Payouts

Payouts allow you to send money to a customer. Funds will be moved from your country wallet in pawaPay to the customers mobile money wallet.In this guide, we'll walk through how to build a high-quality payout flow step by step.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 considerations to take into account when working with mobile money.

How to start

Sort out API tokens and callbacks.

Let's start by hardcoding everything. Throughout this guide we will be taking care of more and more details to make sure everything is accurate.

1

Initiate the payout

Let's send this payload to the initiate payout endpoint.

Copy
Ask AI

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

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

Let's see what's in this payload.We ask you to generate a UUIDv4 `payoutId` to uniquely identify this payout. This is so that you always have a reference to the payout 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 payoutId in your system before initiating the payout with pawaPay.The `amount` and `currency` should be relatively self-explanatory. This is the `amount` that will be disbursed to the customer. Any fees will be deducted from your pawaPay wallet balance on successful completion of the payout. Principal amount (the amount specified in `amount`) will be reserved from your pawaPay wallet. If the payout fails, the funds will be returned to your pawaPay wallet immediately.Then the `provider` and `phoneNumber` specify who is paying exactly. Mobile money wallets are identified by the phone number, like bank accounts are by their IBAN. 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

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

The `status` shows whether this payout was `ACCEPTED` for processing or not. We will go through failure modes later in the guide.

2

The payout will be processed

Payouts do not require any authorisation or other action from the customer to receive it. The payout is usually processed within a few seconds. The customer will get an SMS receipt informing them that they have received a payout.

3

Get the final status of the payment

You will receive a callback from us to your configured callback URL.

Copy
Ask AI

```
    {
        "payoutId": "afb57b93-7849-49aa-babb-4c3ccbfe3d79",
        "status": "COMPLETED",
        "amount": "100.00",
        "currency": "RWF",
        "country": "RWA",
        "recipient": {
            "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 payout has been processed successfully.

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

4

And done!

Congratulations! You have now made your very first payout with pawaPay. We made a call to pawaPay to initiate the payout. 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 maintain

- How to handle failures so that discrepancies between your system and pawaPay don't happen
- And much more…

To make a payout to a customer, we need to have their valid phone number and need to know the provider they are using. We suggest validating that information when you are receiving that information whether that is during account signup or a separate process.

1

Asking 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 in front of the input box for entering the phone number. This makes it clear that the number entered should not include the country code.We have also added `flag` to the country object to make it look a little nicer.The providers that have been configured on your pawaPay account for payouts are in the `providers` property. It includes both the `provider` code to use in calls to initiate a payout and the `displayName` for it. To enable new providers being automatically added to your payout flows, we've also added the logo of the provider as the `logo`. This way it is possible to implement the payout flow dynamically, so that when new providers are enabled on your pawaPay account, they become available for your customers without additional effort.

Copy
Ask AI

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

    {
        "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": ...
                            }
                        ]
                    }
                ]
            }
        ]
        ...
    }
```

You can use the above information to collect the phone number together with the provider.

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

Validating the phone number

To make sure the phone number is a valid phone number, let’s use the predict provider endpoint.First, concatenate the `prefix` to what was entered 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 payout. 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 making it possible to override the prediction as the accuracy is not 100%.

3

Initiate the payout

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 payout endpoint with that information.

Copy
Ask AI

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

    {
        "payoutId": "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.

4

Done again!

We now made a real payout flow that collects information from different sources and avoids any hardcoded data. This way, enabling new providers will not cause additional changes for you.Now let’s handle the amounts correctly.

Depending on your use case, the amount will either be chosen by the customer or determined by you. As the support for decimals in amount differs by provider and transaction limits apply, let’s take a look at how to handle 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=PAYOUT

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

The possible values are `NONE` and `TWO_PLACES`. It’s easy to provide dynamic validation now that is appropriate for the provider. If the amount is determined by your system, you can apply rounding rules based on the above.If you are preparing the payout amounts beforehand and do not need to do it dynamically, this information is also documented in the providers section.

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=PAYOUT

    {
        "companyName": "Demo",
        "countries": [
            {
                "country": "BEN",
                "displayName": {
                    "fr": "Bénin",
                    "en": "Benin"
                },
                "prefix": "229",
                "providers": [
                    {
                        "provider": "MOOV_BEN",
                        "nameDisplayedToCustomer": "PAWAPAY",
                        "currencies": [
                            {
                                "currency": "XOF",
                                "displayName": "CFA",
                                "operationTypes": {
                                    "PAYOUT": {
                                        ...
                                        "minAmount": "100",
                                        "maxAmount":
"2000000",
                                        ...
                                    }
                                }
                            }
                        ]
                    },
                    {
                        "provider": "MTN_MOMO_BEN",
                        "nameDisplayedToCustomer": "PAWAPAY",
                        "currencies": [
                            {
                                "currency": "XOF",
                                "displayName": "CFA",
                                "operationTypes": {
                                    "PAYOUT": {
                                        ...
                                        "minAmount": "100",
                                        "maxAmount":
"2000000",
                                        ...
                                    }
                                }
                            }
                        ]
                    }
                ]
            }
        ]
        ...
    }
```

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

3

And done!

You now have validation that amount is correct for the payout.Let’s take a look at enqueued payouts next.

Providers might have downtime in processing payouts. Let's take a look at how to best handle cases when payout processing may be delayed or temporarily unavailable.

1

Checking if the provider is operational

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 the customer getting failed or delayed payouts, we've also exposed this information over API from the both the provider availability and active configuration endpoints. This way you can be up front with the customer that their payout might take more time to be delivered.Let's use the provider availability endpoint here as it's more common for payout use cases. We are filtering it to the specific country and operationType with the query parameters.

Copy
Ask AI

```
    GET
https://api.sandbox.pawapay.io//v2/availability?country=BEN&op
erationType=PAYOUT

    [
        {
            "country": "BEN",
            "providers": [
                {
                    "provider": "MOOV_BEN",
                    "operationTypes": {
                        "PAYOUT": "OPERATIONAL"
                    }
                },
                {
                    "provider": "MTN_MOMO_BEN",
                    "operationTypes": {
```

```
                                    "PAYOUT": "DELAYED"
                                }
                            }
                        ]
                    }
                ]
```
The values you might see are:

- `OPERATIONAL` - the provider is available and processing payouts normally.
- `DELAYED` - the provider is having downtime and all payout requests are being enqueued in pawaPay.
- `CLOSED` - the provider is having downtime and pawaPay is rejecting all payout requests.

Based on the above information you can inform the customer that their payout will be delayed (`DELAYED`) or is not possible at the moment and they can try again later (`CLOSED`).If your payouts are not initiated by customers, you can validate the status of the provider and delay initiating payouts while the provider is unavailable (`CLOSED`).

2

Handling delayed processing

We rarely close payouts processing. Mostly we switch off processing, but enqueue your payout requests to be processed when the provider is operational again.All payouts initiated when the status of the providers payouts is `DELAYED` will be `ACCEPTED` on initiation. They will then be moved to the `ENQUEUED` status. Our 24/7 payment operations team will be monitoring the provider for when their payout service becomes operational again. When that happens, the payouts will get processed as usual.To find out whether a payout is enqueued, you can check payout status.

Copy
Ask AI

```
    GET https://api.sandbox.pawapay.io/v2/payouts/37b250e0-3075-42c8-92a9-cad55b3d86c8

    {
        "status": "FOUND",
        "data": {
            "payoutId": "37b250e0-3075-42c8-92a9-
cad55b3d86c8",
            "status": "ENQUEUED",
            "amount": "100.00",
            "currency": "XOF",
            "country": "BEN",
            "recipient": {
                "type": "MMO",
                "accountDetails": {
                    "phoneNumber": "22951345789",
                    "provider": "MTN_MOMO_BEN"
                }
            },
            "customerMessage": "DEMO",
            "created": "2025-05-28T06:28:29Z",
        }
    }
```

You do not need to take any action to leave the payout to be processed once the provider is operational again.If you want to cancel the payout while it's enqueued, you use the cancel enqueued payout endpoint.

Copy
Ask AI

```
    GET https://api.sandbox.pawapay.io/v2/payouts/fail-
enqueued/37b250e0-3075-42c8-92a9-cad55b3d86c8

    RESPONSE:
    {
        "payoutId": "37b250e0-3075-42c8-92a9-cad55b3d86c8",
        "status": "ACCEPTED"
    }
```

The request is asynchronous. The payout must be in `ENQUEUED` status. If the payout is already in `PROCESSING` status, the request will be rejected.You will receive a callback with a `FAILED` status when cancellation has been completed.

Having implemented the above suggestions payout initiations should never be rejected. Let's take a look at a couple of edge cases to make sure everything is handled.

1

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 payout. You should verify the status of the payout using the check payout status endpoint. Only if the payout 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.

2

And done!

Initiating payouts should be handled now.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 payout callback with the final status of the payout. If the `status` of the payout 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.
We have standardised the numerous different failure codes and scenarios with all the different providers.The quality 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 would 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 how to ensure consistency of statuses between you and pawaPay.

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

2

Handling network errors and system crashes

The key reason we require you to provide a `payoutId` 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 `payoutId` in your system before initiating a payout.

Copy
Ask AI

```
    var payoutId = new UUIDv4();

    //Let's store the payoutId we will use to ensure we always
have it available even if something dramatic happens
    myOrder.setExternalPaymentId(payoutId).save();
    myOrder.setPaymentStatus(PENDING);

    try {
        var initiationResponse =
pawaPay.initiatePayout(payoutId, ...)
    } catch (InterruptedException e) {
        var checkResult = pawaPay.checkPayoutStatus(payoutId);

        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.
```

```
            myOrder.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 its failure. We use the check payout 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 pendingOrders =
orders.getAllPendingForLongerThan15Minutes();

    for ( order in pendingOrders ) {
        var checkResult =
pawaPay.checkPayoutStatus(order.getExternalPaymentId);

        if ( checkResult.status == "FOUND" ) {
            //Determine if the payment is in a final status
and handle accordingly
```

```
            handleOrderStatus(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.

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.