



This document is strictly confidential and is subject
to a non-disclosure agreement.

API Slotegrator

Overview

This document describes an API based on HTTP/1.1 protocol [RFC 2616](#).

| | |
|------------------|--|
| Document version | 1.2.0 |
| Links | <ul style="list-style-type: none">- RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1- ISO 4217, Currency codes- ISO 639-1, Languages codes- ISO 8601, Date and time format |

Changelog

| Version (date) | Change description |
|--|---|
| 1.0.0 (2016-09-01, d6) | Documentation initialized |
| 1.0.1 (2016-09-07, d2) | Games/init POST request format specified |
| 1.0.2 (2016-09-15, d2) | Limits and self-validate endpoints added |
| 1.0.3 (2016-09-23, d2) | Specify response status on duplicate requests |
| 1.0.4 (2016-10-20, d2) | "is_mobile" parameter added to games |
| 1.0.5 (2016-11-15, d2) | Demo mode |
| 1.0.6 (2017-02-17, d2) | Updated "/limits" response |
| 1.0.7 (2017-03-21, d2) | Added "/jackpots" endpoint |
| 1.0.8 (2017-03-30, d2) | Added "game_uuid" and "player_id" to "bet", "win" and "refund" requests |
| 1.1.0 (2018-02-22, d2) | Added freespins |
| 1.1.1 (2019-07-04, d2) | Added balance notifications |
| 1.1.2 (2020-08-25, d2) | Parameter "is_finished" changed to "finished". |
| 1.1.3 (2020-10-06, d2) | Rollback. Parameter `round_id` is fixed. |
| 1.1.4 (2020-11-25, d2) | FreeSpins. Property `total_bets` added. |
| 1.2.0 (2022-10-21) | <div>Updated "/games"</div> <ul style="list-style-type: none">• Added `expand` request parameter and available expansions list.• Added optional win parameters "bonus", "pragmatic_prize_drop" and "pragmatic_tournament"• Added optional refund parameters "bet", "tip" and "fre spin" |

Game Aggregator

Overview

Integration data provided by Game Aggregator

1. Merchant ID
2. Merchant Key
3. Base API URL

Endpoints and Base API URL

For example, **Base API URL** is `*https://example.com/api/v1*` and **Endpoint** is `*/games/lobby*`

Then calls from integrator to Game Aggregator should be done to `*https://example.com/api/v1/games/lobby*`

Request format

Query parameters should be passed with `application/x-www-form-urlencoded` content type.

Response format

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

List of used HTTP codes

| Code | Interpretation |
|------|--|
| 200 | OK. Everything worked as expected. |
| 201 | A resource was successfully created in response to a `POST` request. The `Location` header contains the URL pointing to the newly created resource. |
| 204 | The request was handled successfully and the response contains no body content (like a `DELETE` request). |
| 304 | The resource was not modified. You can use the cached version. |
| 400 | Bad request. This could be caused by various actions by the user, such as providing invalid JSON data in the request body, providing invalid action parameters, etc. |
| 401 | Authentication failed. |
| 403 | The authenticated user is not allowed to access the specified API endpoint. |
| 404 | The requested resource does not exist. |
| 405 | Method not allowed. Please check the `Allow` header for the allowed HTTP methods. |
| 415 | Unsupported media type. The requested content type or version number is invalid. |
| 422 | Data validation failed (in response to a `POST` request, for example). Please check the response body for detailed error messages. |
| 429 | Too many requests. The request was rejected due to rate limiting. |
| 430 | Internal server error. This could be caused by internal program errors. |

Error response

Generic errors

Generic error response contains a single object with following attributes:

| Attribute: data type | Description |
|---------------------------------|-------------------|
| `name`, `string` | Exception name |
| `message`, `string` | Exception message |
| `code`, `integer`, `default: 0` | Exception code |
| `status`, `integer` | HTTP status code |

Example

| Response |
|--|
| HTTP/1.1 404 Not Found <pre>{ "name": "Not Found Exception", "message": "The requested resource was not found.", "code": 0, "status": 404 }</pre> |

Collections

Collection is a set of objects of the same type. There is an additional metadata for working with collections like pagination or sorting.

Pagination headers

By default, pagination metadata is available via HTTP headers:

| Attribute | Description |
|-----------|-------------|
|-----------|-------------|

| | |
|-----------------------------|--|
| `X-Pagination-Total-Count` | The total number of resources |
| `X-Pagination-Page-Count` | The number of pages |
| `X-Pagination-Current-Page` | The current page (1-based) |
| `X-Pagination-Per-Page` | The number of resources in each page |
| `Link` | A set of navigational links allowing client to traverse the resources page by page |

Collections enveloping

In case your client is incapable of working with HTTP headers you are able to receive this information within response body.

Response

HTTP/1.1 200 OK

...

X-Pagination-Total-Count: 1000

X-Pagination-Page-Count: 50

X-Pagination-Current-Page: 1

X-Pagination-Per-Page: 20

Link: <https://example.com/endpoint?page=1>; rel=self, <https://example.com/endpoint?page=2>; rel=next, <https://example.com/endpoint?page=50>; rel=last Content-Type: application/json; charset=UTF-8

```
{
  "items": [
    {
      "id": 1,
      ...
    },
    { "id": 2,
      ...
    },
    ...
  ],
  "_links": {
    "self": {
      "href": "https://example.com/endpoint?page=1"
    },
    "next": {
      "href": "https://example.com/endpoint?page=2"
    },
    "last": {
      "href": "https://example.com/endpoint?page=50" }
  },
  "_meta": {
    "totalCount": 1000, "pageCount": 50, "currentPage": 1, "perPage": 20
  }
}
```

Game launch flow

Games should be stored/cached on the client side after retrieval. Game could be launched in several steps according to scenario based on lobby availability.

Games without lobby:

1. Call `/games/init`
2. Launch game by redirecting player to the provided URL

Games with lobby:

1. Call `/games/lobby`
2. Call `/games/init` with provided `lobby_data`
3. Launch game by redirecting player to the provided URL

> Info: More info on `/games`, `/games/lobby` and `/games/init` endpoints could be found in corresponding documentation sections.
 > Note: Base API URL should be provided by manager.

Security

All requests should contain authorization headers (except **Launch** phase with player redirection).

Authorization headers

| Attribute | Description |
|------------------------------|---|
| <code>'X-Merchant-Id'</code> | Merchant ID provided by integration manager |
| <code>'X-Timestamp'</code> | Request timestamp. If differ from current timestamp for more than 30 seconds - request considered expired |
| <code>'X-Nonce'</code> | Random string |
| <code>'X-Sign'</code> | Sign calculated with sha1 hmac |

X-Sign calculation

1. Merge request array with authorization headers array
2. Sort resulting array by key in ascending order
3. Generate a URL-encoded query string from this array
4. Use sha1 hmac algorithm with Merchant Key (provided by integration manager) for signing

PHP example of the X-Sign calculation

```
```php
$merchantKey = 'Merchant Key provided by integration manager';

$headers = [
 'X-Merchant-Id' => 'value',
 'X-Timestamp' => time(),
 'X-Nonce' => md5(uniqid(mt_rand(), true)),
];

$requestParams = [
 'game_uuid' => 'abcd12345',
 'currency' => 'USD',
];

$mergedParams = array_merge($requestParams, $headers);
ksort($mergedParams);
$hashString = http_build_query($mergedParams);

$XSign = hash_hmac('sha1', $hashString, $merchantKey);
```

## Example

### Request

GET /games

X-Merchant-Id: ff955b5759b3885f08cf125d4454ceb4  
 X-Timestamp: 1471857411  
 X-Nonce: e115cf0f66a645aca08225c9c1b20b80  
 X-Sign: 1bb7e4cd5c43f9885ba6a1758ad30fc562f88821

## Games

| Endpoint URL                      |
|-----------------------------------|
| /games                            |
| `[ GET / ]` Retrieving games list |

You will receive games collection available for your Merchant ID.

The production server only returns 50 games per page. Per page = 0 doesn't work there.

### Request fields

| Attribute: data type           | Description                                              |
|--------------------------------|----------------------------------------------------------|
| `expand`: `string`, `optional` | Request additional object expansions, separated by comma |

### Game item fields

| Attribute: data type                      | Description                                                                                                                                                                         |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| `uuid`: `string`                          | Game UUID that will be used in `/init` and `/lobby`                                                                                                                                 |
| `name`: `string`                          | Game name                                                                                                                                                                           |
| `image`: `string`                         | Game image url                                                                                                                                                                      |
| `type`: `string`                          | Game type                                                                                                                                                                           |
| `provider`: `string`                      | Game provider name                                                                                                                                                                  |
| `technology`: `string`                    | Game technology                                                                                                                                                                     |
| `has_lobby`: `integer`                    | 1 or 0 - indicates if game has lobby                                                                                                                                                |
| `is_mobile`: `integer`                    | 1 or 0 - indicates if game used for mobile devices and should be opened in new window (not in iframe or some <div> container)                                                       |
| `has_freespins`: `integer`                | 1 or 0 - indicates if game has freespins                                                                                                                                            |
| `has_tables`: `integer`                   | 1 or 0 - indicates if game has game tables                                                                                                                                          |
| `freewin_valid_until_full_day`: `integer` | 1 or 0 - indicates that `freewin/set` property `valid_until` must have time 00:00:00. Example pass valid_unit = 2020-01-25 freewin campaign will be valid until 2020-01-26 00:00:00 |

### Available expansions

| Attribute: data type   | Description                                              |
|------------------------|----------------------------------------------------------|
| `tags`: `object[]`     | assigned tags objects                                    |
| `parameters`: `object` | additional game parameters                               |
| `images`: `object[]`   | game images objects, including high-quality if available |

### Example

| Request                                |
|----------------------------------------|
| GET /games?expand=images,tags HTTP/1.1 |

### Lobby

If game has lobby integrator should call this action to get lobby tables, so player can choose which table to play.

#### Endpoint URL

/games/lobby

[ GET / ] Returns list of tables for the selected game

#### Request fields

| Attribute: data type               | Description                                            |
|------------------------------------|--------------------------------------------------------|
| `game_uuid`: `string`, `required`  | Game UUID provided in `/games`                         |
| `currency`: `string`, `required`   | Player currency that will be used in this game session |
| `technology`: `string`, `optional` | Parameter for lobby tables filtering by technology     |

Can be two types: "html5" or "flash".

#### Response fields

**lobby**: **array**, Contains lobby data of the selected game with following attributes:

| Attribute: data type     | Description                                                            |
|--------------------------|------------------------------------------------------------------------|
| `lobbyData`: `string`    | Data required on `/games/init` phase for <i>*lobby_data*</i> parameter |
| `name`: `string`         | Table name                                                             |
| `isOpen`: `boolean`      | True or false - indicates if game is open right now                    |
| `openTime`: `string`     | Lobby open time                                                        |
| `closeTime`: `string`    | Lobby close time                                                       |
| `dealerName`: `string`   | Dealer name                                                            |
| `dealerAvatar`: `string` | Dealer avatar url                                                      |
| `technology`: `string`   | Lobby technology ("html5" or "flash")                                  |
| `limits`: `array`        | Table limits                                                           |

#### Example

##### Request

GET /games/lobby?game\_uuid=abc123&currency=USD HTTP/1.1



## Response

```
HTTP/1.1 200 OK
...
{
 lobby: {
 lobbyData: "abcd12345",
 name: "Baccarat",
 isOpen: true,
 openTime: "11:00:00",
 closeTime: "12:00:00",
 dealerName: "abcd12345",
 dealerAvatar: "https://avatar-url.com",
 technology: "html5",
 limits: {
 {
 currency: "USD",
 min: 1,
 max: 100
 }
 }
 }
}
```

## Init

This action will prepare game for launch and return final url where player should be redirected to start playing.

## Endpoint URL

### Endpoint URL

</games/init>

`[ POST / ]` Initializing game session

## Request fields

| Attribute: data type                             | Description                                                       |
|--------------------------------------------------|-------------------------------------------------------------------|
| <code>`game_uuid`: `string`, `required`</code>   | Game UUID provided in <code>`/games`</code>                       |
| <code>`player_id`: `string`, `required`</code>   | Unique player ID on the integrator side                           |
| <code>`player_name`: `string`, `required`</code> | Player nickname that will be shown in some games                  |
| <code>`currency`: `string`, `required`</code>    | Player currency that will be used in this game session            |
| <code>`session_id`: `string`, `required`</code>  | Unique game session ID on the integrator side                     |
| <code>`return_url`: `string`, `optional`</code>  | Redirect player to this url after game ends                       |
| <code>`language`: `string`, `optional`</code>    | Player language                                                   |
| <code>`email`: `string`, `optional`</code>       | Player email                                                      |
| <code>`lobby_data`: `string`, `optional`</code>  | Used only for games with lobby. Provided in <code>`/lobby`</code> |

## Example

### Request

POST /games/init HTTP/1.1

game\_uuid=abcd12345&player\_id=abcd12345&player\_name=abcd12345&currency=USD& ....

## Response

HTTP/1.1 200 OK

```
{
 "url": "https://example.com/endpoint"
}
```

## Init demo game (only if provider has demo mode)

This action will prepare game for launch in demo mode and return final url where player should be redirected to start playing.

## Endpoint URL

/games/init-demo

[ POST / ] Initializing game session

## Request fields

| Attribute: data type               | Description                                 |
|------------------------------------|---------------------------------------------|
| `game_uuid`: `string`, `required`  | Game UUID provided in `/games`              |
| `return_url`: `string`, `optional` | Redirect player to this url after game ends |
| `language`: `string`, `optional`   | Player language                             |

## Example

### Request

POST /games/init-demo HTTP/1.1

...

game\_uuid=abcd12345&language=en&return\_url=....

### Response

HTTP/1.1 200 OK

```
{
 "url": "https://example.com/endpoint"
}
```

## Game launch

To launch the game redirect player to the URL returned by `/games/init` or `/games/init-demo`.

## Integrator

### Overview

Integrator should provide endpoint URL to communicate with Game Aggregator during the game session

Game Aggregator could send 4 type of calls to integrator

- Balance
- Win
- Bet
- Refund

## Request format

All calls from Game Aggregator to integrator will be done via `POST` and parameters will be passed with `application/x-www-form-urlencoded` content type

## Response format

All integrator responses should have `Content-Type: application/json` header, `json` format and `HTTP/1.1 200 OK` status code.

## Error format

In case of error integrator should return json object with following attributes and `HTTP/1.1 200 OK` status code.

| Attribute: data type                      | Description                                    |
|-------------------------------------------|------------------------------------------------|
| `error_code`: `string`, `required`        | Error code (specific for every action)         |
| `error_description`: `string`, `required` | Human readable error description. Can be empty |

### Response

```
HTTP/1.1 200 OK
{
 "error_code": "INSUFFICIENT_FUNDS",
 "error_description": "Not enough money to continue playing"
}
```

## Error codes

| Error code                        | Description                                                                                                                   |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <code>'INSUFFICIENT_FUNDS'</code> | code used in <b>bet</b> action when player has insufficient funds                                                             |
| <code>'INTERNAL_ERROR'</code>     | code used in all other cases meaning that action has not been executed: player not found, database or file system errors, etc |

## Security

All Game Aggregator requests contains authorization headers.

## Authorization headers

| Header name     | Description                                                                                               |
|-----------------|-----------------------------------------------------------------------------------------------------------|
| `X-Merchant-Id` | Merchant ID provided by integration manager                                                               |
| `X-Timestamp`   | Request timestamp. If differ from current timestamp for more than 30 seconds - request considered expired |
| `X-Nonce`       | Random string                                                                                             |
| `X-Sign`        | Sign calculated with sha1 hmac                                                                            |

## X-Sign calculation

1. Merge request array with authorization headers array
2. Sort resulting array by key in ascending order
3. Generate a URL-encoded query string from this array
4. Use sha1 hmac algorithm with Merchant Key (provided by integration manager) for signing

## PHP example of the X-Sign validation

```

```php
$merchantKey = 'Merchant Key provided by integration manager';

$headers = [
    'X-Merchant-Id' => 'Get header value',
    'X-Timestamp' => 'Get header value',
    'X-Nonce' => 'Get header value',
];

$XSign = 'Get header value';

$mergedParams = array_merge($_POST, $headers);
ksort($mergedParams);
$hashString = http_build_query($mergedParams);

$expectedSign = hash_hmac('sha1', $hashString, $merchantKey);

if ($XSign !== $expectedSign) {
    throw new \Exception('Invalid sign');
}

```

Example

Request

```

POST <client_callback_endpoint> HTTP/1.1
Content-Type: application/x-www-form-urlencoded
...
X-Merchant-Id: ff955b5759b3885f08cf125d4454ceb4
X-Timestamp: 1471857411
X-Nonce: e115cf0f66a645aca08225c9c1b20b80
X-Sign: 1bb7e4cd5c43f9885ba6a1758ad30fc562f88821

param=value&param2=value2
...

```

Balance

Game Aggregator will call this action to retrieve actual player balance

Endpoint URL

<client_callback_endpoint>

Request

`[POST /]` Balance request

Request fields

| Attribute: data type | Description |
|--------------------------------|-------------------------------------|
| `action`: `string` ["balance"] | Action "balance" |
| `player_id`: `string` | Unique player ID on integrator side |
| `currency`: `string` | Balance currency |
| `session_id`: `string` | Session ID (if option enabled) |

Response fields

| Attribute: data type | Description |
|---------------------------------|------------------|
| `balance`: `double`, `required` | Player's balance |

Example

Request

```
POST <client_callback_endpoint> HTTP/1.1
...
action=balance&player_id=123456&currency=USD&session_id=c4ca4238a0b923820dcc509a6f75849b
```

Response

```
HTTP/1.1 200 OK
...
{
  "balance": 57.12
}
```

Bet

This action is called when player trying to make a bet.

Bet types

| Type | Description |
|-------------|------------------|
| `bet` | Default bet type |
| `tip` | Tip for a dealer |
| `freespins` | Freespin |

Request fields

| Attribute: data type | Description |
|-----------------------------------|--|
| `action`: `string` ["bet"] | Action "bet" |
| `amount`: `double` | Bet amount |
| `currency`: `string` | Bet currency |
| `game_uuid`: `string` | Game UUID from the list of games `/games` |
| `player_id`: `string` | Unique player ID on integrator side |
| `transaction_id`: `string` | Unique transaction ID on Game Aggregator side |
| `session_id`: `string` | Unique integrator game session ID, provided in `/games/init` |
| `type`: `string` | "bet", "tip" or "freespins" |
| `freespins_id`: `string` | Unique campaign identifier provided in /freespins/set (present in case of active freespins campaign) |
| `quantity`: `int` | Number of freespins rounds left in campaign (present in case of active freespins campaign) |
| `round_id`: `string`, `optional` | id of current transaction round |
| `finished`: `boolean`, `optional` | is round is finished in game |

Response fields

| Attribute: data type | Description |
|--|--|
| `balance`: `double`, `required` | Player's balance after transaction |
| `transaction_id`: `string`, `required` | Unique transaction ID on the integrator side |

Important!

Bet with provided **transaction_id** should be processed only once. If you already processed this transaction, then return successful response with processed transaction ID on the integrator side.

Example

Request

```
POST <client_callback_endpoint> HTTP/1.1
...
action=bet&amount=10.00&currency=USD&transaction_id=abcd12345&session_id=abcd12345&type=bet
```

Response

```
HTTP/1.1 200 OK
...
{
  "balance": 27.18,
  "transaction_id": "abcd12345",
}
```

Win

Action called when player win in a game

Win types

| Type | Description |
|------------|----------------------|
| `win` | Default win type |
| `jackpot` | Player get a jackpot |
| `fre spin` | Freespin |

Request fields

| Attribute: data type | Description |
|----------------------------|---|
| `action`: `string` ["win"] | Action "win" |
| `amount`: `double` | Win amount |
| `currency`: `string` | Win currency |
| `game_uuid`: `string` | Game UUID from the list of games <code>/games`</code> |
| `player_id`: `string` | Unique player ID on integrator side |
| `transaction_id`: `string` | Unique transaction ID on Game Aggregator side |
| `session_id`: `string` | Unique integrator game session ID, provided in <code>/games/init`</code> |
| `type`: `string` | "win", "jackpot" or "fre spin". For some Pragmatic provider games also can be "bonus", "pragmatic_prize_drop" or "pragmatic_tournament" |
| `fre spin_id`: `string` | Unique campaign identifier provided in <code>/fre spins/set`</code> (present in case of active fre spin campaign) |
| `quantity`: `int` | Number of fre spin rounds left in campaign (present in case of active fre spin campaign) |

| | |
|--|---------------------------------|
| <code>'round_id': 'string', 'optional'</code> | id of current transaction round |
| <code>'finished': 'boolean', 'optional'</code> | is round is finished in game |

Response fields

| Type | Description |
|---|--|
| <code>'balance': 'double', 'required'</code> | Player's balance after transaction |
| <code>'transaction_id': 'string', 'required'</code> | Unique transaction ID on the integrator side |

Important!

Win with provided **transaction_id** should be processed only once. If you already processed this transaction, then return successful response with processed transaction ID on the integrator side.

We don't provide **round_id** value for freespins wins of the ELK provider.

Example

Request

```
POST <client_callback_endpoint> HTTP/1.1
...
action=win&amount=100.00&currency=USD&transaction_id=abcd12345&session_id=abcd12345&type=win
```

Response

```
HTTP/1.1 200 OK
...
{
  "balance": 170.21,
  "transaction_id": "abcd12345",
}
```

Refund

Refund is a cash back in case bet problems.

After receiving `'refund'` call integrator should cancel corresponding bet transaction and return funds to player. If such bet transaction does not exist then integrator should just save this refund transaction and respond with success.

Request fields

| Attribute: data type | Description |
|---|---|
| <code>'action': 'string ["refund"]'</code> | Action "refund" |
| <code>'amount': 'double'</code> | Refund amount |
| <code>'currency': 'string'</code> | Refund currency |
| <code>'game_uuid': 'string'</code> | Game UUID from the list of games <code>"/games"</code> |
| <code>'player_id': 'string'</code> | Unique player ID on integrator side |
| <code>'transaction_id': 'string'</code> | Unique transaction ID on Game Aggregator side |
| <code>'session_id': 'string'</code> | Unique integrator game session ID, provided in <code>"/games/init"</code> |
| <code>'type': 'string', 'optional'</code> | Transaction type. Available values: <code>'bet'</code> , <code>'tip'</code> , <code>'fre spins'</code> |
| <code>'bet_transaction_id': 'string'</code> | Game Aggregator bet transaction ID to be refunded |
| <code>'fre spins_id': 'string'</code> | Unique campaign identifier provided in <code>/fre spins/set</code> (present in case of active fre spins campaign) |

| | |
|-----------------------------------|--|
| 'quantity': 'int' | Number of freespins rounds left in campaign (present in case of active freespins campaign) |
| 'round_id': 'string', 'optional' | ID of current transaction round |
| 'finished': 'boolean', 'optional' | is round is finished in game |

Response fields

| Attribute: data type | Description |
|--|---|
| 'balance': 'double', 'required' | Player's balance after transaction |
| 'transaction_id': 'string', 'required' | Unique refund transaction ID on the integrator side |

Important!

Bet with provided **bet_transaction_id** should be refunded processed only once. If you already refunded this transaction, then in response return processed refund transaction ID on the integrator side.

Request

```
POST <client_callback_endpoint> HTTP/1.1
...
action=refund&amount=10.
00&currency=USD&transaction_id=abcd12345&session_id=abcd12345&bet_transaction_id=abcd1234&type=bet
```

Response

```
HTTP/1.1 200 OK

{
  "balance": 27.18,
  "transaction_id": "abcd12345",
}
```

Rollback

In case enabled **only for two** providers

Rollback is a cancel of the whole round or part of the session if provider does not support rounds.

After receiving 'rollback' call integrator should cancel corresponding bet, refund and win transactions and actualize player balance. If such bet or win transaction does not exist then integrator should just save this transaction as 'rollbacked' and respond with success.

The integrator should cancel transactions from **the 'rollback_transactions' list only**. Any additional logic based on 'provider_round_id', that will rollback other transactions in the same round, can produce a lot of errors within out of sync session data between provider, game aggregator and integrator.

Request fields

| Attribute: data type | Description |
|----------------------------------|---|
| 'action': 'string ["rollback"]' | Action "rollback" |
| 'currency': 'string' | Rollback currency |
| 'game_uuid': 'string' | Game UUID from the list of games '/games' |
| 'player_id': 'string' | Unique player ID on integrator side |
| 'transaction_id': 'string' | Unique transaction ID on Game Aggregator side |
| 'rollback_transactions': 'array' | List of round transactions |

| | |
|-------------------------------|---|
| `action`: `string` | `bet`, `win` or `refund` action |
| `amount`: `double` | Transaction amount |
| `transaction_id`: `string` | Unique transaction ID on Game Aggregator side for rollbacking transaction |
| `type`: `string` | See types for `bet` or `win` |
| `session_id`: `string` | Unique integrator game session ID, provided in `/games/init` |
| `type`: `string` | "rollback" |
| `provider_round_id`: `string` | Game Aggregator round id |
| `round_id`: `string` | Game Aggregator round id |

Response fields

| Attribute: data type | Description |
|--|--|
| `balance`: `double`, `required` | Player's balance after transaction |
| `transaction_id`: `string`, `required` | Unique rollback transaction ID on the integrator side |
| `rollback_transactions`: `array`, `required` | All transactions id related to rollbacked round should be in array |

If some transaction will be missed Game Aggregator will accept rollback response as failed.



All transactions specified in rollback request should be refunded processed only once. If you already processed some transaction, then transaction_id should be in response as successfully processed.

Example

Request

POST <client_callback_endpoint> HTTP/1.1
...

```
action=rollback&currency=EUR&game_uuid=12345&player_id=4&transaction_id=12345&rollback_transactions[0][action]=bet&rollback_transactions[0][amount]=1.5&rollback_transactions[0][transaction_id]=12346&rollback_transactions[0][type]=bet&rollback_transactions[1][action]=win&rollback_transactions[1][amount]=0&rollback_transactions[1][transaction_id]=12347&rollback_transactions[1][type]=win&session_id=23456&type=rollback&provider_round_id=1
```

Response

HTTP/1.1 200 OK
...

```
{
  "balance": 27.18,
  "transaction_id": "12345",
  "rollback_transactions" => {
    12346,
    12347
  }
}
```

Additional requests to Game Aggregator

Merchant limits

Returns list of limits for merchant

Endpoint URL

/limits

Request

[GET /] Returns list of limits for merchant

Response fields

| Attribute: data type | Description |
|-------------------------|--|
| `amount`: `string` | Amount left |
| `currency`: `string` | Limit currency |
| `provider`: `array` | List of providers attached to this limit |

Example**Response**

GET /limits
...

Response

```
HTTP/1.1 200 OK
...
[
  {
    "amount": "1000.00",
    "currency": "USD",
    "providers": [
      "Provider1",
      "Provider2",
      "Provider3"
    ]
  },
  {
    "amount": "1000.00",
    "currency": "EUR",
    "providers": [
      "Provider1"
    ]
  }
]
```

Merchant freespins limits

Returns list of freespins limits for merchant.

Endpoint URL**Endpoint URL**

/limits/freespin

Request

`[GET /]` Returns list of freespins limits for merchant

Response fields

| Attribute: data type | Description |
|----------------------|--|
| `quantity`: `int` | Quantity of freespins left |
| `currency`: `string` | Freespin limit currency |
| `provider`: `array` | List of providers attached to this freespins limit |

Example

Request

GET /limits/freespin

...

Response

HTTP/1.1 200 OK

...

```
[
  {
    "quantity": 17,
    "currency": "USD",
    "providers": [
      "Provider1",
      "Provider2",
      "Provider3"
    ]
  },
  {
    "quantity": 1000,
    "currency": "EUR",
    "providers": [
      "Provider1"
    ]
  }
]
```

List of jackpots

Returns list of jackpots for every game provider (if available) assigned to merchant key.
List of jackpots is cached for 60 seconds.

Endpoint URL

/jackpots

Request

`[GET /]` Returns list of jackpots assigned to merchant key

Response fields

| Attribute: data type | Description |
|----------------------|-------------|
|----------------------|-------------|

| | |
|--------------------------|---|
| `name`: `string`, `null` | Jackpot name (string or null if game provider does not have names for jackpots) |
| `amount`: `string` | Amount left |
| `currency`: `string` | Limit currency |
| `provider`: `string` | Game provider |

Example

Request

GET /jackpots
...

Response

```
HTTP/1.1 200 OK
...
[
  {
    "name": "jackpot name",
    "amount": "1000.00",
    "currency": "USD",
    "provider": "Provider1"
  },
  {
    "name": null,
    "amount": "1000.00",
    "currency": "EUR",
    "provider": "Provider2"
  }
]
```

Balance notification

Notify every game provider (if available) assigned to merchant key about balance changes.

Endpoint URL

/balance/notify

[POST /] Notify that player's balance was changed

| Attribute: data type | Description |
|---------------------------------|--|
| `balance`: `double`, `required` | Updated player balance |
| `session_id`: `string` | Unique integrator game session ID, provided in `/games/init` |

Example

Request

POST /balance/notify HTTP/1.1
...
balance=11.23&session_id=23456

Response

```
HTTP/1.1 200 OK
...
...
...
HTTP/1.1 500 Internal Server Error
...
{
  "name": "Internal Server Error",
  "message": "Session related to casino_session_uuid was not found",
  "code": 0,
  "status": 500
}
```

List of available freespins bets for chosen game and currency

Endpoint URL

/freespins/bets

[GET /]` Get list of available freespins bets for chosen game and currency

Request fields

| Attribute: data type | Description |
|-----------------------------------|---|
| `game_uuid`: `string`, `required` | Game UUID provided in `/games` |
| `currency`: `string`, `required` | Player currency that will be used in freespins campaign |

Response fields

| Attribute: data type | Description |
|--------------------------|--------------------------------------|
| `denominations`: `array` | Available denominations |
| `bets`: `array` | Available freespins bets, optional |
| `total_bets`: `array` | Possible total bets values, optional |

Bets description

| field name | type | description |
|--------------|---------------|--|
| bet_id | string | id of bet in list |
| bet_per_line | string /float | if it is float it means bet amount for one line. Or it can be one of the next values: "max", "mid", "min". It means that final bet amount determined by provider |
| lines | integer | lines count of game |

Total bets description

| field name | type | description |
|------------|---------|--------------------------------------|
| bet_id | integer | id of bet in total_bets list |
| amount | float | free spins total bet amount per spin |

Example

Request:

Request

GET /freespins/bets?game_uuid=abcd12345¤cy=USD HTTP/1.1

...

Response:

Response

HTTP/1.1 200 OK

```
...
{
  "denominations":["0.01","0.1","1"],
  "bets":[
    {
      "bet_id":"0",
      "bet_per_line":1,
      "lines":25
    },
    {
      "bet_id":"1",
      "bet_per_line":2,
      "lines":25
    },
    ...
  ],
  "total_bets":[
    {
      "bet_id": 0,
      "amount": 10.0
    },
    {
      "bet_id": 1,
      "amount": 25.0
    },
    ....
  ]
}
```

Set a freespins campaign

Endpoint URL

/freespins/set

Request

`[POST /]` Set a freespins campaign for player

Request fields

| Attribute: data type | Description |
|-------------------------------------|--|
| `player_id`: `string`, `required` | Unique player ID on the integrator side |
| `player_name`: `string`, `required` | Player nickname that will be shown in some games |
| `currency`: `string`, `required` | Player currency that will be used in this freespins campaign |
| `quantity`: `int`, `required` | Number of freespins rounds in this campaign |
| `valid_from`: `int`, `required` | Start date (Timestamp) of campaign |

| | |
|--|--|
| <code>`valid_until`: `int`, `required`</code> | End date (Timestamp) of campaing. Also see description in method <code>`/games`</code> , property <code>`fre spin_valid_until_full_day`</code> |
| <code>`fre spin_id`: `string`, `required`</code> | Unique identifier of campaing |
| <code>`bet_id`: `integer`, `optional`</code> | Bet ID provided in <code>`/fre spins/bets`</code> |
| <code>`total_bet_id`: `integer`, `optional`</code> | Total bet ID |
| <code>`denomination`: `double`, `optional`</code> | Denomination provided in <code>`/fre spins/bets`</code> , <code>`required`</code> if <code>`bet_id`</code> used |
| <code>`game_uuid` : `string`, `required`</code> | Game UUID provided in <code>`/games`</code> that will be included in campaing |

One of optional fields ``bet_id`` and ``denomination`` or ``total_bet_id`` is required.

Example

| Request |
|--|
| POST <code>/fre spins/set</code> HTTP/1.1 ... player_id=abcd12345&player_name=abcd12345¤cy=USD&quantity=5&valid_from=1518610000&valid_until=1519610000& |

| Response |
|------------------------|
| HTTP/1.1 200 OK ... |

Get a fre spin campaign

| Endpoint URL |
|---|
| <code>/fre spins/get</code> |
| Request |
| <code>` [GET /]`</code> Get list of set campaigns |

Request fields

| Attribute: data type | Description |
|--|-------------------------------|
| <code>`fre spin_id`: `string`, `required`</code> | Unique identifier of campaing |

Response fields

| Attribute: data type | Description |
|-------------------------------------|---|
| <code>`player_id`: `string`</code> | Unique player ID on the integrator side |
| <code>`currency`: `string`</code> | Player currency that will be used in this fre spin campaign |
| <code>`quantity`: `int`</code> | Number of fre spin rounds in this campaing |
| <code>`quantity_left`: `int`</code> | Number of fre spin rounds left in this campaing |

| | |
|--------------------------|--|
| `valid_from`: `int` | Start date (Timestamp) of campaign |
| `valid_until`: `int` | End date (Timestamp) of campaign |
| `freespins_id`: `string` | Unique identifier of campaign |
| `bet_id`: `int` | Bet ID provided in `/freespins/bets` |
| `total_bet_id`: `int` | Total bet ID |
| `denomination`: `double` | Denomination provided in `/freespins/bets` |
| `game_uuid`: `string` | Game UUID provided in `/games` |
| `status`: `string` | Status of campaign |
| `is_canceled`: `int` | is campaign canceled |
| `total_win`: `double` | Total win |

Example

Request

```
GET /freespins/get?freespins_id=abcd12345
...
```

Response

```
HTTP/1.1 200 OK
...
{
  "player_id": "abcd12345",
  "currency": "USD",
  "quantity": 10,
  "quantity_left": 8,
  "freespins_id": "abcd12345",
  ...
}
```

Cancel set camping

Endpoint URL

```
/freespins/cancel
```

`[POST /]` Cancel set camping for player

Request fields

| Attribute: data type | Description |
|--------------------------------------|-------------------------------|
| `freespins_id`: `string`, `required` | Unique identifier of campaign |

Example

Request

```
POST /freespins/cancel HTTP/1.1
...
freespins_id=abcd12345
```


Response

HTTP/1.1 200 OK
...

Integrator self validation

Integrator could check if implementation on his side is correct. To start validation integrator should have active game session (opened not longer than 15 minutes ago). Game Aggregator will send set of requests ('bet', 'win', etc) during validation and return result in response.

Endpoint URL

/self-validate

`[POST /]` Self validation

Response fields

| Attribute: data type | Description |
|----------------------|---|
| `success`: `boolean` | true or false - indicates if validation is passed and implementation is correct |
| `log`: `array` | Validation log |

Example

Request

POST /self-validate
...

Response

HTTP/1.1 200 OK
...
{
 "success": true,
 "log": [
 "Log message",
 "Log message",
 ...
]
}