

## Sportsbook API integration Last update: 2023/03/30

### Terms and concepts

- CLIENT - the company which will integrate sportsbook.
- OPERATOR - the company which provides sportsbook.

### The CLIENT provides:

	Staging	Production
Server IP address	54.151.247.5	54.151.247.5
Callback URL	https://backend.jackpotx.net/api/payment/webhook/igpx	
Website URL	https://jackpotx.net	https://jackpotx.net
Currency or currencies	USD, EUR, GBP	USD, EUR, GBP
Supported languages	en, es, fr	en, es, fr

### After filling this form, OPERATOR will provide following variables

<API\_URL> - CLIENT will call by this URL to interact with the OPERATOR's server  
<API\_VERSION>  
<CLIENT\_USERNAME>  
<CLIENT\_PASSWORD>  
<SECURITY\_HASH>

This integration consists of 2 parts. The first part describes requests which CLIENT must make. And the second part describes requests which call OPERATOR and must be handled by CLIENT.

## CLIENT requests

### Auth

Firstly, CLIENT must do auth request to get a token

Method: POST Params:

- username - CLIENT\_USERNAME
- password - CLIENT\_PASSWORD

#### Request:

```
POST <API_URL>/<API_VERSION>/auth
Content-Type: application/json

{
  "username": "<CLIENT_USERNAME>",
  "password": "<CLIENT_PASSWORD>"
}
```

#### Response:

```
{
  "token": "<CLIENT_TOKEN>",
  "expires_in": "<TOKEN_EXPIRATION>"
}
```

- token - will be used in next requests.
- expires\_in - shows (in seconds) how much time the token will be valid.

## Start session

When the player opens the website, the CLIENT must do this request to get a URL which will be opened in player's browser in `<iframe />`

Method: POST Params:

- `user_id` - CLIENT-side player Identified, must be a String, required.
- `urrency` - player's currency, required.
- `lang` - language code, optional.

### Request:

```
POST <API_URL>/<API_VERSION>/start-session
Content-Type: application/json
Authorization: Bearer <CLIENT_TOKEN>

{
  "user_id": "<USER_ID>",
  "currency": "<CURRENCY>",
  "lang": "<LANGUAGE_CODE>"
}
```

### Response:

```
{
  "url": "<IFRAME_URL>"
}
```

## Transactions

This section describes the requests which will be called by OPERATOR by the provided CallbackURL and must be handled by CLIENT.

All requests' methods are POST and each one Contains extra header "X - Security-Hash". When the CLIENT server receives a request, it must take the raw body and create a hash by using the Sha256 algorithm and use <SECURITY\_HASH> as secret, then compare "X -Security-Hash" with the calculated hash and accept only that requests where hashes are match. All responses must have http status 200 and must be in JSON format, when the request is processed successfully, the error field must be null, otherwise the error must have an appropriate value.

```
# Success
{
  "error": null,
  . . .
}

# Error
{
  "error": "INTERNAL_ERROR"
}
```

- Transaction request
- Transaction-rollback request

## Transactions

### Request:

```
POST <CALBACK_URL>/transaction
X-Securty-Hash: <SECURITY_HASH>
Content-Type: application/json

{
  "transaction_id": "<TRANSACTION_ID>",
  "action": "bet|result",
  "user_id": "<USER_ID>",
  "currency": "<CURRENCY>",
  "amount": "<AMOUNT>"
}
```

- transaction\_id - OPERATOR side unique identifier of the transaction
- action - this field can have two values "bet" or "result", in case of "bet" the CLIENT must withdraw the amount from the user's balance, and in case of "result" the CLIENT must deposit the amount to the user's balance.
- user\_id & currency - equals to fields from /start-session requests
- amount - decimal number, it is the amount to be subtracted or added

### Response:

```
{
  "error": null,
  "transaction_id": "<TRANSACTION_ID>"
}
```

- transaction\_id - is the CLIENT side transaction identifier, must be unique

```
{
  "error": "INSUFFICIENT_FUNDS"
}
```

All requests must be handled only once. When the CLIENT receives the transaction with the same transaction\_id from OPERATOR balance user should not be changed, but the response must be successful and be with the same transaction id as the first time. The CLIENT may decline bet requests when the player is banned or has insufficient balance, but all result requests must be processed successfully. "Bet" requests will only be sent once when a player places a bet, but "result" requests will be sent a few more times if not answered.

## Transaction-rollback

### Request:

```
POST <CALBACK_URL>/transaction-rollback
X-Securty-Hash: <SECURITY HASH>
Content-Type: application/json

{
  "transaction_id": "<TRANSACTION_ID>",
  "rollback_transaction_id": "<ROLLBACK_TRANSACTION_ID>",
  "action": "rollback",
  "user_id": "<USER_ID>",
  "currency": "<CURRENCY>",
  "amount": "<AMOUNT>"
}
```

- transaction\_id - OPERATOR side unique identifier of the transaction
- rollback\_transaction\_id - OPERATOR side transaction identifier which must be rolled back
- action - always have "rollback" value.
- user\_id & currency - equals to fields from /start-session requests
- amount - decimal number, it is the amount to be subtracted or added

### Response:

```
{
  "error": null,
  "transaction_id": "<TRANSACTION_ID>"
}
```

- transaction\_id - the CLIENT-side transaction identifier, must be a unique

All requests must be handled only once. When the CLIENT receives the transaction with the same transaction\_id from OPERATOR balance user should not be changed, but the response must be successful and be with the same transaction id as the first time. If the transaction with

"rollback\_transaction\_id" does not exist the transactions must be inserted without changing the user's balance and be handled successfully. Rollback requests, like result requests, can be sent multiple times.