PAY FOR FUN

WALLET API – VERSION 1.7.0

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

TABLE OF CONTENTS

Contents

TABLE OF CONTENTS

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 (E-WALLET)

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

This API document is for Pay4Fun (e-wallet), where customers can transfer to the merchant website from his/hers pre-funded Pay4Fun account.

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

API Pay4Fun

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);
- Customize the Pay4Fun Authentication page with your logo;
- Dynamically control the minimum or maximum amount of payments that you want to offer;

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 following endpoints form the basis of a resource URI:

Warning - Sandbox URLs (Testing):

<P4F_URL>:

- **Website:** **http://test.p4f.com**

<P4F_API>:

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

Important – Production URLs (Live stage)

<P4F_URL>:

- **Website:** **https://p4f.com**

<P4F_API>:

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

Fast Account Registration

In order to make easy to your Customers enjoy the benefits of having a Pay4Fun account, we strongly suggest to use the Fast Registration feature.

The Fast Registration allows redirect Customers to a prepopulated Registration page where they can quickly create a Pay4Fun's account.

Just add the URL below, with real customer's information, to your cashier page

FAST REGISTRATION URL:

GET <P4F_URL>**/account/register?e=<e>&fn=<fn>&dob=<dob>&id=<id>**

FAST REGISTRATION PARAMETERS

Parameter	Data type	Description
e	String	Customer's e-mail
fn	String	Customer's Full name
dob	String	Customer's Date-of-birth (format YYYY-MM-DD)
id	String	Customer's Main ID (CPF)

Important

Before the account registration, Customers will be able to review/edit the prepopulated information.

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.

Web requests quota control per IP

Pay4Fun's API has a feature to control quota of web requests per IP; it means that requests originated from the same IP will be blocked if quota limits are reached.

The quota control has limits for different endpoints allowing an IP to make a maximum number of calls in a time interval.

The quota control exists to ensure the service is up and running for all Merchants, avoiding API running out of response capacity due to misuse of features or malicious requests.

The actual quota limits are specified below:

REQUESTS QUOTA CONTROL

API endpoint	HTTP verb	Quota (requests per second)
/api/1.0/payout/payout	POST	4
/api/1.0/payout/transaction	POST	3
/1.0/wallet/process	POST	6
/1.0/wallet/process/transaction	POST	3
* **(any other endpoints together)**	POST	6

These quota limits above are not customizable, they are the same for every Merchant.

Please, note that these limits may change without prior notice.

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 URLs (Testing):

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

Important – Production URLs (Live stage)

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

At the menu **ACCOUNT** / **API** / **CREDENTIALS**, 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

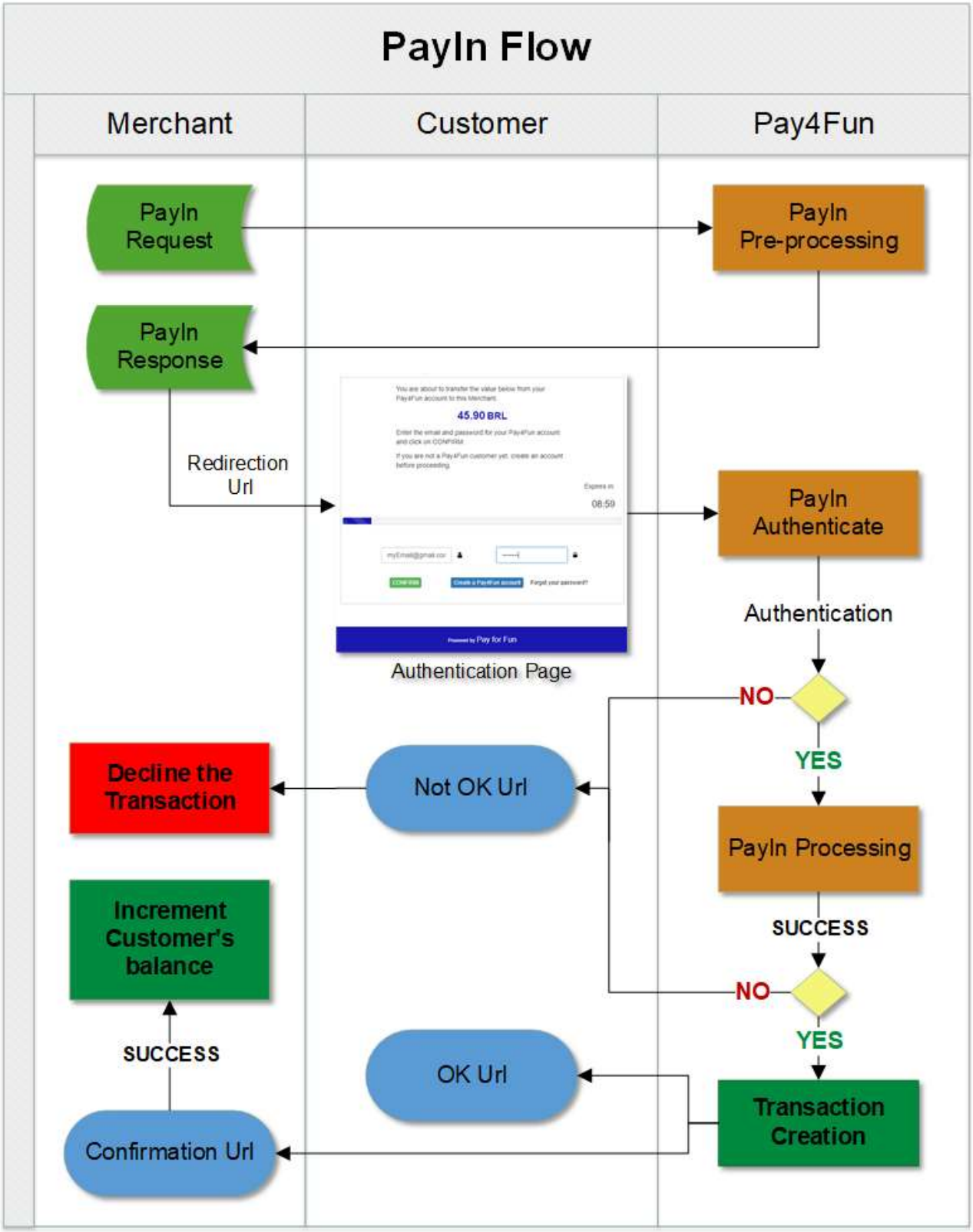
Not adding the IPs to your whitelist will avoid the API's confirmations to reach your servers.

Pay In

A Pay In transaction occurs when a Pay4Fun Customer transfers from his/hers wallet into the Merchant wallet.

There are no minimum and maximum limits pre-defined by Pay4Fun. Minimum and maximum limits are to be defined by the Merchant. Pay In transactions are never pending, it is either successful or not.

Pay in Flow Overview



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 debited from customer’s account and credited to merchant's account. This transaction is a successful transaction.

FAILED TRANSACTION

If for any reason the transaction processing fails, money is not deducted from customer’s account. **The 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

There is no pending API transaction status! The transaction or is processed successfully or not.

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.

PAY IN REQUEST

PAY IN API REQUEST:

POST <P4F_API>**/1.0/wallet/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",
   "merchantLogo": "https://merchant.com/image/logo.png"
}
```

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
merchantLogo	String	Merchant's logo URL (https – max height is 40 pixels)	No
p4fAccountEmail	String	Locks the transaction to the specific P4F's account e-mail address.	No
p4fMainId	String	Locks the transaction to the specific P4F's account Main Id	No
labelId	Integer	If a valid label ID is informed, transaction will be tagged. Please check Appendix 3 – Merchant Labels for more details	No

PAY IN REQUEST

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

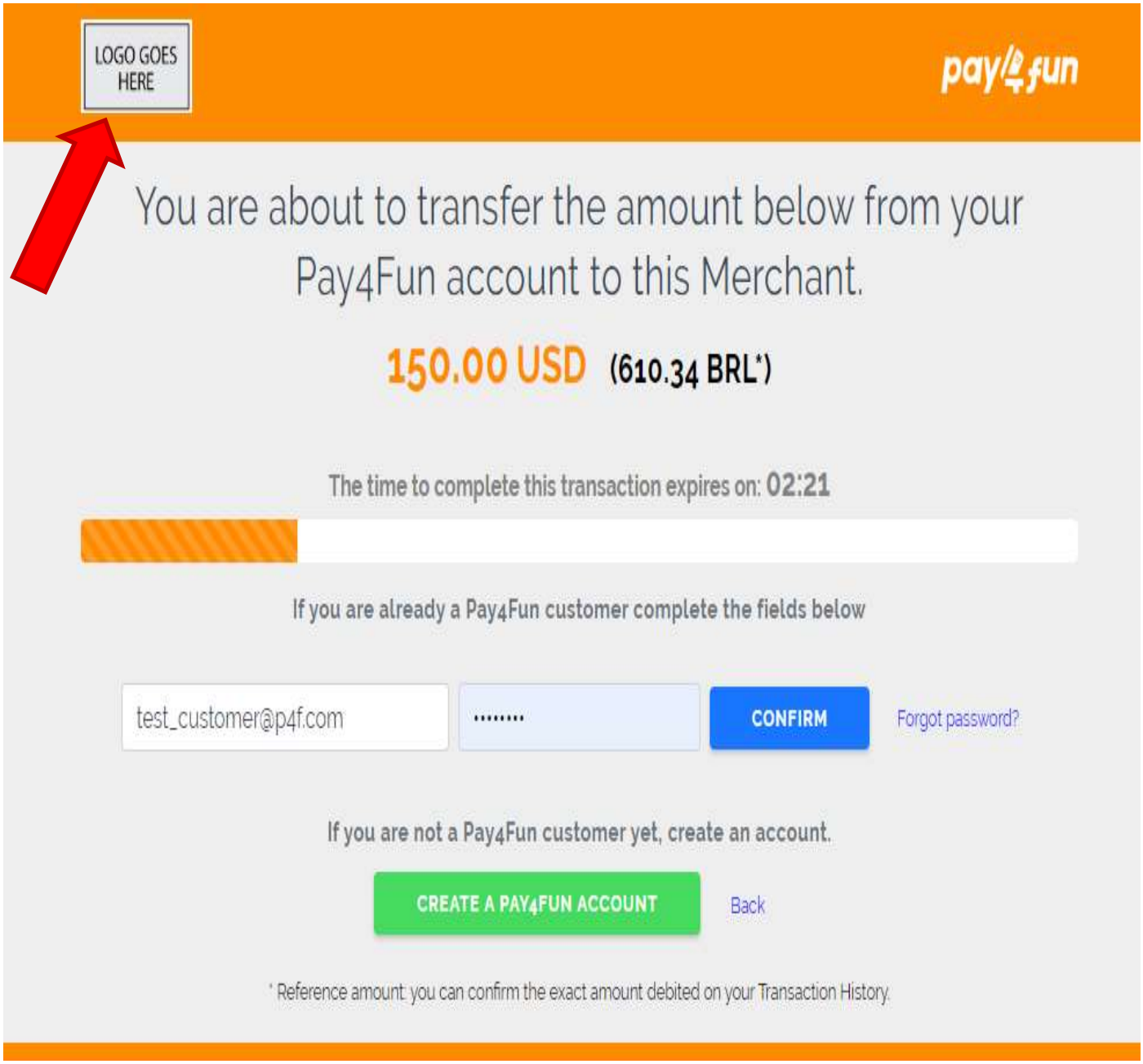
Even in cases of **flat amounts**, for instance, 1000.00, **cents must be included at the seed string**.

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.

Authentication Page

The authentication page is where your customers will provide P4F's credentials to perform transactions.

Check below how merchant's logo (sent by you as part of the Pay in request) will be used to customize the authentication page:



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.

In order to authenticate in this page, you should use a test P4F's account with funds. The credentials for authentication (email and password) will be provided on a need-to-know basis

After providing the authentication credentials - a valid Pay4Fun's account email and password – 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_URL>/api/payin/process/<KEY>"
}
```

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

PAY IN RESPONSE

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

Important

In case customer fails on providing valid credentials at the authentication page or has insufficient funds, no transaction will be generated by P4F which means you will not be able to see them at your back-office.

.

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.

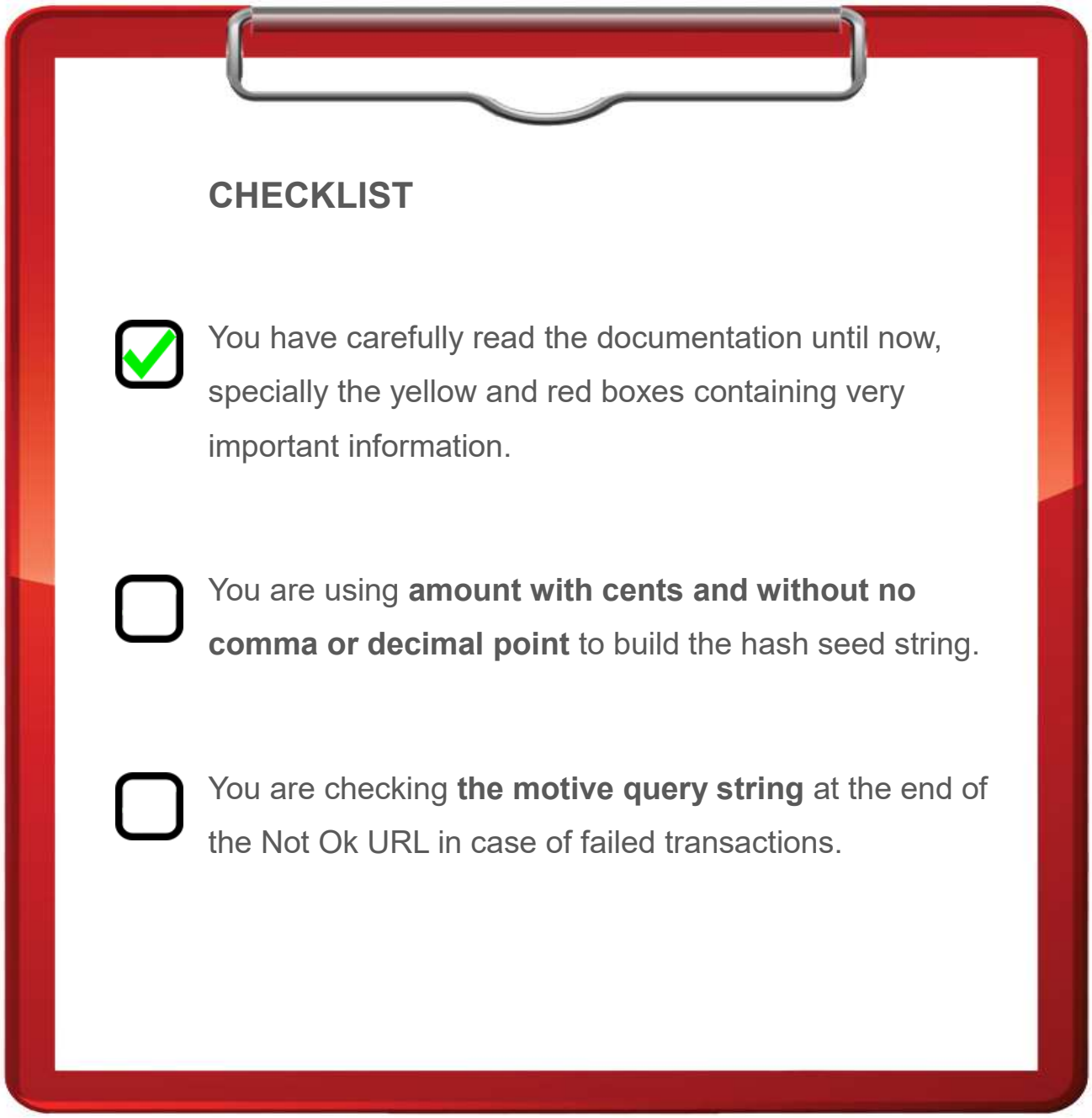
Important

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=**insufficient_balance**

Pay In Checklist

Before going any further and definitely before the Go Live, we suggest you go through this highly recommend checklist to confirm the integration is ready to go and avoid problems in the future:



Pay In Confirmation

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.

Please check **Pay In**

A Pay In transaction occurs when a Pay4Fun Customer transfers from his/hers wallet into the Merchant wallet.

There are no minimum and maximum limits pre-defined by Pay4Fun. Minimum and maximum limits are to be defined by the Merchant. Pay In transactions are never pending, it is either successful or not.

PAY IN CONFIRMATION

Pay in Flow Overview in order to better understand the information flow.

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

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

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.

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

> **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.

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.

PAY IN CONFIRMATION

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.

The response you will receive at you 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",
    "Hash": "10B9952C14DCC29259BFBA2DF5A5F8B9CA9AE1E11",
    "Sign": "0CE3325211878F6A9131252128EEAA1EB0398B594"
}
```


RESPONSE PARAMETERS SPECIFICATION

Parameter	Type	Description
TransactionId	String	P4F's transaction unique identifier
Amount	Decimal	Transaction amount
FeeAmount	Decimal	Transaction's fee amount

PAY IN CONFIRMATION

merchantInvoiceId	String	Merchant's transaction **unique identifier**
Currency		Pay in currency (ISO 4217 - 3 characters code)
Status	String	Status 201 denotes transaction success. 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	P4F's account email that has originated the transaction. Use it as additional information on compliance processes
Hash	String	This parameter is deprecated. Use **Sign** instead.
Sign	String	Contains details of the response hashed according **Pay in Sign** below

The response contains two hashed parameters to provide you ways to confirm that information hasn't been tampered: the parameters **Hash (deprecated)** and **Sign.**

PARAMETER HASH (DEPRECATED)

The parameter **Hash**, part of the confirmation request, contains the **Merchant's invoice ID** - sent by you in the original request - **hashed with your Merchant's key.** So, the seed string for this hash is your Merchant Invoice ID sent in the original request. Use this parameter to quickly check if the request is valid or not.

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

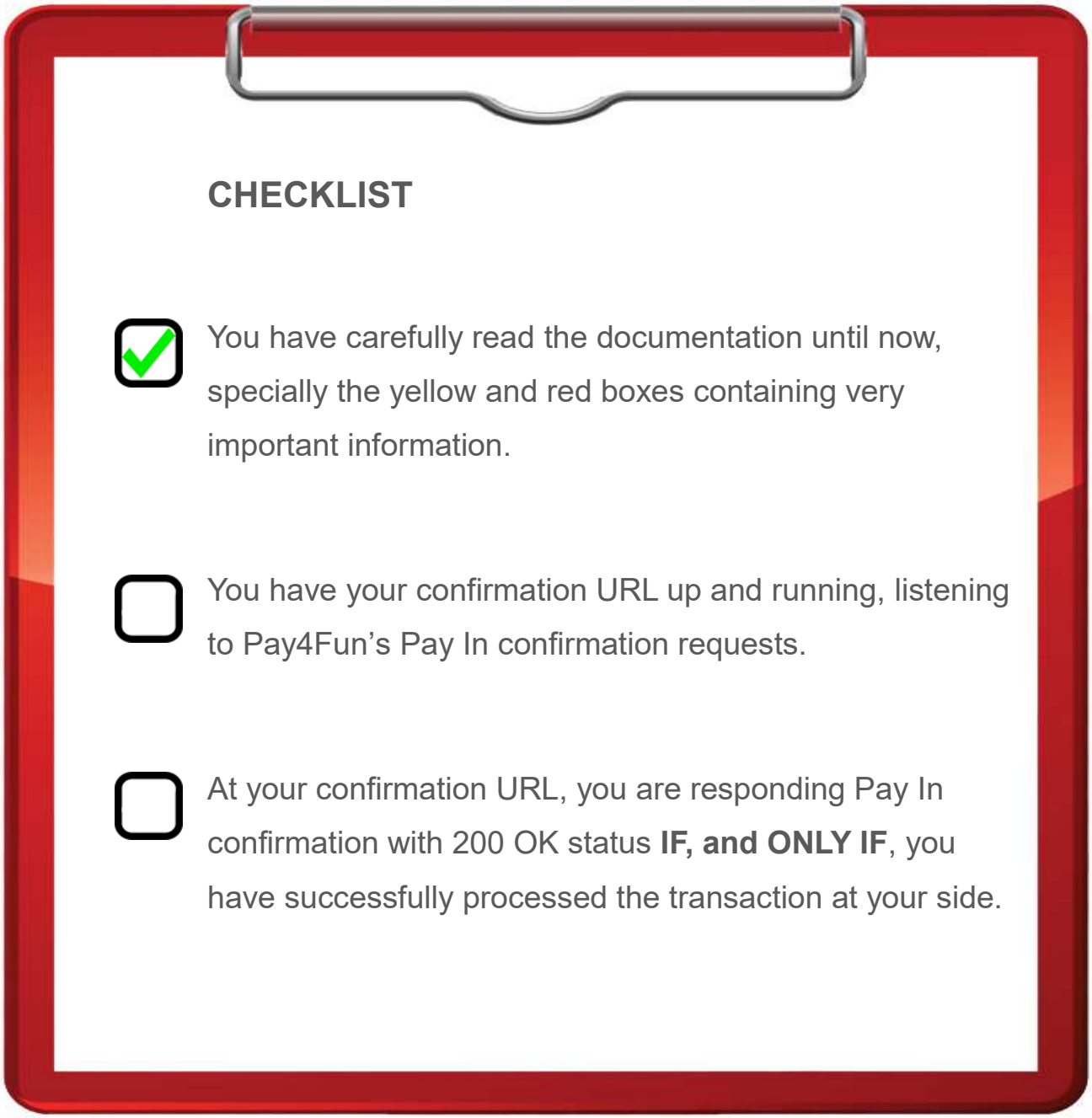
Seed string = **34567 1000 ABC777 201**

> **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 Confirmation Checklist

Before going any further and definitely before the Go Live, we suggest you go through this highly recommend checklist to confirm the integration is ready to go and avoid problems in the future:



Pay In Query Status

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

Important

We recommend 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!

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

PAY IN QUERY STATUS REQUEST:

POST <P4F_API>**/1.0/wallet/transaction/**

Header

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

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

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

PAY IN QUERY STATUS

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 Confirmation 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 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 Pay in Query Status request is like below:


```
[
  {
    "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

RESPONSE PARAMETERS SPECIFICATION

Parameter	Type	Description
transactionId	String	Pay4Fun Transaction ID
merchantInvocieId	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	Pay4Fun account's e-mail
LiquidationDate	String	Date the transaction's amount will be transfer to Merchant's Available Balance.

Regarding the pay in transaction's status you should consider as successful transactions the following:

- **VERIFIED**: transaction was successful and the amount transferred immediately to Merchant's Available Balance.
- **LIQUIDATION**: transaction was successful and the amount will be transferred to Merchant's Available Balance at liquidation date.

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**

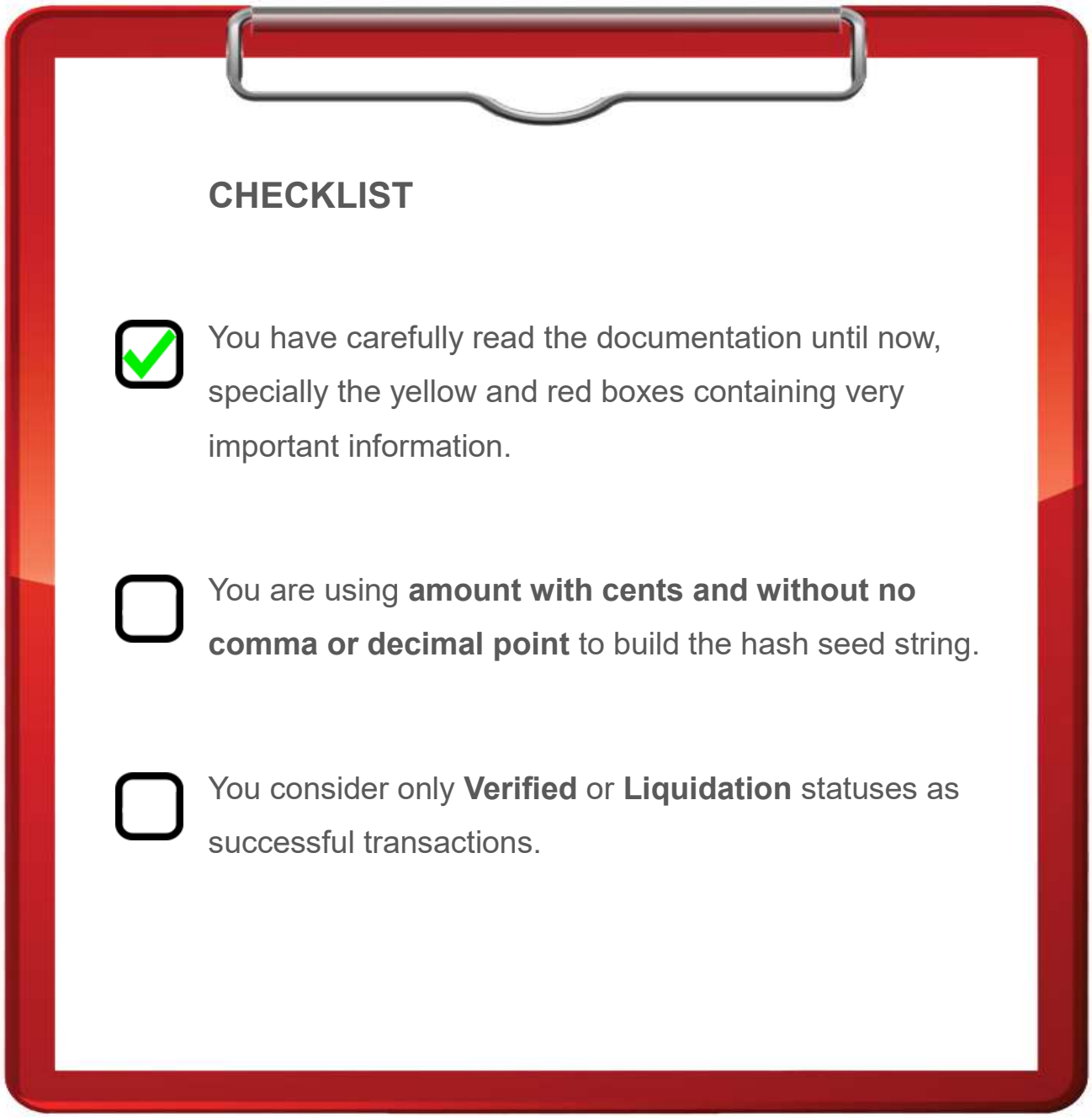
Important

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

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

Pay In Query Status Checklist

Before going any further and definitely before the Go Live, we suggest you go through this highly recommend checklist to confirm the integration is ready to go and avoid problems in the future:



Payout

A Payout transaction occurs when the Merchant transfers from its wallet into a Pay4Fun Customer wallet.

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

There are no minimum and maximum limits pre-defined by Pay4Fun. Minimum and maximum limits are to be defined by Merchant. Payout transactions are never pending, it is either successful or not.

There are two ways to conduct Payouts:

- Standard Payout: one-by-one, manually, directly in the admin back-office – no integration required.
- API Payout Request: please follow integration instructions below.

In both cases, make sure to include Pay4Fun withdraw option on Merchant website withdrawal page.

Only the Pay4Fun customer account (e-mail) and the amount is required to process the withdrawal.

We suggest the same Pay4Fun customer account (e-mail) used on deposit to be auto-filled on the withdrawal request or, alternatively, customer fill in the e-mail twice, to avoid typing mistakes.

Payout Request

This API endpoint allows merchants to perform amount transfers to customer accounts.

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

PayOut transaction only will be created after a series of validations. **In case any of them are not satisfied, transaction will be declined and will not be created at Merchant's back-office**; 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=**insufficient_balance**

Please, be careful to use the correct URL:

Sandbox (Testing):

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

Production (Live stage)

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

PAYOUT REQUEST

Payout API Request:

POST <P4F_API>/**api/1.0/payout/payout/**

Header

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

A valid request (**must be a JSON array**) would be:

```
[
    {
        "amount": 10.00,
        "currency": "BRL",
        "targetCustomerEmail": "customer@email.com",
        "merchantInvoiceId": "ABC777",
        "targetCustomerMainId": "XXXXXXXXXXXXXX",
        "sign": "878F6A89131E8AE221CE35A19BCB594EAE9F2"
    }, { . . . }
]
```

Important

Payout **request body must be a JSON array**, so, instead of sending individual requests, we recommend sending multiple Payouts within the same request.

PAYOUT REQUEST

REQUEST PARAMETERS SPECIFICATION

Parameter	Type	Description	Mandatory
hash	String	Request hash to avoid fraud (check Appendix 1 – Generating the Hash section)	Yes
amount	Decimal	Payout amount (2 digits after the decimal point)	Yes
merchantInvoiceId	String	Merchant's transaction **unique identifier** (maximum 250 characters)	Yes
currency	String	Payout currency (ISO 4217 - 3 characters code)	Yes
targetCustomerEmail	String	Specify the P4F's account e-mail address to transfer the amount to.	Yes
targetCustomerMainId	String	If informed, locks the transaction to the specific P4F's account Main Id	No
sign	String	Hash built with specific payout pieces of information to avoid fraud.	Yes
labelId	Integer	If a valid label ID is informed, transaction will be tagged. Please check Appendix 3 – Merchant Labels for more details	No

PAYOUT REQUEST

PAYOUT HASH SEED STRING

Concatenate the following information, in that order, to obtain the seed string for hash generation for Payout requests:

- Merchant's ID
- Payout batch size
- Merchant's secret

Assuming merchant's ID is **34567**, Payout batch size is **1** and merchant's secret is "**abcdef**", the seed string for hash generation, for a request containing just one payout (batch size equals 1), would be (in that order):

Seed string = **345671abcdef**

This hash will be used to validate the request before the processing starts; additionally, to the **Hash** parameter each payout within the batch should have its own **Sign** hash to allow Pay4Fun to validate the request authenticity.

The next section will describe how to build the **Sign** hash

PAYOUT SIGN SEED STRING

The parameter **Sign**, allows Pay4Fun to confirm the veracity of all relevant information contained within the request.

The Sign 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** 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. payout's currency;
4. target customer's email (Pay4Fun's account e-mail);
5. merchant's secret;

Seed string = **34567**1000USD**customer@email.com**abcdef

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

Payout Response

After receiving a Payout request, Pay4Fun will check the Sign hash for each payout within the batch and try to process them individually.

Warning

Payout's batch size is limited to 20 transactions per request.

The response will contain the following parameters for each payout within the batch:

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

```
{
      "transactionId": 98765,
      "currency": "BRL",
      "amount": 10.00,
      "feeAmount": 0.43,
      "merchantInvoiceId": "ABC777",
      "originalCurrency": "USD",
      "originalAmount": 2.00,
      "status": "Confirmed",
      "message": "success",
      "customerEmail": "customer@email.com",
      "sign": "878F6A89131E8AE221CE35A19BCB594EAE9F2"
}
```

PAYOUT RESPONSE

RESPONSE PARAMETERS SPECIFICATION

Parameter	Type	Description
TransactionId	String	P4F's transaction unique identifier
Currency		Payout currency in the regards of Merchant's account currency (ISO 3 characters code)
Amount	Decimal	Transaction amount in the regards of Merchant's account currency
FeeAmount	Decimal	Transaction's fee amount in the regards of Merchant's account currency
MerchantInvoiceId	String	Merchant's transaction **unique identifier**
OriginalCurrency	String	Original Payout currency (sent within the request - ISO 3 characters code)
OriginalAmount	Decimal	Original Payout amount (sent within the request)
Status	String	Status **Confirmed** denotes transaction success. Any other status denotes failed transaction
Message	String	Success or fail message
TargetCustomerEmail	String	P4F's account email that has originated the transaction. Use it as additional information on compliance processes
Sign	String	Contains details of the response hashed according **Payout Response Sign Hash** below

PAYOUT RESPONSE

PAYOUT RESPONSE SIGN HASH

The parameter **Sign**, allows you to confirm the veracity of all relevant information contained within the response.

For **successful payouts**, in order to generate and compare the Sign hash, you should concatenate the following payout's response pieces of information:

1. payout's amount (**Amount** response's parameter) with cents;
2. payout's currency (**Currency** response's parameter);
3. target customer e-mail;
4. merchant's invoice ID;

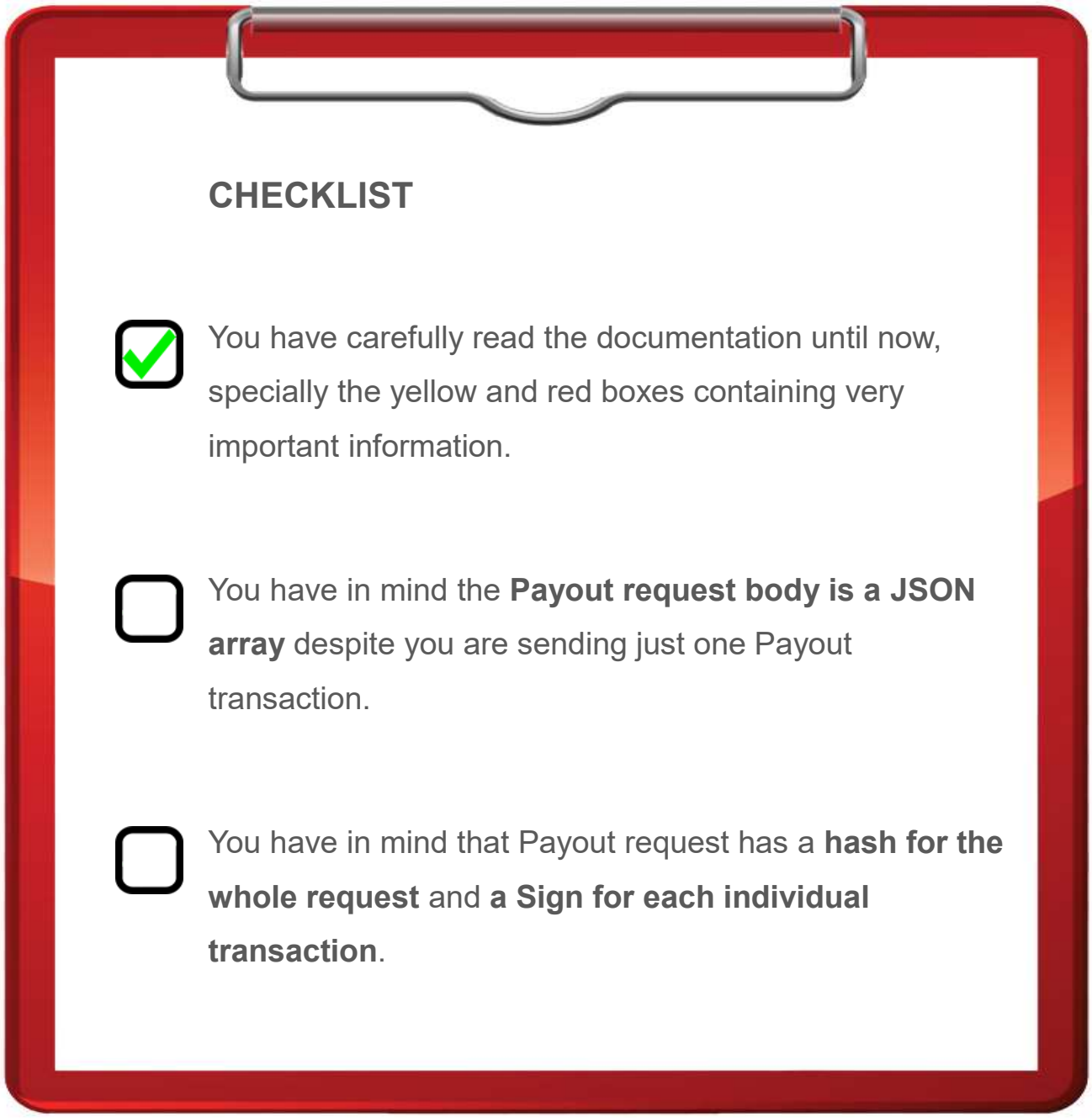
For failed payouts, no Sign hash will be generated, and you should focus on the **Message** parameter to check the motive of the failure.

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

Please, refer to the **Appendix 2 – Error Messages** for the possible error messages.

Payout Checklist

Before going any further and definitely before the Go Live, we suggest you go through this highly recommend checklist to confirm the integration is ready to go and avoid problems in the future:



Payout Query Status

This API endpoint allows the Merchant to retrieve the status of a list of Pay-outs by searching for its Merchant's invoice ids.

Important

We recommend 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!

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

PAYOUT QUERY STATUS REQUEST:

POST <P4F_API>**/api/1.0/payout/transaction/**

Header

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

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

Body

```
[
    "24234", "24254"
]
```

PAYOUT QUERY STATUS

Request parameters specification

Parameter	Type	Description	Mandatory
merchantId	String	Pay-out's 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-outs **unique identifiers**	Yes

Concatenate the following information, in that order, to obtain the seed string for hash generation for Pay-out Confirmation requests:

- 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 Pay-out Query Status request is like below:


```
[
      {
            "transactionId": 31,
            "merchantInvoiceId": "24234",
            "status": "Verified",
            "sign": "878F6A3891E58AE25B039894EAE9F240",
            "amount": 37.14,
            "customerEmail": "customer@email.com",
            "liquidationDate": ""
      },{
            "transactionId": 0,
            "merchantInvoiceId": "24254",
            "status": "NotFound",
            "sign": "78F6A3891E58A25B039894EAE9F24360",
            "amount": 0,
            "customerEmail": null,
            "liquidationDate": null
      }
]
```

PAYOUT QUERY STATUS

RESPONSE PARAMETERS SPECIFICATION

Parameter	Type	Description
transactionId	String	Pay4Fun Transaction ID
merchantInvocieId	String	Pay-out's Merchant's Transaction ID
status	String	Transaction Status
sign	String	Hash sign to allow veracity confirmation
amount	Decimal	Transaction amount
customerEmail	String	Pay4Fun account's e-mail
LiquidationDate	String	Empty.

Regarding the transaction's status you should consider as successful transactions the following:

- **VERIFIED**: transaction was successful and the amount transferred immediately to Merchant's Available Balance.

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

PAYOUT QUERY STATUS

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

Seed string = **3456710000ABC777Verified**

Important

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

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

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: **merchant_not_authorized**.

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	Scope
customer_not_authorized	Customer account's is suspended or blocked thus not able to perform transactions	Pay In Payout
customer_not_found	The Payout request parameter "CustomerEmail" contains a non-valid Pay4Fun's account	Payout
declined_by_customer	Transaction was declined intentionally by the Customer.	Pay In Payout
email_locked_by_merchant	The Pay4Fun's account e-mail used to authenticate the Transaction is different from the account e-mail specified by the Merchant within the request.	Pay In
external_invoice_id_already_used	There is a previous Transaction using the specified Merchant's Invoice ID.	Pay In Payout
insufficient_balance	Merchant's account (Payout) or Customer's account (Pay In) hasn't sufficient balance to perform the transaction.	Pay In Payout

invalid_customer_main_id	The target Pay4Fun's customer account has a different Main ID that the value specified within the request.	Pay In
invalid_request	The requested sent by Merchant is not valid.	Pay In Payout
invalid_request_currency	The Currency sent within the request is not valid.	Pay In Payout
invalid_security_hash	Something is wrong with the Hash sent within the request. Check the instructions on how generate the Hash correctly.	Pay In Payout
maximum_payout_batch_size_exceeded	Transaction's batch size is greater than permitted.	Payout
merchant_account_not_live	The Merchant account is not yet Live.	Pay In Payout
merchant_ip_not_authorized	The IP used to perform API request is not whitelisted.	Pay In Payout
merchant_not_authorized	The Merchant's account is not enabled.	Pay In Payout
merchant_not_found	The Merchant ID sent within the request is not valid.	Pay In Payout
processing_error	Something very bad has happened while processing Merchant's request.	Pay In Payout
timeout	The time available for authentication in the API authentication page has expired.	Pay In Payout

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/#:~:text=Reason%20%231.&text=If%20you%20create%20an%20iframe,change%20the%20source%20site%20URL.

- 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 – Pay4Fun Logo

You can find all Pay4Fun logos on:

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