failure codes are handled.

## **Add another layer of security**

To ensure your funds are safe even if your API token should leak, you can always implement signatures for financial calls to add another layer of security.

## **And when you are ready to go live**

Have a look at what to consider to make sure everything goes well.