PAY FOR FUN

GO API (GATEWAY) – VERSION **2.8**

(download the latest version: https://api.p4f.com/developer/1.0/getgoapidocument)

TABLE OF CONTENTS

Contents

Changes from last API document version 1
Considerations 2
Introduction 3
API Pay4Fun GO 4
Planned Maintenance 5
API Endpoints 6
Supported Currencies 7
Obtaining Merchant’s Account Credentials 8
Understanding Merchant's Account Credentials 9
Whitelisting Pay4Fun’s Servers IPs 10
White Label Control 11
Pay In 12
Pay In Flow Overview 13
Pay In Algorithm 14
Pay In Flow Explained 16
Pay In Request 17
Pay In Response 23
Pay In Confirmation 25
Pay In Query Status 30
Payment Methods 36
PayOut 38
Screening Process Flow 40
PayOut Screening Algorithm 41
PayOut Screening Request 42
PayOut Screening Response 47

TABLE OF CONTENTS

PayOut Screening Confirmation ____ 49
PayOut Process Flow ____ 51
PayOut Process Algorithm ____ 52
PayOut Processing Request ____ 53
PayOut Processing Response ____ 56
PayOut Processing Confirmation ____ 59
PayOut Query Status ____ 61
Bank List ____ 67
Appendix 1 – Generating the Hash or Sign ____ 69
Appendix 2 – Error Messages ____ 71
Appendix 3 – Merchant Labels ____ 73
Appendix 4 – Security issues using iframes ____ 74
Appendix 5 – PayOut Test Data ____ 75
Appendix 6 – Pay4Fun Logo ____ 77

Changes from last API document version

Date	Page	Change
2021-12-16	60	Added unsuccessful confirmation response example
2022-01-26	14	Added Pay In Algorithm
2022-02-04	41	Added PayOut Screening Algorithm
2022-02-04	52	Added PayOut Processing Algorithm
2022-03-09		Added Release status description
2022-04-18		Branding parameters
2022-07-11		Request’s quota limits exclusion
2022-10-06		Added new error messages

Considerations

WARNING

The information within this document is subject to change without notice.

The software described in this document is provided under a license agreement and may be used or copied only in accordance with this agreement.

P4F makes no warranty, either express or implied, with respect to this product, its merchantability or fitness for a particular purpose, other than as expressly provided in the license agreement of this product. For further information, please contact us.

Introduction

PAY4FUN GO

Welcome to Pay4Fun, we allow consumers and merchants to make transactions quickly, conveniently and safely.

This API document is for Pay4Fun GO, a gateway system for direct deposit (Pay In) and withdrawal (Pay Out) on the merchant website.

Make sure you also integrate Pay4Fun (e-wallet) for more payment options to your customer. Please refer to the Pay4Fun (e-wallet) API document on https://api.p4f.com/developer/1.0/GetGoApiDocument/

API Pay4Fun GO

Pay4Fun's API works with a **redirect integration model**: you will redirect customers from your website to the Pay4Fun Authentication page where they type their Pay4Fun's credentials to complete the payment. Pay4Fun is responsible for:

- Validating the data entered by the customer;
- Ensuring that the data is held and disposed securely;

Important

Due security matters, **loading any Pay4Fun URL inside an iframe is not allowed!** Please check Appendix 4 – Security issues using iframes for more information.

This model allows you to accept payments from Pay4Fun, without adding complexity (such as handling payment data) to your own system. With the redirect model, you can also choose to:

- Specify your website URLs that redirect the customers after they complete the payment (success or fail);
- Dynamically control the minimum or maximum amount of payments that you want to offer;

Planned Maintenance

Occasionally, Pay4Fun will have to cause shorts periods of downtime in order to perform a planned maintenance; a proactive approach to execute needed actions in order to provide the maximum level of security and performance for our customers.

During such shorts periods of time, will not be possible to login at merchant's back-office and all API calls will return the following error:

Response code 503 - Service Unavailable

Response body

```
"Retry-After 2019-04-23 21:16:25 GMT"
```

Response headers

```
"content-type: application/json; charset=utf-8"
"date: Tue, 23 Apr 2019 21:11:25 GMT"
"retry-after: 2019-04-23 21:16:25 GMT"
```

Notice that the response header contains the GMT date and time (retry-after) which the planned maintenance is expected to end. You should gracefully handle this error at your side and wait until the maintenance period ends.

Important

No transaction will be processed or even stored at Pay4Fun's side during a planned maintenance.

API Endpoints

The P4F RESTful API, where all responses are in JSON, is available in the P4F Sandbox environment for integration testing purposes. To switch between the Sandbox and the live production system you only need to change the endpoint URI and the credentials.

The main URI **<P4F_API>** is defined below for Sandbox and Production environments. Please, check carefully the method explanation to use the right URI for each request:

Warning - Sandbox URL (Testing):

- **API:** **https://apitest.p4f.com**

Important – Production URL (Live stage)

- **API:** **https://api.p4f.com**

Supported Currencies

The following currencies are supported by P4F's API:

SUPPORTED CURRENCIES

ISO 4217 code	Name
BRL	Brazilian Real
GBP	British Pound
USD	United States Dollar
EUR	European Euro

Transactions will be processed according the specified currency at the request.

Make sure to send the correct information to the API. Pay4Fun takes no responsibility for incorrect amount or currency parameters sent to the API.

Obtaining Merchant's Account Credentials

After your account has been successfully approved by P4F, you will receive credentials to access the back-office (user name and password) where you will be able to retrieve the account credentials needed to configure the API requests accordingly.

Warning - Sandbox URL (Testing):

- **http://backtest.p4f.com/auth/merchant/login/**

Important – Production URL (Live stage)

- **https://back.p4f.com/auth/merchant/login/**

At the menu **ACCOUNT** / **API** / **CREDENTIALS**, subsection **DIRECT API**, you will find:

- Account Credentials:
  - **Merchant ID;**
  - **Merchant Key;**
  - **Merchant Secret;**
- Latest version of API document;
- Integration helper kit with JSON API requests;

Understanding Merchant's Account Credentials

Account credentials are available in your back-office for both staging and production environment. These credentials consist of:

MID (Merchant ID) - This is a unique identifier provided to every merchant by Pay4Fun. MID is part of your account credentials and is different on staging and production environment.

Merchant Key - This is a private key used for secure encryption of every request. This needs to be kept on server side and should not be shared with anyone.

Merchant Secret - This is a secret piece of information used as part of the seed string to generate communication hash. This needs to be kept on server side and should not be shared with anyone.

Important

Merchant secret and **Merchant key** are sensitive data and should be kept secure and safe from non-authorized personal and never should be included as plain text in any communication, electronic or not!

Whitelisting Pay4Fun's Servers IPs

In case you keep an IP whitelist to authorize access to your server, you must add Pay4Funs IPs to it.

Accessing the Merchants back-office, under the menu **ACCOUNT** / **API** / **CREDENTIALS**, you can check the IP list to be whitelisted for Pay4Fun's Test and Live stages.

Important

Not adding the IPs to your whitelist will avoid the API's confirmations to reach your servers.

White Label Control

PAY IN

If desired, Pay4Fun's API offers the possibility to control header color and logo display. To use the feature, the proper parameters should be sent on Pay In requests.

The parameter "layoutColor" must have its value set with a hexadecimal color code and it will control the header color on all Pay4Fun's interfaces.

The parameter "merchantLogo" must have its value set with a URL for the image to be displayed.

PAYOUT

If desired, Pay4Fun's API offers the possibility to control header color and logo display. To use the feature, the proper parameters should be sent on Screening requests.

The parameter "layoutColor" must have its value set with a hexadecimal color code and it will control the header color on all Pay4Fun's interfaces.

The parameter "merchantLogo" must have its value set with a URL for the image to be displayed.

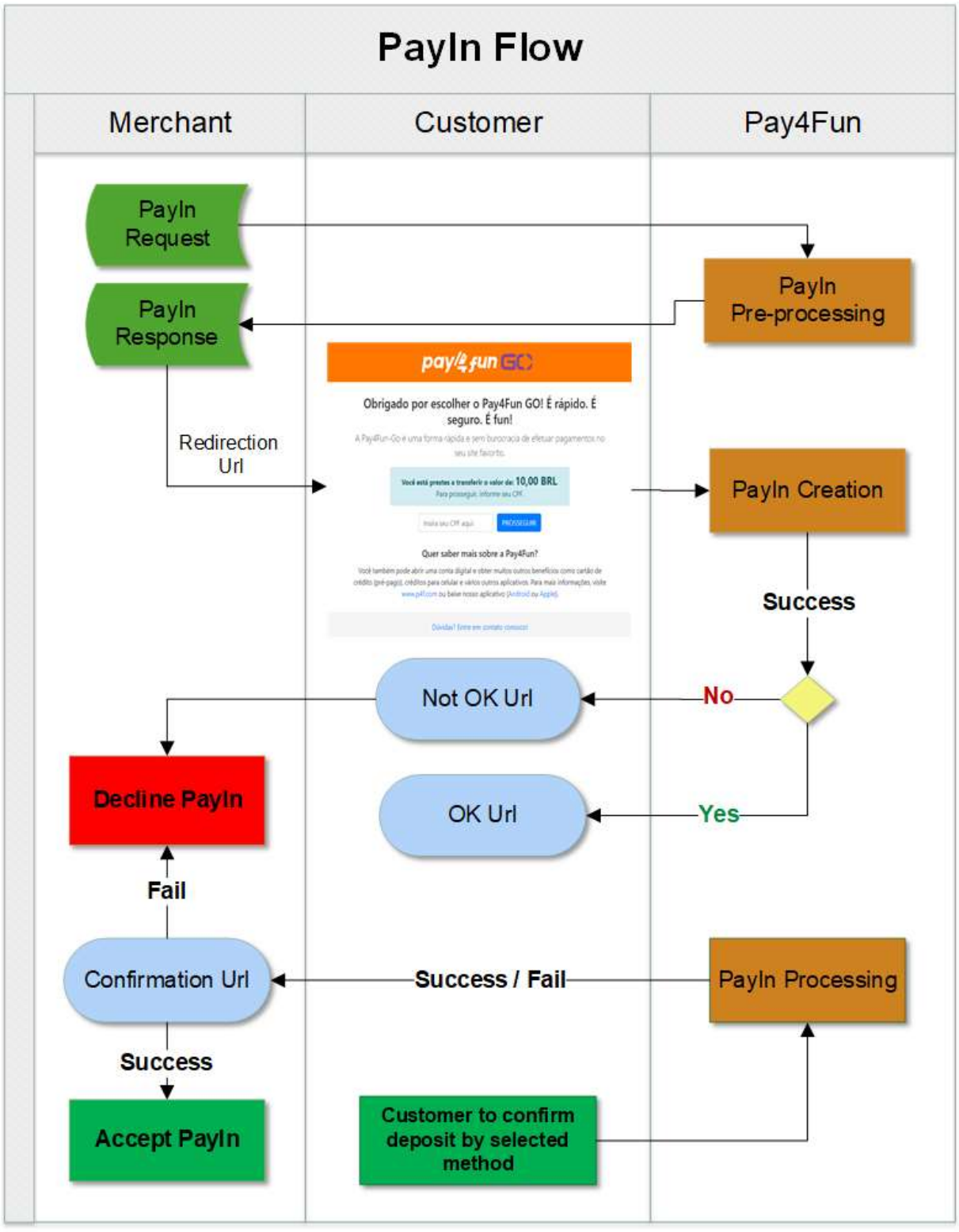
Pay In

A Pay In transaction occurs when a Customer starts a direct deposit in the Merchant website. Confirmed deposits will be credited into the Merchant wallet account.

Minimum and maximum limits are to be defined by the Merchant but must be between BRL20.00 and BRL30,000.00/day.

Maximum amount varies according to the deposit method selected. All transactions (and maximum limit) are subject to verification by Pay4Fun due to KYC and AML-CFT policies, and might be declined.

Pay In Flow Overview



Pay In Algorithm

This algorithm describes the expected steps and procedures to be followed regarding the Pay In process.

1. You send us a Pay In request
2. We analyze the request and return a **Response**
3. In case of **Response Code 400** (**fail**):
    - you analyze the Message to find out the failure motive
    - you correct what's wrong
    - you retry
4. In case of **Response Code 201** (**success**):
    - you redirect the Customer to the URL within the **Response**
    - you wait
5. Customer lands in our authentication page
6. In case of **Authentication failure**, limits exceeded, general failure or canceled by Customer:
    - we redirect Customer to your NOT_OK_URL (with the failure motive as query string)
    - we send you a **Confirmation** with **Status 400** or **500** (**fail**), meaning ultimate process failure
    - you analyze the message to find out the failure motive
    - you decline the transaction at your side
    - process finishes
7. In case of **Authentication success**:
    - Customer confirms the Pay In request parameters (amount, payment method etc.)
    - we redirect Customer to your OK_URL

- we send you a **Confirmation** with **Status 102** (**pending**), meaning Customer payment is pending
- you wait for Customer payment

8. In case of **Payment failure** (Customer payment fails or transaction expires):
    - we send you a **Confirmation** with **Status** different from 102 or 201 (**fail**), meaning ultimate process failure
    - you analyze the message to find out the failure motive
    - you decline the transaction at your side
    - process finishes
9. In case of **Payment success** (Customer performs the payment successfully):
    - we send you a **Confirmation** with **Status 201** (**success**), meaning the process was successfully completed
    - you verify the Amount within the **Confirmation** (effectively paid by Customer – Some payment methods allow higher or lower Amount)
    - you credit the verified Amount to the Customer's balance
    - process finishes

Pay In Flow Explained

TRANSACTION CREATION

When a transaction request is received at Pa4Fun's server, there are multiple validations carried out like valid source of request, structure of request, uniqueness of Merchant Invoice ID, etc. Once these validations are passed, a transaction is created.

SUCCESSFUL TRANSACTION

Customer fills authentication details to authorize the payment. Once the authorization is successful, money is credited to merchant's account. This transaction is a successful transaction.

FAILED TRANSACTION

If for any reason the transaction processing fails, t**he transaction is not created at Pay4Fun's system** and merchant will be notified through the NOT OK UL specified at the API's request.

PENDING TRANSACTION

The transaction is waiting for Customer’s payment.

Pay In Request

The purpose of this request is allow your Customers authenticate their Pay4Fun's wallet.

Important

No transaction will be generated due this request!!

You will receive an URL within the response and you should redirect your Customer to this URL where his authentication will take place.

The request header should contain two parameters: **merchantId** and **hash** for security and authorization matters.

Please, be careful to use the correct URL:

Sandbox (Testing):

<P4F_API>: **http://apitest.p4f.com**

Production (Live stage)

<P4F_API>: **https://api.p4f.com**

PAY IN REQUEST

PAY IN API REQUEST:

POST **/1.0/go/process/**

Header

```
{
   merchantId: 123456,
   hash: "xxxxxxxxx"
   Content-Type: "application/json"
}
```

An example of a valid Pay In request is shown below:

```
{
   "amount": 10.00,
   "merchantInvoiceId": "ABC777",
   "language": "en-US",
   "currency": "BRL",
   "okUrl": "https://merchant.com/ok_url",
   "notOkUrl": "https://merchant.com/not_ok_url",
   "confirmationUrl": "https://merchant.com/confirmation_url"
}
```

PAY IN REQUEST

REQUEST PARAMETERS SPECIFICATION

Parameter	Type	Description	Mandatory
hash	String	Hash to avoid fraud (check Appendix 1 – Generating the Hash section)	Yes
amount	Decimal	Pay in amount (2 digits after the decimal point)	Yes
merchantInvoiceId	String	Merchant's transaction **unique identifier** (maximum 250 characters)	Yes
language	String	Customer's desired language (pt-BR, en-US or es-ES)	No
currency	String	Pay in currency (ISO 4217 - 3 characters code)	Yes
okUrl	String	Merchant's redirection URL for successful transactions	Yes
notOkUrl	String	Merchant's redirection URL for unsuccessful transactions	Yes
confirmationUrl	String	Merchant's confirmation URL	Yes
labelId	Integer	If a valid label ID is informed, transaction will be tagged. Please check Appendix 3 – Merchant Labels for more details	No
Onboard Data parameters (check * below)			
zipcode	String	Customer's Brazilian postal code	No
email	String	Customer's e-mail	No
birthDate	String	Customer's birthdate (**format yyyy-MM-dd, e.g., 2000-03-30**)	No

PAY IN REQUEST

fullName	String	Customer's full name	No
p4fMainId	String	Customer's CPF	No
Payment Method parameters (check ** below)			
paymentMethod	String	Desired Payment Method Type: **PIX**, **Boleto** or **BankTransfer**	No
paymentMethodId	Integer	Check Error! Not a valid result for table. section for more details	No
Branding			
merchantLogo	String	Merchant's logo URL (https). Maximum height is 40 pixels	No
layoutColor	String	Hexadecimal color code	No

* ONBOARD PARAMETERS

To facilitate the user experience, you have the option to inform personal data from your customer.

If you decide to send, we will treat on our end and populate the necessary fields during the process. If you decide not to send, we will request the customer to fill in the fields.

We will treat only the first request of the customer. If you keep sending onwards we will disregard these data, so feel free to send it only once.

Parameters are: postal code, e-mail address, date of birth and full name.

PAY IN REQUEST

** PAYMENT METHOD PARAMETERS

When offering Pay4Fun GO on your deposit page, you have the option to present it as Pay4Fun GO (one icon) and/or one icon for each different payment method.

So when using the Pay4Fun GO icon you should NOT inform any of the Payment Method parameters in the request. We will show the available payment methods so customer can decide which one to use.

When using the icon for each payment method, use ONE of the following parameters in the request: paymentMethod or paymentMethodId. We will redirect and show the confirmation screen for the selected payment method.

PAY IN HASH SEED STRING

Concatenate the following information, in that order, to obtain the seed string for hash generation for Pay in requests (check Appendix 1 – Generating the Hash section):

- Merchant's ID
- Pay in amount in cents (comma or decimal point **with two decimal places**)
- Merchant's invoice ID
- Merchant's secret

Consider a Pay in request of **10.00** USD related to the merchant's invoice ID **ABC777**.

Assuming merchant's ID is **34567** and merchant's secret is "**abcdef**", the seed string for hash generation would be (in that order):

1. merchant's ID;

2. transaction’s amount with cents;
3. merchant’s invoice ID;
4. merchant’s secret;

Seed string = **34567**1000**ABC777**abcdef

The Merchant’s secret should be kept safe because is the secret factor that prevents others to generate valid hash.

Important

Even in cases of **flat amounts**, for instance, 1000.00, **cents must be included at the seed string**.

The P4F API will generate a hash for each request and compare to the Merchant’s hash. Any attempt of forging a request without the knowledge of the Merchant’s secret will lead to an invalid hash and the transaction will be declined.

Pay In Response

SUCCESS RESPONSE

In case of successful response you will receive a response containing an URL to redirect your customer to the P4F's authentication page.

After providing the authentication credentials - a valid CPF (Brazilian personal ID number) – the Pay in processing will start.

If the authentication went well, you should receive a **successful response** as below:

```
{
      "code": 201,
      "message": "success message",
      "url": "<P4F_API>/1.0/go/auth/<KEY>"
}
```

You must redirect your customer to the URL where the Pay in processing will occur, and the transaction performed.

RESPONSE PARAMETERS SPECIFICATION

Parameter	Type	Description
code	Integer	HTTP response code
message	String	Response message
url	String	URL to redirect Customer to be authenticated by P4F and confirm the transaction

UNSUCCESSFUL RESPONSE

An example of a **fail response**:

```
{
    "code": 400,
    "message": "error message"
}
```

If any detail prevents transaction being completed, API will redirect your customer to your Not Ok URL.

Important

Please, notice that the **motive of the failure will always be added to the end of your Not Ok URL as query string** as shown below:

http://yourNotOkUrl?motive=**invalid_request**

Pay In Confirmation

Important:

You must check the transaction effective amount defined at Confirmation's response in order to update it at your side.

In some payment methods, such as bank transfer, Customer could deposit a smaller amount (or higher) than the one requested at your side (we cannot enforce Customer's to deposit the exact requested amount). Once we receive the deposit confirmation on our side we will update the amount as the case may be.

After receiving the request P4F API will redirect the Customer to the Merchant's redirection URL according the transaction status (Ok or Not Ok). Merchant should inform the Customer accordingly regarding the transaction outcome, success or fail.

The Merchant's **confirmation URL** will receive the confirmation request with all parameters needed to validate if Pay In was successful or not.

If Pay4Fun receives a non-positive response (any response different from Status 200 OK) from you or no response at all (communication failure), a series of subsequent request retries will be triggered.

Important

Please notice that **you should ONLY respond with a 200 OK response if transaction was received and handled successfully at your side**.

PAY IN CONFIRMATION

Pay4Fun's API will send confirmation requests to your confirmation URL for a 24 hours period, with 10 minutes between requests.

If Pay4Fun's API doesn't receive a Status 200 OK until the last attempt, the retry operation will be terminated and no further attempts will occur.

Important

In case of any communication failure on receiving Pay4Fun's confirmation you should keep listening to your Confirmation URL until receiving the subsequent confirmation and, then, complete the transaction on your side.

The official transaction confirmation will only be made through the informed Merchant's Confirmation URL; the confirmation JSON will be sent through POST to this URL.

When a Pay In transaction is created by the customer, the API will send a confirmation with status **102** (Pending).

After the customer payment confirmation, the API will send a confirmation with status **201** (Successful).

Customer has a certain amount of time to confirm the payment (it varies according to the selected deposit method). If customer does not confirm payment within the time frame, the transaction expires and the API will send a confirmation with status **Failed**.

PAY IN CONFIRMATION

The response you will receive at your Confirmation URL looks like the following:


```
{
    "TransactionId": 98765,
    "Amount": 10.00,
    "FeeAmount": 0.00,
    "MerchantInvoiceId": "ABC777",
    "Currency": "USD",
    "Status": "201",
    "LiquidationDate": "2019-01-15",
    "Message": "success",
    "CustomerEmail": "customer@email.com",
    "Sign": "0CE3325211878F6A9131252128EEAA1EB0398B594",
    "PaymentMethod": "Pix"
}
```


RESPONSE PARAMETERS SPECIFICATION

Parameter	Type	Description
TransactionId	String	P4F's transaction unique identifier
Amount	Decimal	Transaction amount
FeeAmount	Decimal	Transaction's fee amount
merchantInvoiceId	String	Merchant's transaction **unique identifier**
Currency		Pay in currency (ISO 4217 - 3 characters code)

PAY IN CONFIRMATION

Status	String	Status 201 denotes transaction success. Status 102 denotes PENDING transaction Any other status denotes failed transaction
LiquidationDate	String	Date the transaction's amount will be transferred to Merchant's Available Balance.
Message	String	Success or fail message
CustomerEmail	String	E-mail that has originated the transaction. Use it as additional information on compliance processes
Sign	String	Contains details of the response hashed according **Pay in Sign** below
PaymentMethod	String	Informs payment method type used by Customer (boleto, PIX, bank transfer, etc.)

The response contains a hashed parameter to provide you way to confirm that information hasn't been tampered: the parameter **Sign.**

PAY IN SIGN HASH SEED STRING

The parameter **Sign**, allows you to confirm the veracity of all relevant information contained within the response. The Sign hash should be used to confirm the information has not be tampered.

Considering a Transaction amount of **10.00** USD related to the merchant's invoice ID **ABC777** and assuming merchant's ID is **34567** and the response was successful (response's Status parameter equals to 201), the seed string for hash generation would be (in that order):

1. merchant's ID;
2. transaction's amount with cents;
3. merchant's invoice ID;
4. response's Status parameter

Seed string = **345671000ABC777201**

Warning

These parameters' seed strings **are different from the hash seed for API Pay in original request.** Please, be aware of that!

Check Appendix 1 – Generating the Hash section for more details.

Pay In Query Status

Important

We strongly recommend:

Do not use this feature before you have received the automatic confirmation request from this API. Otherwise, you may receive an inaccurate intermediary status while the transaction is in processing. The response time for the automatic confirmation request may vary depending on the payment method selected.

Sending just **one batch with several Invoice ids** instead of sending several batches, all at the same time, with just one Invoice id each to avoid blockages!

This API endpoint allows the Merchant to retrieve the status of a list of Pay In Merchant's invoice ids.

The request header should contain two parameters: **merchantId** and **hash** for security and authorization matters.

```
{
    merchantId: 123456,
    hash: "xxxxxxxxx"
    Content-Type: "application/json"
}
```

PAY IN QUERY STATUS

PAY IN QUERY STATUS REQUEST:

POST **/1.0/go/transaction/**

Body (list of Merchant's Invoice IDs - Pay In unique identifiers)

```
[
    "ABC777", "ABD778"
]
```

REQUEST PARAMETERS SPECIFICATION

Parameter	Type	Description	Mandatory
merchantId	String	Merchant unique ID	yes
hash	String	Request hash to avoid fraud (check Appendix 1 – Generating the Hash section)	Yes
merchantInvoiceId	String	List of Merchant's Pay In **unique identifiers**	Yes

Concatenate the following information, in that order, to obtain the seed string for hash generation for Pay in Query Status requests (check Appendix 1 – Generating the Hash section):

- Merchant's ID
- Merchant's secret

Assuming merchant's Id is **34567** and merchant's secret is "**abcdef**", the seed string for hash generation would be:

Seed string = **34567abcdef**

Important

The Merchant's secret should be kept safe because is the secret factor that prevents others to generate valid hash.

PAY IN QUERY STATUS RESPONSE:

Notice the response contains information regarding any Merchant Invoice ID informed, even if it hasn't been found! In those cases, the Status will be **NotFound** which means there's no transaction related to the specific Merchant Invoice ID.


```
[
  {
     "transactionId": 31,
     "merchantInvoiceId": "ABC777",
     "status": "Verified",
     "sign": "878F6A3891313521128EE1EB03984A1A9B94EAE9F240",
     "amount": 222.60,
     "customerEmail": "customer@email.com",
     "liquidationDate": "2019-02-25"
   },
   {
     "transactionId": 0,
     "merchantInvoiceId": "ABD778",
     "status": "NotFound",
     "sign": "878F6A389131E58AE25210CE335A1A9BCB594EAE9F2",
     "amount": 0,
     "customerEmail": null,
     "liquidationDate": null
   }
]
```

PAY IN QUERY STATUS

The response for the Pay in Query Status request is like below:

RESPONSE PARAMETERS SPECIFICATION

Parameter	Type	Description
transactionId	String	Pay4Fun's transaction ID (zero value means the process has failed in pre-authentication steps and no transaction was created in Pay4Fun's system; you should consider your transaction as failed)
merchantInvoiceId	String	Merchant's Transaction ID
status	String	Pay in transaction status
sign	String	Hash sign to allow veracity confirmation
amount	Decimal	Transaction amount
customerEmail	String	Customer's e-mail
LiquidationDate	String	Date the transaction's amount will be transfer to Merchant's available balance.

Regarding the pay in transaction's statuses you should consider the following:

- **PENDING**: transaction is waiting for Customer's payment;
- **LIQUIDATION**: transaction was successfully processed and the amount will be transferred to Merchant's Available Balance at liquidation date;
- **VERIFIED** or **RELEASED**: transaction was successfully processed and the amount transferred to Merchant's Available Balance;
- **DECLINED**: transaction was irreversible declined;

The sign parameter contained within the response is built as the following description:

Consider a Transaction amount of **10.00** USD related to the merchant's invoice ID **ABC777**.

Assuming merchant's ID is **34567** and transaction status is **Verified**, the seed string for hash generation would be (in that order):

1. merchant's ID;
2. merchant's invoice ID ;
3. transaction's amount with cents;
4. transaction's status;

Seed string = **34567ABC7771000Verified**

Important

Only Transactions with **Verified, Released** or **Liquidation** statuses should be considered as successfully processed. Any other status indicates unsuccessful transaction.

Check Appendix 1 – Generating the Hash section for more details.

Payment Methods

Retrieve a list with all available payment methods for P4F GO API.

PAYMENT METHODS API REQUEST:

GET **/1.0/go/paymentMethods/**

Header

```
{
   merchantId: 123456,
   hash: "xxxxxxxxx"
   Content-Type: "application/json"
}
```

An example of a valid PaymentMethods request is shown below:

```
[{
   "paymentMethodId": 1987,
   "name": "Itau",
   "paymentMethodTypeName": "Bank Transfer"
},
{
   "paymentMethodId": 2009,
   "name": "PIX",
   "paymentMethodTypeName": "Bank Transfer"
}, …
]
```

PAYMENT METHOD HASH SEED STRING

Concatenate the following information, in that order, to obtain the seed string for hash generation for Payment methods requests (check Appendix 1 – Generating the Hash section):

- Merchant's ID
- Merchant's secret

Assuming merchant's ID is **34567** and merchant's secret is "**abcdef**", the seed string for hash generation would be:

Seed string = **34567abcdef**

PayOut

A Pay Out transaction occurs when a Customer requests a withdrawal in the Merchant website. Confirmed withdrawals will be debited from the Merchant wallet account.

Minimum and maximum limits are to be defined by the Merchant but must be between BRL20.00 and BRL20,000.00/day. All transactions (and maximum limit) are subject to verification by Pay4Fun due to KYC and AML-CFT policies, and might be declined.

This function is very useful to pay Customers withdrawals or to refund a Pay In.

Pay Out process has two distinctive parts:

SCREENING

Merchant sends all the Pay Out data, Pay4Fun then validates CPF, full name, telephone, date of birth and e-mail. If “successful” Screening is complete and Pay Out is ready to be confirmed by the Merchant if/when suitable. If “not successful” Merchant must analyze the reason and decide to cancel or edit the request (provide a way for Customer to update the data correctly). If reason for “not successful” is related to transaction limits, there is nothing the Merchant can do and should advise Customer to contact Pay4Fun. During the Screening process no data or transaction is registered on Pay4Fun’s side, not even Merchant Invoice ID.

PAY OUT (CONFIRMATION)

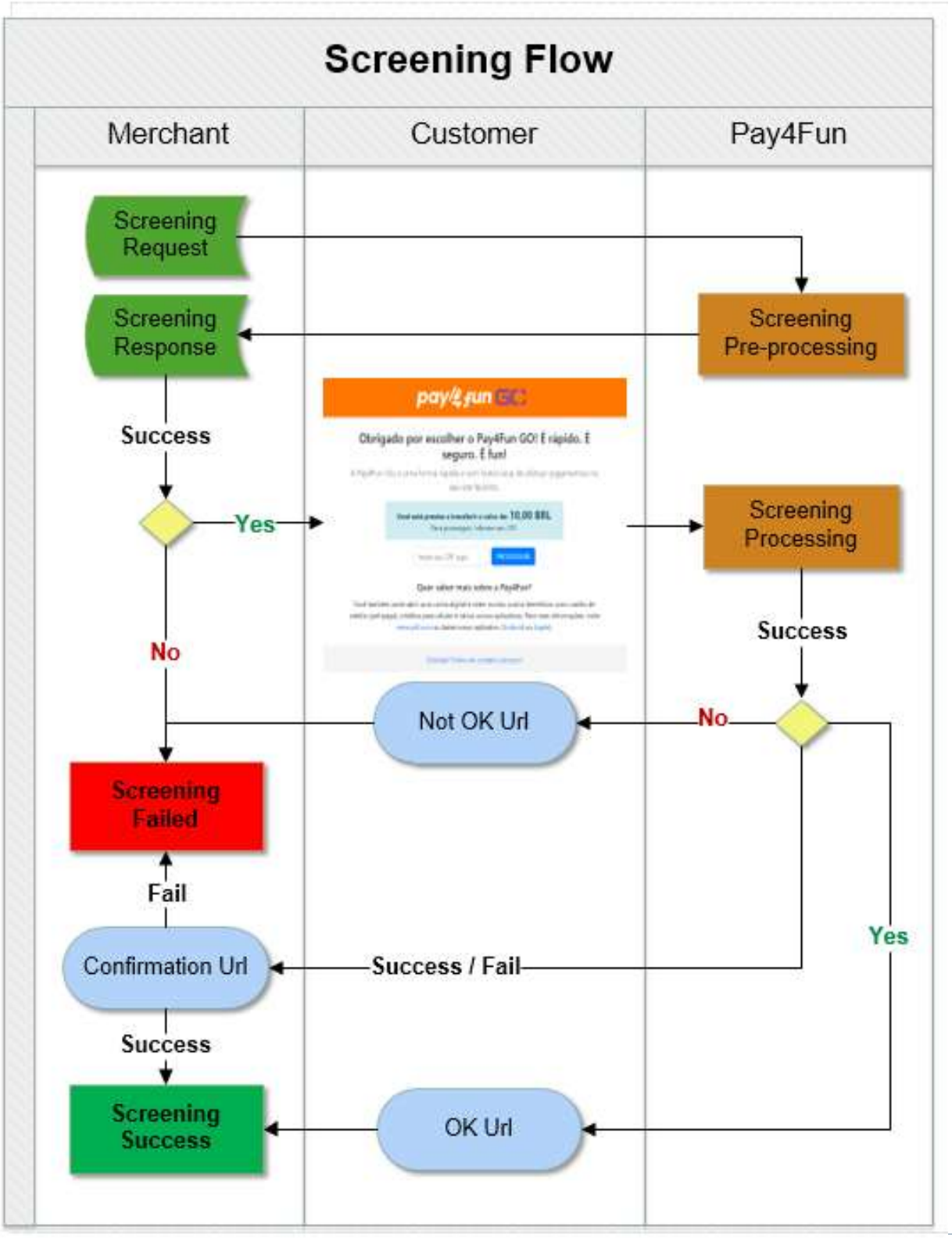
When Merchant confirms the PayOut (when the Finance Team releases it for payment, for example), Pay4Fun validates the request again to confirm that Customer's data and limits are Ok. If not, Merchant will receive a "not successful" with correspondent error message. If Ok, the expected status is "pending" meaning the transaction is pending processing by the Pay4Fun Finance Team or a third party. Once the transaction is processed, the Merchant will receive a confirmation or a "fail message".

So, Merchant should expect three confirmations from the API:

1. Screening confirmation: successful or not successful;
2. Pay Out confirmation: pending or not successful;
3. Processing confirmation: successful (process is closed) or not successful (transaction amount and fee will be credit back into the Merchant wallet account, as the case may be).

In order to test both process, please, check the **Appendix 5 – PayOut Test Data**.

Screening Process Flow



PayOut Screening Algorithm

This algorithm describes the expected steps and procedures to be followed regarding the PayOut Screening process.

1. You send us a PayOut Screening request
2. We analyze the request and return a **Response**
3. In case of **Response Code 400** (**fail**):
    - you analyze the Message to find out the failure motive
    - you correct what's wrong
    - you retry
4. In case of **Response Code 200** (**success**):
    - you redirect the Customer to the URL within the **Response**
    - you wait
5. Customer lands in our screening page
6. In case of **Screening failure**, limits exceeded, general failure or canceled by Customer:
    - we redirect Customer to your NOT_OK_URL (with the failure motive as query string)
    - we send you a **Confirmation** with **Status 400** or **500** (**fail**), meaning ultimate process failure
    - you analyze the message to find out the failure motive
    - process finishes
7. In case of **Screening success**:
    - we validate Customer Screening data
    - we redirect Customer to your OK_URL
    - we send you a **Confirmation** with **Status 102** (**pending**), meaning the PayOut processing is in standby waiting for your PayOut Processing request
    - process (Screening) finishes

PayOut Screening Request

This API endpoint allows merchants to start the Screening process that will validate Customer's data.

PayOut transaction won't be created at this time. In case the Screening process fails you should always analyze the failure motive when Customers are redirected to your NotOK URL.

Important

Please, notice that the **motive of the failure will always be added to the end of your Not Ok URL as query string** as shown below:

http://yourNotOkUrl?motive=**maximum_amount_exceeded**

Please, be careful to use the correct API url:

Sandbox (Testing):

<P4F_API>: **http://apitest.p4f.com**

Production (Live stage)

<P4F_API>: **https://api.p4f.com**

PAYOUT SCREENING REQUEST

Step 1 - PayOut Screening:

POST **/1.0/goout/screening/**

An example of a valid JSON request (valid format only) would be:

```
{
      "amount": 10.00,
      "merchantInvoiceId": "ABC777",
      "language": "pt-BR",
      "currency": "BRL",
      "okUrl": "https://url.com/ok_url",
      "notOkUrl": "https://url.com/not_ok_url",
      "confirmationUrl": "https://url.com/confirmation_url"
      "merchantId": 0,
      "hash": "878F6A389131E58AE25210CE335A1A9BCB594EAE9F2",
      "targetCustomerMainId": "54365432123",
      "targetCustomerEmail": "johndoe@example.com",
      "fullName": "John Doe",
      "targetCustomerBirthDate": "1990-05-31",
      "targetCustomerPhoneNumberCountryCode": "55",
      "targetCustomerPhoneNumberAreaCode": "11",
      "targetCustomerPhoneNumber": "987654321"
}
```

PAYOUT SCREENING REQUEST

REQUEST PARAMETERS SPECIFICATION

Parameter	Type	Description	Mandatory
amount	Decimal	Payout amount (2 digits after the decimal point)	Yes
merchantInvoiceId	String	Merchant's transaction **unique identifier** (maximum 250 characters)	yes
language	String	Customer's desired language (pt-BR, en-US or es-ES)	No
currency	String	Payout currency (ISO 4217 - 3 characters code)	Yes
okUrl	String	Merchant's redirection URL for successful transactions	Yes
notOkUrl	String	Merchant's redirection URL for unsuccessful transactions	Yes
confirmationUrl	String	Merchant's confirmation URL	Yes
merchantId	Integer	Merchant's unique identifier at Pay4Fun	Yes
hash	String	Request hash to avoid fraud (check Appendix 1 – Generating the Hash section)	Yes
targetCustomerMainId	String	Locks the transaction to a specific Customer's Main Id	Yes
targetCustomerEmail	String	Specify the target Customer e-mail	Yes

PAYOUT SCREENING REQUEST

fullName	String	Customer's full name	Yes
targetCustomerBirthDate	String	Customer's birth date (format yyyy-mm-dd)	Yes
targetCustomerPhoneNumberCountryCode	String	Customer's phone country code	Yes
targetCustomerPhoneNumberAreaCode	String	Customer's phone area code	Yes
targetCustomerPhoneNumber	String	Customer's mobile phone number	Yes
		Branding	
merchantLogo	String	Merchant's logo URL (https). Maximum height is 40 pixels	No
layoutColor	String	Hexadecimal color code	No

PAYOUT HASH SEED STRING

The parameter **Hash** allows both sides, Pay4Fun and Merchants to confirm the veracity of all relevant information contained within the requests or responses.

The Hash will be used to confirm the information has not be tampered. Considering a Payout amount of **10.00 USD** related to the Customer email **customer@email.com**.

Assuming merchant's ID is **34567**, merchant Invoice ID is 12345 and merchant's secret is "**abcdef**", the seed string for sign hash generation would be (in that order):

1. merchant's ID;
2. payout amount with cents (without comma or decimal point);
3. merchant invoice ID;
4. target customer's email;
5. merchant's secret;

Seed string = **34567100012345customer@email.comabcdef**

Check Appendix 1 – Generating the Hash section for more details.

PayOut Screening Response

SUCCESSFUL RESPONSE

In case of successful screening you will receive a response containing an URL to redirect your customer to the P4F's authentication page.

```
{
      "code": 200,
      "message": "pending",
      "url": "<P4F_API>/1.0/goout/screening/<KEY>"
}
```

You must redirect your customer to the URL where the PayOut processing will continue.

After providing the authentication credentials the PayOut Screening will finish.

RESPONSE PARAMETERS SPECIFICATION

Parameter	Type	Description
code	Integer	HTTP response code
message	String	Response message
url	String	URL to redirect Customer to be authenticated by P4F and confirm the transaction

PAYOUT SCREENING RESPONSE

UNSUCCESSFUL RESPONSE

An example of a **fail response**:

```
{
    "code": 400,
    "message": "error message"
}
```

If any detail prevents the process being completed, the API will redirect your customer to your Not Ok URL.

Important

Please, notice that the **motive of the failure will always be added to the end of your Not Ok URL as query string** as shown below:

http://yourNotOkUrl?motive=**maximum_amount_exceeded**

PayOut Screening Confirmation

A confirmation will be sent to your Confirmation-URL with the final result for the PayOut Screening process.

Parameters **Content-Type: application/json**


```
{
      "TransactionId": 0,
      "MerchantInvoiceId": "1632839298187",
      "Currency": "USD",
      "Amount": 10.00,
      "ConvertedCurrency": "BRL",
      "ConvertedAmount": 54.67,
      "FeeAmount": 0.0,
      "Status": 102,
      "Message": "pending",
      "CustomerEmail": "customer@email.com",
      "Hash": "878F6A389131E58AE25210CE335A1A9BCB594EAE9F2",
      "Process": "Screening"
}
```


In case of failure the motive should be analyzed and, if possible, correct to restart the process.

In case of success, you will be able to perform the PayOut request.

PAYOUT CONFIRMATION HASH SEED STRING

The parameter **Hash**, allows you to confirm the veracity of all relevant information contained within the response. The Sign hash should be used to confirm the information has not be tampered.

Considering a Transaction amount of **10.00** USD related to the merchant's invoice ID **ABC777** and assuming merchant's ID is **34567** and the target customer e-mail is "customer@email.com", the seed string for hash generation would be (in that order):

1. merchant's ID;
2. transaction's amount with cents;
3. merchant's invoice ID;
4. target customer e-mail

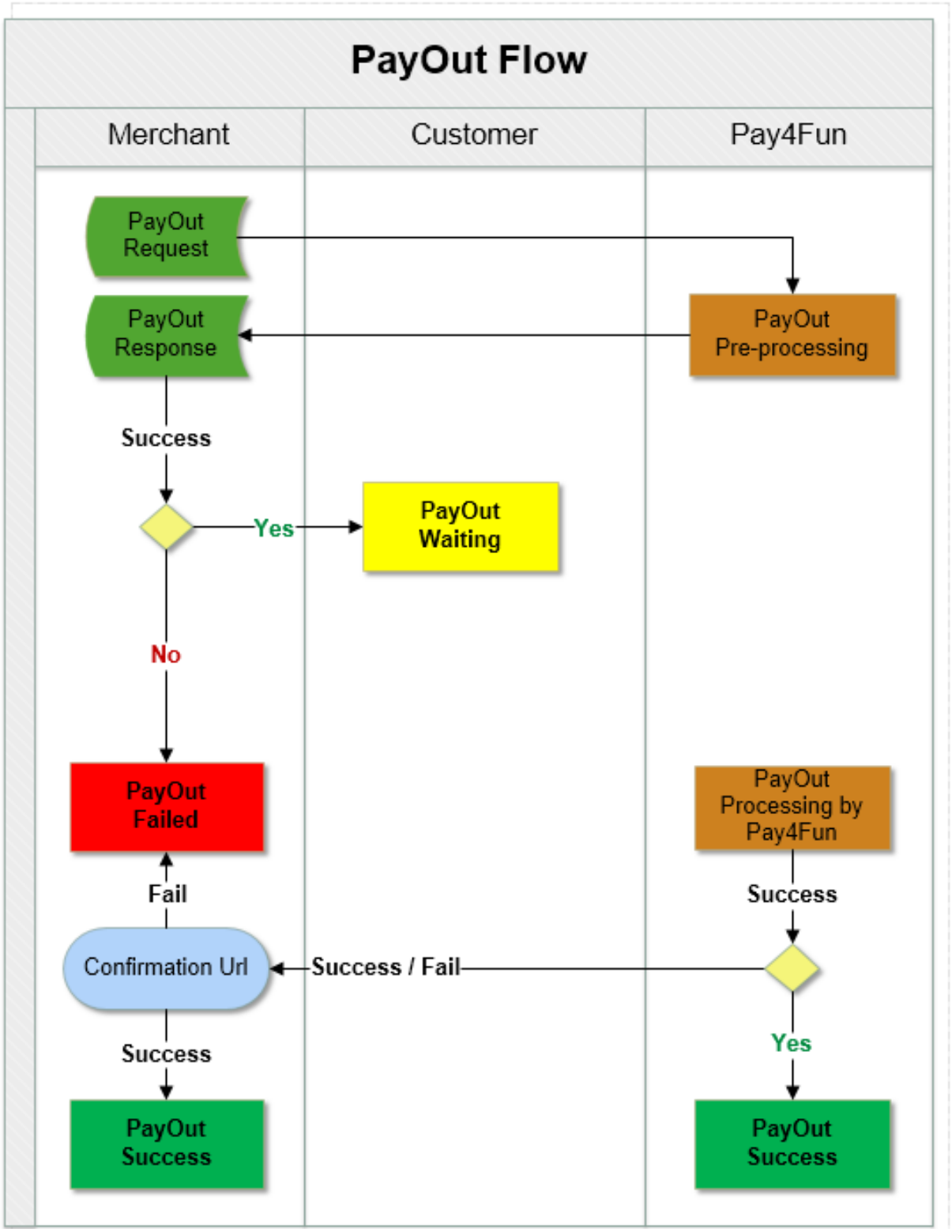
Seed string = **34567****1000****ABC777****customer@email.com**

> **Warning**
>
> These parameters' seed strings **are different from the hash seed for API PayOut original request.** Please, be aware of that!

The same seed string should be used for all PayOut Confirmation requests (Screening and PayOut).

Check Appendix 1 – Generating the Hash section for more details.

PayOut Process Flow



PayOut Process Algorithm

This algorithm describes the expected steps and procedures to be followed regarding the PayOut process.

1. You send us a PayOut Process request
2. We analyze the request and return a **Response**
3. In case of **Response Code 400** (**fail**):
    - you analyze the Message to find out the failure motive
    - you correct what's wrong
    - you retry
4. In case of **Response Code 102** (**pending**):
    - We will process the PayOut
    - you wait for PayOut **Confirmation**
5. In case of **Payment expiration**, general failure, etc.:
    - we send you a **Confirmation** with **Status 400** or **500** (**fail**), meaning ultimate process failure
    - you analyze the message to find out the failure motive
    - you decline the transaction at your side
    - process finishes
6. In case of **Payment success**:
    - we transfer the PayOut Amount to Customer’s bank account
    - we send you a **Confirmation** with **Status 201** (**success**), meaning PayOut was successfully processed
    - you process the PayOut at your side accordingly
    - process finishes

PayOut Processing Request

This step requires that the previous step, PayOut Screening, has been performed successfully.

PayOut processing requests occurs when the Merchant Finance Team releases it for payment and the PayOut transaction only will be created after a series of validations. **In case any of them are not satisfied, the PayOut process will fail and no Transactions will be created at Merchant's back-office**; you should always analyze the failure motive sent to your Confirmation-URL.

PAYOUT PROCESSING REQUEST:

POST **/1.0/goout/process/**

PAYOUT PROCESSING REQUEST

A valid JSON request would be:

```
{
      "amount": 10.00,
      "merchantInvoiceId": "ABC777",
      "language": "pt-BR",
      "currency": "BRL",
      "confirmationUrl": "https://url.com/confirmation_url",
      "merchantId": 0,
      "labelId": 1,
      "hash": "878F6A389131E58AE25210CE335A1A9BCB594EAE9F2",
      "targetCustomerEmail": "johndoe@example.com",
      "targetCustomerMainId": "11111111111",
      "pixKeyType": "CPF",
      "bankCode": "15",
      "bankBranch": "12345",
      "bankAccount": "678-9",
      "bankAccountType": "personal"
}
```

REQUEST PARAMETERS SPECIFICATION

Parameter	Type	Description	Mandatory
amount	Decimal	Payout amount (2 digits after the decimal point)	Yes
merchantInvoiceId	String	Merchant's transaction **unique identifier** (maximum 250 chars)	Yes
language	String	Customer's desired language (pt-BR, en-US or es-ES)	No

currency	String	Payout currency (ISO 4217 - 3 characters code)	Yes
confirmationUrl	String	Merchant's confirmation URL	Yes
merchantId	Integer	Merchant's unique identifier at Pay4Fun	Yes
hash	String	Request hash to avoid fraud (check Appendix 1 – Generating the Hash section)	Yes
targetCustomerMainId	String	If informed, locks the transaction to a specific Customer's Main Id	Yes
targetCustomerEmail	String	Specify the target Customer e-mail	Yes
pixKeyType	String	Pix key type to be performed: **CPF**, **Email**, **Telefone** or **DadosBancarios**	Yes
bankCode	String	Bank's Code	Yes/No*
bankBranch	String	Customer's bank branch	Yes/No*
bankAccount	String	Customer's bank account	Yes/No*
bankAccountType	String	Customer's bank account type: **SAVING** or **PERSONAL**	Yes/No*

*If **PixKeyType** is "CPF", "Email" or "Telefone" then these parameters are not mandatory. If **PixKeyType** is "DadosBancarios", then these parameters are mandatory.

PayOut Processing Response

SUCCESSFUL RESPONSE

In case of success, you will receive a **successful response** as below:


```
{
      "TransactionId": 0,
      "MerchantInvoiceId": "1632839298187",
      "Currency": "USD",
      "Amount": 10.00,
      "ConvertedCurrency": "BRL",
      "ConvertedAmount": 54.67,
      "FeeAmount": 0.0,
      "Status": 102,
      "Message": "pending",
      "CustomerEmail": "customer@email.com",
      "Hash": "878F6A389131E58AE25210CE335A1A9BCB594EAE9F2",
      "Process": "PayOut"
}
```


If so, PayOut will be processed and you will receive a final confirmation as soon as it is done.

UNSUCCESSFUL RESPONSES

In case of failure, you will receive an **unsuccessful response** as below:

```
{
      "code": 400,
      "message": "error message"
}
```

If two requests are made within an interval of less than 2 minutes, the API will return the following fail response:

```
{
      "code": 409,
      "message": "transaction_in_processing"
}
```

In these cases, PayOut will not be processed, and you will not receive any further confirmation.

Warning

Customers are able to interrupt the PayOut process any time during the authentication or PayOut payment. In that case, the error message will be "**declined_by_customer**".

PayOut Processing Confirmation

The response will be sent to the Confirmation URL specified at the PayOut request and contains the following parameters:

SUCCESSFUL CONFIRMATION RESPONSE

In case of success, you will receive a **successful confirmation response** as below:


```
{
      "TransactionId": 7904320,
      "MerchantInvoiceId": "1632839298187",
      "Currency": "USD",
      "Amount": 10.00,
      "ConvertedCurrency": "BRL",
      "ConvertedAmount": 54.67,
      "FeeAmount": 0.0,
      "Status": 201,
      "Message": "success",
      "CustomerEmail": "customer@email.com",
      "Hash": "878F6A389131E58AE25210CE335A1A9BCB594EAE9F2",
      "Process": "Payout"
}
```

UNSUCCESSFUL CONFIRMATION RESPONSE

In case of fail, you will receive an un**successful confirmation response** as below:


```
{
      "TransactionId": 7904320,
      "MerchantInvoiceId": "1632839298187",
      "Currency": "USD",
      "Amount": 10.00,
      "ConvertedCurrency": "BRL",
      "ConvertedAmount": 54.67,
      "FeeAmount": 0.0,
      "Status": "failed",
      "Message": "failed",
      "CustomerEmail": "customer@email.com",
      "Hash": "878F6A389131E58AE25210CE335A1A9BCB594EAE9F2",
      "Process": "Payout"
}
```

PayOut Query Status

Important

We strongly recommend:

Do not use this feature before you have received the automatic confirmation request from this API. Otherwise, you may receive an inaccurate intermediary status while the transaction is in processing. The response time for the automatic confirmation request may vary depending on the payment method selected.

Sending just **one batch with several Invoice ids** instead of sending several batches, all at the same time, with just one Invoice id each to avoid blockages!

This API endpoint allows the Merchant to retrieve the status of a list of PayOuts by Merchant's invoice ids.

The request header should contain two parameters: **merchantId** and **hash** for security and authorization matters.

```
{
    merchantId: 123456,
    hash: "xxxxxxxxx"
    Content-Type: "application/json"
}
```

PAYOUT QUERY STATUS

PAYOUT QUERY STATUS REQUEST:

POST **/1.0/goout/transaction/**

Body (list of Merchant's Invoice IDs - PayOut unique identifiers)

```
[
    "ABC777", "ABD778"
]
```

REQUEST PARAMETERS SPECIFICATION

Parameter	Type	Description	Mandatory
merchantId	String	Merchant unique ID	yes
hash	String	Request hash to avoid fraud (check Appendix 1 – Generating the Hash section)	Yes
merchantInvoiceId	String	List of Merchant's PayOut **unique identifiers**	Yes

Concatenate the following information, in that order, to obtain the seed string for hash generation for PayOut Query Status requests (check Appendix 1 – Generating the Hash section):

- Merchant's ID
- Merchant's secret

Assuming merchant's Id is **34567** and merchant's secret is "**abcdef**", the seed string for hash generation would be:

Seed string = **34567abcdef**

Important

The Merchant's secret should be kept safe because is the secret factor that prevents others to generate valid hash.

PAYOUT QUERY STATUS RESPONSE:

Notice that the response will contain information regarding any Merchant Invoice ID present in the request, even if this ID hasn't been found! In those cases, the Status will be **NotFound** which means there's no transaction related to that specific Merchant Invoice ID.

The response for the PayOut Query Status request is like below:


```
[
  {
    "transactionId": 35441,
    "merchantInvoiceId": "ABC777",
    "status": "Verified",
    "sign": "878F6A38913135211 28EE1EB03984A1A9B94EAE9F240",
    "amount": 222.60,
    "customerEmail": "customer@email.com",
    "liquidationDate": "2019-02-25"
  },
  {
    "transactionId": 0,
    "merchantInvoiceId": "ABD778",
    "status": "NotFound",
    "sign": "878F6A389131E58AE25210CE335A1A9BCB594EAE9F2",
    "amount": 0,
    "customerEmail": null,
    "liquidationDate": null
  }
]
```

PAYOUT QUERY STATUS

RESPONSE PARAMETERS SPECIFICATION

Parameter	Type	Description
transactionId	String	Pay4Fun Transaction ID (A value of zero means the process has failed in pre-authentication steps and no transaction was created in the Pay4Fun system; you should consider your transaction as failed)
merchantInvoiceId	String	Merchant's Transaction ID
status	String	PayOut transaction status
sign	String	Hash sign to allow veracity confirmation
amount	Decimal	Transaction amount
customerEmail	String	Customer's e-mail
LiquidationDate	String	Date the transaction's amount was effectively transferred from Merchant's available balance.

Regarding the PayOut transaction's statuses you should consider the following:

- **VERIFIED**: transaction was successfully processed and the amount transferred to Merchant's Available Balance;
- **DECLINED**: transaction was irreversible declined;

The sign parameter contained within the response is built as the following description:

Consider a Transaction amount of **10.00** USD related to the merchant's invoice ID **ABC777**.

Assuming merchant's ID is **34567** and transaction status is **Verified**, the seed string for hash generation would be (in that order):

5. merchant's ID;
6. merchant's invoice ID ;
7. transaction's amount with cents;
8. transaction's status;

Seed string = **34567ABC7771000Verified**

Important

Only Transactions with **Verified** statuses should be considered as successfully processed. Any other status indicates unsuccessful transaction.

Check Appendix 1 – Generating the Hash section for more details.

Bank List

Retrieve a list with all available banks for P4F GO OUT API.

BANK LIST API REQUEST:

GET **/1.0/goout/banklist/**

Header

```
{
   merchantId: 123456,
   hash: "xxxxxxxxx"
   Content-Type: "application/json"
}
```

An example of a PaymentMethods response is shown below:

```
[
  {
     "bankCode": "001",
     "bankName": "Banco do Brasil"
  },
  {
     "bankCode": "003",
     "bankName": "Banco Santander"
  }, …
]
```

BANK LIST

BANK LIST METHOD HASH SEED STRING

Concatenate the following information, in that order, to obtain the seed string for hash generation for Bank List requests (check Appendix 1 – Generating the Hash section):

- Merchant's ID
- Merchant's secret

Assuming merchant's ID is **34567** and merchant's secret is "**abcdef**", the seed string for hash generation would be:

Seed string = **34567abcdef**

Appendix 1 – Generating the Hash or Sign

Keeping the communication safe is our top priority, therefore, to avoid any impersonation attempt, all request should contain a **Hash-based Message Authentication Code** (HMAC) by using the SHA256 hash function.

Specific information that are part of the request should be encrypted together with the Merchant's secret to ensure that all information received by P4F is exactly the same sent by the Merchant. You will be using your Merchant Key as encryption key for this purpose.

Keep in mind that each request type demands a particular seed string, formed by fragments of the own request, to generate the hash and any request containing incorrect hash will be declined; the error message you will receive will be: **invalid_security_hash**

The **Appendix 1 – Generating the Hash or Sign** contains a guideline on generating hashes coded in C sharp.

Please, for an on-line tool for generating hashes access the URL below:

https://dotnetfiddle.net/bYQfgP

Using this online tool you will be able to generate your hashes for testing purposes and confirm you are generating hashes accordingly.

APPENDIX 1 – GENERATING THE HASH OR SIGN

The method below (C Sharp) should be used as a guideline to generate all required API hashes.

The encryption key to be used is your **Merchant Key**.

```
public string HMACSHA256(string toBeEncripted, string
merchantKey)
{
     byte[] key = Encoding.UTF8.GetBytes(merchantKey);
     using (HMACSHA256 hmac = new HMACSHA256(key))
     {
          hmac.Initialize();
          byte[] bytes_hmac_in =
          Encoding.UTF8.GetBytes(toBeEncripted);

          byte[] bytes_hamc_out =
          hmac.ComputeHash(bytes_hmac_in);

          string str_hamc_out =
          BitConverter.ToString(bytes_hamc_out);

          str_hamc_out = str_hamc_out.Replace("-", "");

          return str_hamc_out;
     }
}
```

Appendix 2 – Error Messages

Above listed are the possible error messages sent by Pay4Fun in response to your API's request:

ERROR MESSAGES

Error message	Description
customer_account_has_been_locked	In case Customer already have a Pay4Fun account and it is blocked
customer_account_must_be_approved	In case Customer already have a Pay4Fun account and it is not Approved
customer_money_in_limit_for_payment_method_type_reached	Customer limit reached. Please inform Customer to contact Pay4Fun to improve his/hers personal profile
customer_not_authorized	Customer suspended or blocked thus not able to perform transactions
declined_by_customer	Transaction was declined intentionally by the Customer.
err_blocked_by_response	You may receive this error if you try load Pay4Fun's URL inside an Iframe.
external_invoice_id_already_used	There is a previous Transaction using the specified Merchant's Invoice ID.
external_invoice_id_already_used	Merchant Invoice ID already used
insufficient_balance	Merchant's available Balance is not sufficient to perform the Transaction

APPENDIX 2 – ERROR MESSAGES

invalid_bank_account_data	Invalid Customer's bank account data
invalid_customer_main_id	The target customer has a different Main ID that the value specified within the request.
invalid_email	Invalid Customer's e-mail
invalid_request	The requested sent by Merchant is not valid.
invalid_request_currency	The Currency sent within the request is not valid.
invalid_security_hash	Something is wrong with the Hash sent within the request. Check the instructions on how generate the Hash correctly.
maximum_amount_exceeded	The maximum allowed transaction amount was exceeded
merchant_account_not_live	The Merchant account is not yet Live.
merchant_ip_not_authorized	The IP used to perform API request is not whitelisted.
merchant_not_authorized	The Merchant's account is not enabled.
merchant_not_found	The Merchant ID sent within the request is not valid.
minimum_amunt_not_reached	The minimum allowed transaction amount was not reached
money_out_limit_reached	Customer limit reached. Please inform Customer to contact Pay4Fun to improve his/hers personal profile
processing_error	Something very bad has happened while processing Merchant's request.

Appendix 3 – Merchant Labels

Merchant labels are a way to tag transactions (pay in or payout) as originated from a specific merchant's brand or website.

Accessing the merchant's back-office, through the menu **Account / Api / Labels** is possible to manage your labels (create or rename).

Labels have only two properties: ID and Name. The Name has its length limited to 12 characters

In order to label a transaction as originated from the specific brand or website, Merchant should use a valid Label ID to fill up the Api request parameter LabelId.

Merchant's reports will show labeled transaction's label in the reports results.

Keep in mind that transactions will only be labeled with a specific label if the request parameter **LabelId** is filled with a valid Label ID otherwise the transaction will remain untagged.

Appendix 4 – Security issues using iframes

The use of iframes is not allowed, because your site would become vulnerable to cross-site attacks.

The use of iframes brings a sort of security risks:

1- You may get a submittable malicious web form, phishing your users' personal data.
2- A malicious user can run a plug-in.
3- A malicious user can change the source site URL.
4- A malicious user can hijack your users' clicks.
5- A malicious user can hijack your users' keystrokes.
6- Put your visitors at risk to the XSS attacks.

For more references, please visit the following:

- https://www.ostraining.com/blog/webdesign/against-using-iframes/

- https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html#X-Frame-Options_Header_Types

Several articles regarding this issue can be found Googling the keywords "iframe" and "security" if you wish more information.

Appendix 5 – PayOut Test Data

You may use the following Customer Data to perform tests with an expected outcome.

SCREENING PROCESS

In order to test the Screening process, please, use the data below to obtain the related outcome.

The “New Customer” need to use a valid e-mail and mobile phone number. All other Customers should contain only the CPF and e-mail stated below:

Customer Status	CPF	E-mail	Outcome
New Customer	15262971083		Success
Confirmed	08703247058	confirmed@gooutp4f.com	Success
Analysis	48832398028	analysis@gooutp4f.com	Customer in analysis
Blocked	36703436010	blocked@gooutp4f.com	Blocked Customer
Limit exceeded	53985822085	limitscreening@gooutp4f.com	Limit exceeded

APPENDIX 5 – PAYOUT TEST DATA

PAYOUT PROCESS

In order to test the PayOut process, after completing successfully the Screening process, please use the data below to obtain the related outcome:

Customer Status	CPF	E-mail	Pix Key Type	Outcome
Approved	56443853024	approved@gooutp4f.com	Email	Successful
Approved	56443853024	approved@gooutp4f.com	CPF	CPF analysis error
Limit exceeded	83137368014	limit@gooutp4f.com		Limit exceeded
Analysis	48832398028	analysis@gooutp4f.com		Customer in analysis
Blocked	36703436010	blocked@gooutp4f.com		Blocked Customer
Decline	78493353094	decline@gooutp4f.com		Declined

Appendix 6 – Pay4Fun Logo

You can find all Pay4Fun logos on:

https://blog.p4f.com/wp-content/uploads/2021/03/Package_P4F.zip