Slotegrator

Reverse Integration

Reverse Integration

Overview

This document describes an API based on HTTP/1.1 protocol [[RFC 2616] (https://tools.ietf.org/html/rfc2616)].

Document version

1.0.3

Changelog

Version (date)	Changes description
1.0.1 (15.09.2020)	New parameter **return_url** wasadded into "Get game url" request. New parameter**game_id** was added into "Bet", "Win", "Refund" requests.
1.0.2 (13.04.2021)	Error handling for "Bet" transactions was added. Changed type of parameter **amount** in "Bet" and "Win" requests. Extended description of provider actions in case ot timeout during "Win" and "Refund" transactions.
1.0.3 (01.07.2021)	New parameter **return_url** was added into "Get demo game url" request. New parameter **language**was added into "Get demo game url" request.

Links

- RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1
- ISO 4217, Currency codes
- ISO 8601, Date and time format
- ISO 639-1, Language format

Integration data provided by Game Aggregator

1. Token
2. Client id
3. Endpoints for requests

Integration data provided by Game Provider

1. Endpoint for get demo game URL
2. Endpoint for get game URL
3. Endpoint for get jackpots list (optional)
4. Endpoint for create freespin campaign (optional)
5. Endpoint for cancel freespin campaign (optional)

Request format

Default request format is *json* with *Content-Type: application/json* header

Response format

Default response format is *json* with *Content-Type: application/json* header

Error response

```
{
  "status" : false,
  "code" : "WRONG_INPUT_PARAMETERS",
  "message": "Wrong input parameters"
}
```

Error codes

INTERNAL_ERROR

SESSION_NOT_FOUND

TRANSACTION_PREPARING_ERROR

TRANSACTION_DENIED

TRANSACTION_IN_PROGRESS

INVALID_SIGN

WRONG_INPUT_PARAMETERS

AMOUNT_SHOULD_BE_POSITIVE

INSUFFICIENT_BALANCE

BET_FAILED_KNOWN_ERROR

BET_FAILED_UNKNOWN_ERROR

ACTION_IS_NOT_EXIST

Requests from *Game Aggregator* to *Game Provider*

[POST /] Get demo game url

Request fields

- *client_id: string* (id of client)
- *game_id: string* (id of game)
- *language: string* (language of game)
- *return_url: string* (URL for exit to casino)

Request example

```
{
  "client_id": "myClient1",
  "game_id": "some_game_id",
  "language": "en",
  "return_url": "https://example.domain.com/api/exit?uuid=4b6d955cdab74fac919b4ebd05c81874"
}
```

Response fields

- url: string (url for launch demo game)

Response example

```
{
  "url": "http://127.0.0.1/"
}
```

[POST /] Get game url
Request fields

- *client_id: string* (id of client)
- *game_id: string* (id of game)
- *currency: string* (currency name)
- *language: string* (language of game)
- *session_id: string* (session id that will be send)
- *player_id: string* (id of player)
- *return_url: string* (URL for exit to casino)

Request example

```
{
  "client_id": "myClient1",
  "game_id": "some_game_id",
  "currency": "USD",
  "language": "en",
  "session_id": "c4ca4238a0b923820dcc509a6f75849b",
  "player_id": "4f6ad0ca25074bec9dd22cf3a689ddb4USD",
  "return_url": "https://example.domain.com/api/exit?uuid=4b6d955cdab74fac919b4ebd05c81874"
}
```

Response fields

- url: string (url for launch game)

Response example

```
{
  "url": "http://127.0.0.1/"
}
```

[POST /] Jackpots (optional)
Request fields

- *client_id: string* (id of client)

Request example

```
{
  "client_id": "MyClient1"
}
```

Response fields

- *status: boolean* (status of request)
- *jackpots[].name: string* (name of level)
- *jackpots[].amount: float* (jackpot amount)
- *jackpots[].currency: currency* (currency of the jackpot)

Response example

```
{
  "status": true,
  "jackpots": [
      {
        "name": "Bronze",
        "amount": 100.0000,
        "currency": "USD"
      },
      {
        "name": "Silver",
        "amount": 1000.0000,
        "currency": "USD"
      },
      {
        "name": "Gold",
        "amount": 10000.0000,
        "currency": "USD"
      }
  ]
}
```

[POST /] Create freespin campaign (optional)

Request fields

- *client_id: string* (id of client)
- *game_id: string* (game id with freespins)
- *player_id: string* (player id)
- *campaign_id: string* (campaign id on the game aggregator side)
- *currency: string* (currency code)
- *quantity: integer* (count of freespins)
- *valid_from: integer* (start time freespin campaign UTC+0)
- *valid_until: integer* (end time freespin campaign UTC+0)
- *bet_id: integer* (bet per line index of value)

Request example

```
{
  "client_id": "MyClient1",
  "game_id": "some_game_name",
  "player_id": "c81e728d9d4c2f636f067f89cc14862cUSD",
  "campaign_id": "c4ca4238a0b923820dcc509a6f75849b",
  "currency": "USD",
  "quantity": 10,
  "valid_from": 1537522916,
  "valid_until": 1537522929,
  "bet_id": 1
}
```

Response fields

- *status: boolean* (status of request)
- *new_campaign_id: string* (id created campaign)

Response example

```
{
  "status": true,
  "new_campaign_id": "15de21c670ae7c3f6f3f1f37029303c9"
}
```

[POST /] Cancel freespin campaign (optional)

Request fields

- *client_id: string* (client id)
- *campaign_id: string* (campaign id on the game aggregator side)

Request example

```
{
  "client_id": "MyClient1",
  "campaign_id": "c4ca4238a0b923820dcc509a6f75849b"
}
```

Response fields

- *status: boolean* (status of request)

Response example

```
{
  "status": true
}
```

Requests from *Game provider* to *Game aggregator*

[POST /] Bet

Request fields

- *action: string* (action name 'bet')
- *type: string* (bet type 'bet', 'freespin', 'tip')
- *session_id: string* (session id provided by Game Aggregator)
- *amount: string* (bet amount)
- *transaction_id: string* (unique id of transaction)
- *round_id: string* (id of round)
- *game_id : string* (id of current game)

Request example

```
{
  "action": "bet",
  "type": "bet",
  "session_id": "c4ca4238a0b923820dcc509a6f75849b",
  "amount": 10.0001,
  "transaction_id": "eccbc87e4b5ce2fe28308fd9f2a7baf3",
  "round_id": "3",
  "game_id": "current_game_id"
}
```

Response fields

- *status: boolean* (status of request)
- *balance: float* (balance amount, max precision - 4)

Response example

```
{
  "status": true,
  "balance": 100.0001
}
```

Error handling:

You can get 3 types of errors from us:

- Known error
- Unknown error
- Timeout more than 5 seconds

In case of known error you have to show an error message from our response in the game.

By known errors we mean:

```
{
  "status":false,
  "code":"INTERNAL_ERROR",
  "message":"Not enough limits"
}
```

and

```
{
  "status":false,
  "code":"INSUFFICIENT_BALANCE",
  "message":"Insufficient balance"
}
```

In case of unknown error you have to show an error message “Unknown error” in the game and send us a refund request.

Unknown error response example:

```
{
  "status":false,
  "code":"BET_FAILED_UNKNOWN_ERROR",
  "message":"Bet failed unknown error"
}
```

In case we are not responding to you more than 5 seconds you have to send us a refund request.

[POST /] Win

In case of any erroror response delay more than 5 seconds you have to resend it until you get a successful response

Request fields

- *action: string* (action name 'win')
- *type: string* (win type 'win', 'freespin', 'jackpot')
- *session_id: string* (session id provided by Game Aggregator)
- *amount: string* (win amount)
- *transaction_id: string* (transaction id)
- *round_id: string* (id of round)
- *game_id : string* (id of current game)

Request example

```
{
  "action": "win",
  "type": "win",
  "session_id": "c4ca4238a0b923820dcc509a6f75849b",
  "amount": 10.0001,
  "transaction_id": "eccbc87e4b5ce2fe28308fd9f2a7baf3",
  "round_id": "3",
  "game_id": "current_game_id"
}
```

Response fields

- *status: boolean* (status of request)
- *balance: float* (balance amount, max precision - 4)

Response example

```
{
  "status": true,
  "balance": 110.0001
}
```

[POST /] Refund

In case of any erroror response delay more than 5 seconds you have to resend it until you get a successful response

Request fields

- *action: string* (action name 'refund')
- *session_id: string* (session id provided by Game Aggregator)
- *transaction_id: string* (bet transaction id)
- *round_id: string* (id of round)
- *game_id : string* (id of current game)

Request example

```
{
  "action": "refund",
  "session_id": "c4ca4238a0b923820dcc509a6f75849b",
  "transaction_id": "eccbc87e4b5ce2fe28308fd9f2a7baf3",
  "round_id": "3",
  "game_id": "current_game_id"
}
```

Response fields

- *status: boolean* (status of request)
- *balance: float* (balance amount, max precision - 4)

Response example

```
{
  "status": true,
  "balance": 110.0001
}
```

[POST /] Balance

Request fields

- *action: string* (action name 'balance')
- *session_id: string* (session id provided by Game Aggregator)

Request example

```
{
  "action": "balance",
  "session_id": "c4ca4238a0b923820dcc509a6f75849b"
}
```

Response fields

- *status: boolean* (status of request)
- *balance: float* (balance amount, max precision - 4)

Response example

```
{
  "status": true,
  "balance": 100.0001
}
```

Security

All requests contain authorization headers.

Authorization headers

- X-Sign: Sign calculated with sha256 hmac

X-Sign calculation
Sha256 hmac algorithm with Token is used for signing.

PHP example of the X-Sign calculation:

```
$token = 'MVaoC9xKdJesozkX';

$requestParams =  [
 'client_id' => 'abcd12345',
 'game_id'   => 'zxcv67890',
];

$requestJson = json_encode($requestParams);

$sign = hash_hmac('sha256', $requestJson, $token);

$headers = [
 'Content-Type: application/json',
    'X-SIGN: ' . $sign
];
```