# DRAGONGAMING

## CONNECT API DOCUMENTATION

## SEAMLESS WALLET

# Contents.

# Introduction.

As a game provider, our objective is to deliver exceptional games and experiences to our customers and their players. We would like to see this translate into you integrating our games onto your casino lobby.

This document provides an insight into our API, known as “Chronos”. It addresses the key interactions and requirements from both DragonGaming™ and your perspective. It provides coverage over a wide range of services which can be used by you on your websites, game-servers, and any other API services.

You are required to follow a standard integration process; however, this document will provide a brief overview on several key functions required of our API. This includes integration to Chronos API, API request details, server-to-server integration, game integration and wallet transactions.

We have a separate integration guide that explains the overall integration spec and also includes all the various tools, currencies and languages that we support.

# Glossary.

| Terms | Description |
|---|---|
| Operator | Anyone that owns / operates a website / frontend. |
| RGS | Remote Game Server (by provider) |
| Provider | DragonGaming™ / Gaming software provider |
| NG-CONNECT | DragonGaming™ API. RGS-API |
| Session ID / Token | Session ID of the player |
| API Key | Unique key provided by DragonGaming to each operator. |
| Frontend / Site / White-label / Platform | Operator’s website in which games are displayed. |
| Game Lobby | Area where games are shown on operator’s website. |

*The following steps will aid in the integration between DragonGaming and new operators.*

# Authentication.

- Each API user must obtain a unique API key
- Each request must contain the API key so each API request can be validated.

# API URLs + API Keys.

- Any API URL points, and API Keys items will be shared separately with operators.

# Request and Response.

- HTTP Header “Content-Type”: ”application/json”
- Data sent in POST request should be in JSON format
- All the API responses are in JSON format

## Server to Server Communication.

Server-level communications between provider and operator would be conducted in JSON format and is transmitted over HTTP. The current default allows one request at any one time; however, simultaneous requests are possible if required. By default, all currency values are provided in cents with no decimal values. In exceptional cases, these can be provided in a base currency (Pounds/Euros).

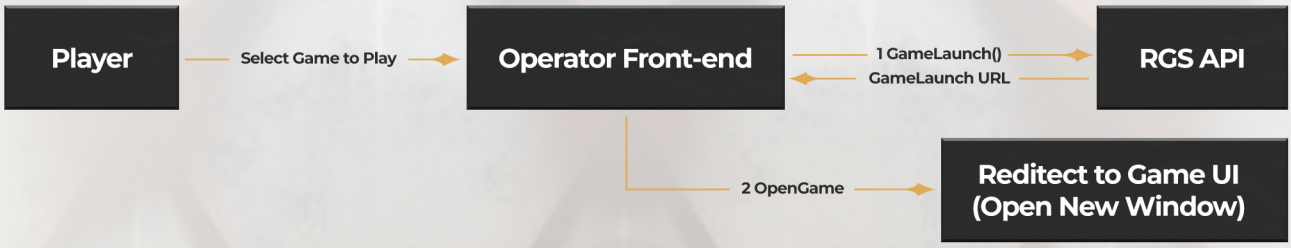
## API Requests Flow .

*Following would be the typical flow of few of the API requests.*

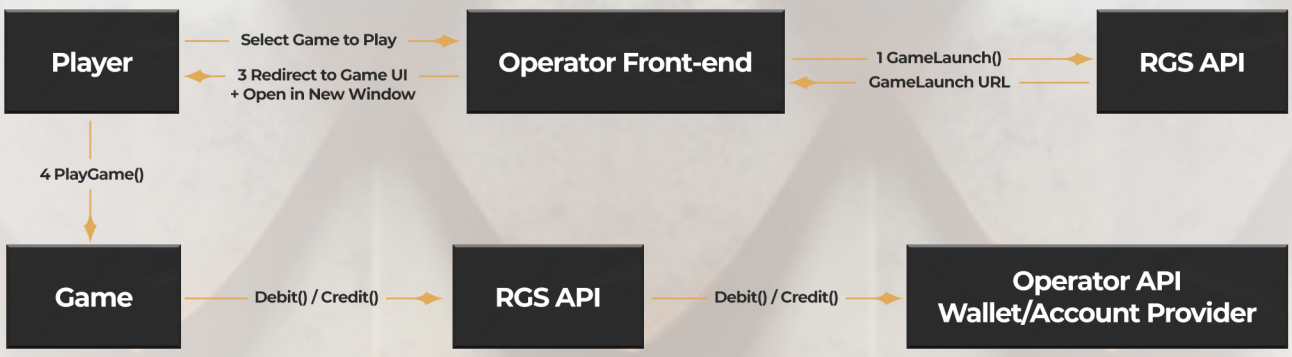
- CasinoSite (front-end of the Operator) is the example website name.
- Player A visits and logins into CasinoSite by providing Username and Password.
- CasinoSite verifies Username and Password match and performs typical checks. If everything is fine, a unique session ID (token) is created for the player in CasinoSite platform.
- Player A visits the game lobby.
- CasinoSite makes the API call GetGames() to Chronos to get the list of games enabled for the CasinoSite. Chronos replies with the list of games enabled.
- List of games are shown to the player in game lobby.
- Player A clicks on any of DragonGaming™ games. CasinoSite needs to make GameLaunch() API call to Chronos API by providing the required details as part of API request.
- Chronos registers the session ID and creates player details if not present and ignores if already available in RGS platform and returns game launch URL to CasinoSite.
- Game is launched (in a new window or redirected to the URL) in CasinoSite with the given GameLaunch URL sent by Chronos.
- Player A plays the game which results in debits and credits. RGS game engine makes debit and credit calls to Chronos accordingly.
- The API Chronos, realizing that the player's wallet is hosted in CasinoSite platform, redirects the calls to the CasinoSite API platform. CasinoSites designs and implements this API and performs the operations on their account wallets. In return, CasionSite sends player's balance details post debit/credit.
- During every call, API_KEY is passed to Chronos by CasinoSite API.
- During all these calls, the session ID is passed to Chronos by CasinoSite and same session ID will be sent back to CasinoSite during player transactions API calls. CasinoSite recognizes the provided session ID and performs the operations on their wallet.

## GetGames() Flow

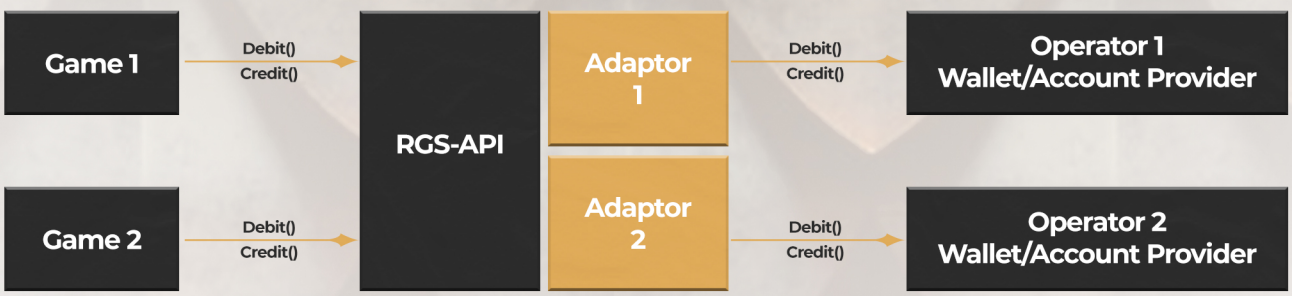

## GetLaunch() Flow




## Wallet Typical Flow




## Debit() & Credit() Flow



# API Requests from Operator to Provider.

The following examples are some API requests that are currently provided to operators and third parties. This non-exhaustive list is provided as a brief overview. The number and type of requests can increase, as per an operator's requirements.

## GetGames() API.

This API request allows the operator to receive a full list of our games which are enabled for their particular front-end. If no games are enabled, the response would return as “empty”.

### GetGames() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/games/get-games/ |
| Headers | “Content-Type”:”application/json” |
| POST Parameters | `json object e.g.:`<br>`{`<br>`"api_key": "1pghjl4v5apt2bks"`<br>`}` |

| Parameters | Type | Description |
|---|---|---|
| api_key | String | API Key allocation to each frontend |

### GetGames Request Sample

```
{
  "api_key":"1pghjl4v5apt2bks"
}
```

## Get Games Response Example

Response would be in JSON format. Type: dictionary. Dictionary contains a root level key result and game details as value. Data in the response is self-explanatory.


```
{
    "result": {
        "games": {
            "slots": [
                {
                    "game-fruityfeast": {
                        "game_id": 4,
                        "game_name": "fruityfeast",
                        "game_title": "Fruity Feast",
                        "category": "slots",
                        "supplier": "test_games",
                        "story": "Reap the Fruity Rewards",
                        "logos": [
                            {
                                "url": "https://test-gam-
ing.com/images/lobby/200x150/fruityfeast.png",
                                "width": "200",
                                "height": "150"
                            },
                            {
                                "url": "https://test-gam-
ing.com/images/lobby/400x300/fruityfeast.png",
                                "width": "400",
                                "height": "300"
                            }
                        ],
                        "launch_params": [
                            {
                                "width": 800,
                                "height": 600,
                                "resizable": true,
                                "scrollbars": false,
                                "type": "browser",
                                "launch_type": "new_window",
                                "window_title": "%full_site_code%_%category%",
                                "channel": "desktop",
                                "launch_url": "https://test-
games.testsite.com/game_launcher.php?
session_id=%session_id%&channel=desktop&full_site_code=PFMNGOP&language=en&
game_name=%game_i d%&category=slots&amount_
type=%amount_type%&reality_check=%reality_check%"
                            },
                            {
                                "type": "browser",
                                "launch_type": "same_window",
                                "channel": "mobile",
                                "launch_url": "https://test-
games.testsite.com/game_launcher.php?
session_id=%session_id%&channel=mobile&
full_site_ code=PFMNGOP&language=en&game_name=%game_id%&
category=slots&amount_type=%amount_type%&reality_check=%reality_check%"
                            }
                        ],
                        "amount_types": [
                            {
                                "id": "cash",
                                "value": 1,
                                "name": "Play"
                            },
                            {
                                "id": "bonus",
                                "value": "2",
                                "name": "Play"
                            }
                        ]
                    }

                },
                {
                    "game2": "game_details"
                }
            ],
            "table_games": [],
            "scratch_cards": []
        }
    }
}
```


| Parameters | Type | Description |
|---|---|---|
| result | JSON Object | Key values would be displayed as a dictionary. |
| games | JSON Object | Presented as a key inside the dictionary. |
| game_type | JSON Object | Inside each game is a list containing game specific details in dictionary form. |

### GetGames() Error Response

Any response would be provided in JSON format. Display Type: dictionary. The dictionary contains two keys names “error” and “error_details”. If an error occurs, the error key will contain the value 1 and the error_details key would contain the information detailing that specific error. Any displayed messages would be setup as appropriate to each error.

```
{
  "api_key":"1"
  "error_detail": {
      "id":"1001",
      "code":"INVALID_API_KEY",
      "message":"Invalid API key."
}
```

## GameLaunch() API.

Operator needs to make GameLaunch() API request to RGS API to get game launch URL. Though game launch URL would have been sent already in GetGames() API call response, it allows RGS API to register the session, builds the launch URL and sends it back to the caller.

### GameLaunch() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/games/game-launch/ |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.:<br>{<br>"api_key": "1pghjl4v5apt2bks",<br>"session_id": "b6ef88a8054e328f4459b625baf38fae71981a34",<br>"provider": "dragongaming",<br>"game_type": "slots",<br>"game_id": "5",<br>"platform": "desktop",<br>"language": "en",<br>"amount_type": "real",<br>"lobby_url": "",<br>"deposit_url": "",<br>"context": {<br>"id": "1232",<br>"username": "username",<br>"country": "GB",<br>"currency": "GBP"<br>}<br>} |

| Parameters | Type | Description |
|---|---|---|
| api_key | String | API Key allocation to each frontend |
| session_id | String | Session ID created for the player during login |
| provider | String | Game Provider. DragonGaming™ here. |
| game_type | String | Type of Game (eg: slots, table_games, scratch_cards) |
| game_id | String | Game ID provided by the provider |
| platform | String | Channel in which player is playing the game. Ex: desktop, mobile |
| language | String | Language the game will be played in |
| amount_type | String | **Mode of Play**<br>For Example:<br>'real' – when playing with real cash<br>'fun' – when playing fun mode<br>'promo_freespin' – when playing promotions |
| lobby_url | String | Home or Lobby URL of the operator |
| deposit_url | String | Deposit URL of the operator |
| context | JSON Object | JSON object containing player details such as first name, last name, username, country, gender, agent_id** etc. |

***Note:***

** When amount type is **fun**, please send random session_id, 0 as player_id, 'fun_player' as username in context as we have uniform API for game launch.*

*** Agent_id parameter is optional. However, once the agent_id is given for a particular player for the first time it will be recorded in our database along side with player id and it will remain the same even if a different ID is being sent by operator.*

## GameLaunch() Response Example

Response would be in JSON format. Type: *dictionary*.

```
{
    "launch_url": "https://staging-games.dragongaming.com/game_launcher.php?
session_id=b6ef88a8054e328f4459b625baf38fae71981a34&channel=desktop&full_site_code=PF
MNGOP&language=en&game_name=twindragons&category=slots&amount_type=1&reality_check=12
0&lobby_url=<lobby_url>&deposit_url=<deposit_url>"
}
```

| Object / Key in response object | Type | Description |
|---|---|---|
| launch_url | JSON Object | Key launch_url contains the Game Launch URL as value that is used to launch the game. |

## GameHistory() API.

This API request enables the operator to get history of recent 100 game rounds played with real cash mode by default.

### GameHistory() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/games/game-history/ |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.: {<br>"api_key": "1pghjl4v5apt2bks",<br>"player_id": 1002,<br>"amount_type": "real",<br>"start_date": "2020-02-20 00:00:00",<br>"end_date": "2020-02-21 00:00:00",<br>"page_num": 1,<br>} |

| Parameters | Type | Description |
|---|---|---|
| api_key | String | API Key allocated to each frontend |
| player_id | String | ID of the player |
| amount_type | String | Mode using which game was played |
| start_date | String | End date time in the following format “YYYY-MM-DD HH:MM:SS” |
| end_date | String | End date time in the following format |
| player_id (optional) | String | Unique ID of player. If this paramtere sent in request, then response will be return in player specific. |

## GameHistory() Response Example

Response would be in JSON format. Type: *dictionary*.

```
{
    "game_history": {
        "headers": [
            "Game Name",
            "Round ID",
            "Bet Amount", "Win
                 Amount", "Amount Type", "Date Time"],
                "data": [
                [
                    "Test Game Name-1",
                    1001,
                    100,
                    100,
                    "real",
                    "2019-07-23 17:20:57"
                ],
                [
                    "Test Game Name-1",
                    1002,
                    100,
                    100,
                    "real",
                    "2019-07-23 17:20:57"
                ],
                [
                    "Test Game Name-2",
                    1003,
                    100,
                    100,
                    "real",
                    "2019-07-23 17:20:57"
                ]
            ]
        }
    }
```

| Object / Key in response object | Type | Description |
|---|---|---|
| game_history | JSON Object | Response status |
| game_history['headers'] | JSON List | Table headers |
| game_history['data'] | JSON List of Lists | List of Lists of game rounds data |

# UnfinishedGames() API.

This API request enables the operator to get unfinished games list of last 30 days.

## UnfinishedGames() request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/games/unfinished-games/ |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.: {<br>"api_key": "1pghjl4v5apt2bks",<br>"player_id": "1",<br>"game_type": "slots",<br>"amount_type": "real"<br>} |

| Parameters | Type | Description |
|---|---|---|
| api_key | String | API Key allocated to each frontend |
| player_id | String | Player ID of the player known to the Operator |
| game_type | String | Type of Game (eg: slots, table_games, scratch_-cards) |
| amount_type | String | Mode of Play For Example: 'real' – when playing with real cash 'freespins' – when playing with promo_freespin |

## UnfinishedGames() Response Example

Response would be in JSON format. Type: dictionary.


```
{
    "game_history": {
        "freespins": {
            "headers": [
                "Game Name",
                "Round ID",
                "Amount Type",
                "Num Spins",
                "Spins left",
                "Date Time"
            ],
            "data": [
                [
                    "Test Game Name-1",
                    1001,
                    "real",
                    10,
                    10,
                    "2019-07-23 17:20:57"
                ],
                [
                    "Test Game Name-1",
                    1002,
                    "real",
                    "2019-07-23 17:20:57"
                ]
            ]
        }
    },
    {
        "bonus_games": {
            "headers": [
                "Game Name",
                "Round ID",
                "Amount type",
                "Num Picks",
                "Num of User Picks",
                "Date Time"
            ],
            "data": [
                [
                    "Test Game Name-1",
                    1201,
                    "real",
                    5,
                    0,
                    "2019-07-23 17:20:57"
                ],
                [
                    "Test Game Name-1",
                    1202,
                    "real",
                    1,
                    0,
                    "2019-07-23 17:20:57"
                ],
                [
                    "Test Game Name-2",
                    1303,
                    "real",
                    4,
                    3,
                    "2019-07-23 17:20:57"
                ]
            ]
        }
    }
}
```

| Object / Key in response object | Type | Description |
|---|---|---|
| game_history | JSON Object | Key game_history contains the JSON object as value. Contains 2 keys 'freespins' and 'bonus_games' |
| game_history ['freespins'] | JSON Object | JSON Object with 2 keys freespins and data. |
| headers | JSON List | List of table headers |
| Game Name | String | Name of the game |
| Round ID | Integer | Round ID which triggered free-spins |
| Amount Type | String | Mode using which game was played |
| Num Spins | Integer | Number of spins initially awarded |
| Spins Left | Integer | Number of remaining spins |
| Date Time | String | Server UTC time game played at |
| data | JSON List of Lists | List of Lists of freespins. |
| game_history ['bonus_games'] | JSON Object | JSON Object with 2 keys bonus_games and data. |
| headers | JSON List | List of table headers |
| Game Name | String | Name of the game |
| Round ID | Integer | Round ID which triggered free-spins |
| Amount Type | String | Mode using which game was played |
| Num Picks | Integer | Number of picks initially awarded |
| Num of User Picks | Integer | Number of remaining picks |
| Date Time | String | Server UTC time game played at |
| data | JSON List of Lists | List of Lists of bonus games |

# GrantBonus()API.

This API request enables the operator to grant bonus free spins (promotional free spins) to the player

## GrantBonus() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/wallet/grant-bonus/ |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.: {<br>"api_key": "15OY0Rx1BC0hYbCm",<br>"amount_type": "promo_freespin",<br>"campaign_id": "BOn-3",<br>"game_ids": [<br>1032,<br>1041<br>],<br>"coin_value_level": 1,<br>"num_rounds": 10,<br>"player_ids": [<br>"267"<br>],<br>"currencies": [<br>"USD"<br>],<br>"start_date": "2021-01-19 14:00:00",<br>"end_date": "2021-01-30 15:00:00",<br>"max_win_limit":0,<br>} |

| Parameters | Type | Description |
|---|---|---|
| api_key | String | API Key allocated to each front-end |
| amount_type | String | **Mode of the play.** Ex: 'promo_freespin' – It is when the player is playing with promo freespins |
| campaign_id | String | It should be unique campaign id |
| game_idstransaction_id | List | List of game_id for which promo free spins are-granted. Ex: game_ids: [1,10,6,3] |
| coin_value_level | Integer | The value must be an integer and must be from allowed list of coin values levels. Before you start with promo freespins, we will provide the list of allowed coin value levels. Ex: coin_values:[10, 10, 20, 30] |
| num_rounds | Integer | It indicates the number of free spin rounds that is granted to the player. |
| player_ids | List | Player ID of the player known to the operator. Free spins are granted to all these player IDs mentioned in this list and for all the games mentioned in the game_ids list. |
| currencies | List | Currency of the player and it should match with the player currency which is already registered with us. Currencies count should match with the players count. Ex:- ["EUR", "USD"] |
| start_date | String (Optional) | Date time from which this bonus can be used. It should not be less than are equal to current time format = "YYYY-MM-DD HH:MM:SS" |
| end_date | String | Expire date time of the promotion. It cannot be less than the start_date format = "YYYY-MM-DD HH:MM:SS" |
| max_win_limit (optional) | Integer | This parameter is used to cap the winnings from promo free spins. We need to provide this value while granting free spins. It's value is in cents. It should be 0 when there is no cap on the winnings.<br><br>For example, Player with USD currency is awarded with 10 free spins with 100 cents as cap amount. If the winnings are USD 20 from these 10 rounds, as per the cap amount, only 1 USD is credited to the player wallet. If the cap value is set a 0, entire win amount of USD 20 is credited to player wallet. |

## GrantBonus() API Additional Notes

- Combination of bonus_id, player_id and game_id is always unique per operator/aggregator.
- Coin value levels are explained below. Below are the coin values per game and per currency for ref-erence. Actual coin values will be provided during the integration.

| | Game Name → | Game Name 1 | | | | |
|---|---|---|---|---|---|---|
| | Coin Levels → | 1 | 2 | 3 | 4 | 5 |
| **Country** | **Currencies** | **Coin values** | | | | |
| European Union | EUR | 1 | 2 | 3 | 4 | 6 |
| Argentina | ARS | 30 | 60 | 90 | 120 | 180 |
| Australia | AU | 1 | 2 | 3 | 4 | 6 |
| Brazil | BRL | 5 | 10 | 15 | 20 | 25 |
| Bulgaria | BGN | 1 | 2 | 3 | 4 | 5 |
| Canada | CAD | 1 | 2 | 3 | 4 | 5 |
| China | CNY | 10 | 20 | 30 | 40 | 50 |

Example: For Game-1 if the coin value level 3 is chosen, coin value for EUR player would be 3, coin value for ARS player would be 90, coin value for BRL would be 15. These are the coin values that will be used during the promo free spins play.

## GrantBonus() Response Example

Response would be in JSON format. Type: *dictionary*.

```
{
    "result": {
        "status": "success",
        "promo_freespin_id": 131
    }
}
```

| Object / Key in response object | Type | Description |
|---|---|---|
| result | JSON Object | Key result contains the JSON object as value. |
| status | String | Response status |
| promo_freespins_id | Integer | This is providers ID. |

# BonusList() API.

This API request enables the operator to get the list already awarded bonuses.

## BonusList() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/wallet/bonus-list/ |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.:<br><br>{<br>"api_key": "15OY0Rx1BC0hYbCm",<br>"start_date":"2021-01-19 14:00:00",<br>"end_date":"2021-01-30 15:00:00",<br><br>"limit_level":1<br>} |

| Parameters | Type | Description |
|---|---|---|
| api_key | String | API Key allocated to each frontend |
| start_date | String | Date time from which this bonus can be used. format = “YYYY-MM-DD HH:MM:SS” |
| end_date | String | Expire date time of the promotion. format = “YYYY-MM-DD HH:MM:SS” |
| limit_level | Integer | It should be from 1 to n. Per request, API sends only recent 400 records in the response. To get 2nd recent 400 records, it value should be 2. This is similar to pagination. |
| player_id (optional) | String | Unique ID of player. If this paramtere sent in request, then response will be return in player specific. |

### BonusList() Response Example


```
{
     "result": {
        "status": "success",
        "headers": '[campaign_id, player_id, currency, game_ids, coin_value_level,
num_rounds, state, max_win_limit, promo_start_date, promo_end_date]',
        "bonus_list": [['NewYear-1', 'user123', 'USD', '[1032, 1041]', 1, 10, 'completed',
100, '2021-03-08 09:37:20', 2021-03-09 09:37:20'],
['NewYear-2', 'user123', 'USD', '[1032, 1041]', 1, 10, 'completed', 100, '2021-03-08
09:37:20', 2021-03-09 09:37:20']]
    }
}
```


## pfscoinvalues() API.

This API request enables the operator to fetch the coin values of all supported currencies for promo free spins. Values present in bet_values object are the ones that are used for wagering and same values are shown in the game UI. These coin values are in cents.

### pfscoinvalues() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/more/pfs-coin-values / |
| Headers | "Content-Type": "application/json" |
| POST Parameters | `json object`<br>`Ex: {"api_key":"6pjutYd23hL6vdGt", "game_id":6}` |

| Parameters | Type | Description |
|---|---|---|
| api_key | String | API Key allocated to each frontend |
| game_id | Integer | Please send the game ID, Operator can also find the game ID in get_games() API response. |

### pfscoinvalues () response example


```
{
    "result": {
        "game_name": "leprechaunsloot",
        "game_id": 6,
        "title": "Leprechaun's Loot",
        "bet_values": [{
            "AMD": [15000, 30000, 45000, 60000, 75000],
            "ARS": [900, 1800, 2700, 3600, 4500],
            "AUD": [30, 60, 90, 120, 150],
            "BGN": [30, 60, 90, 120, 150],
            "BRL": [150, 300, 450, 600, 750],
            "CAD": [30, 60, 90, 120, 150]
        }]
    }
}
```


## cancelpromotion() API.

This API request enables the operator to cancel the awarded bonus free spins (promotional free spins).

### cancelpromotion() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/more/cancel-promotion/ |
| Headers | "Content-Type": "application/json" |
| POST Parameters | `json object`<br><br>`Ex: {"api_key":"6pjutYd23hL6vdGt", "promo_freespin_id":"qw-ertyuiopasdfghjk"}` |

| Parameters | Type | Description |
|---|---|---|
| api_key | String | API Key allocated to each frontend |
| promo_freespin_id | String | This ID will be sent in grantbonus() API response, same promo_freespin_id should be sent here. |

### cancelpromotion() response example


```
{
    "result": {
        "status": "success",
        "campaign_id": "test_of1",
        "promo_freespin_id": "iuytrewqasdfghjk"
    }
}
```

# API Requests from Provider to Operator.

The following examples are some API requests that are currently provided to operators and third parties. This non-exhaustive list is provided as a brief overview. The number and type of requests can increase, as per an operator's requirements.

## GetSession() API.

Any Request parameters that are part of this API request would be provided with a token / session_id, typically a unique string assigned to each player. This response would contain information from the given session_id, such as player's account ID, username / alias, first name last name etc.

### GetSession() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/get_session |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.:<br>{<br>"token": "3c2ef39d02581e2db3ddf6c713e83e05"<br>} |

| Parameters | Type | Description |
|---|---|---|
| token | String | Unique string assigned to each player |

### GetSession() Response Example

Response would be in JSON format. Type: *dictionary*.

```
{
  "status":"0",
  "account_id":"123123",
  "username":"test_username",
  "country":"US",
  "token":"3c2ef39d-2581e2db3ddf6c713e83e05",
  "balance":"12345",
  "currency":"USD",
}
```

| Object / Key in response object | Type | Description |
|---|---|---|
| status | Integer | Response status |
| account_id | String | Account ID of the player |
| username | String | Username of the player or alternatively it can be the value of account_id when the actual username can't be shared |
| country | String | Country of the player |
| token | String | Unique string assigned to each player (session ID). |
| balance | Integer | Balance of the player |
| currency | String | Currency of the player |

## GetBalance() API.

This API request enables the operator to get the available balance of a player.

### GetBalance() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/get_balance |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.:<br>{<br>"token": "3c2ef39d02581e2db3ddf6c713e83e05"<br>"account_id": "123123"<br>} |

| Parameters | Type | Description |
|---|---|---|
| token | String | Unique string assigned to each player |
| account_id | String | Account ID of the player |

## GetBalance() Response Example

Response would be in JSON format. Type: *dictionary*.

```
{
  "status":"0",
  "account_id":"123123",
  "country":"US",
  "token":"3c2ef39d-2581e2db3ddf6c713e83e05",
  "balance":"12345",
  "currency":"USD",
}
```

| Object / Key in response object | Type | Description |
|---|---|---|
| status | Integer | Response status |
| account_id | String | Unique ID of the plaver |
| country | String | Country of the player in ISO format |
| token | String | Unique string assigned to each player (session ID). |
| balance | Integer | Available balance of the player |
| currency | String | Currency of the player |

# DebitAPI.

This API request debits a currency amount from a player's balance. This triggered every time a player plays a game. The response would contain information and details of any debited amounts.

## Debit() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/debit |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.:<br>{<br>"token": "cc918a3e4dea45ef5c31d6e3b9dce4afed3c02eb",<br>"account_id": "259823",<br>"amount": 100,<br>"amount_type": "real",<br>"currency": "GBP",<br>"game_id": 1,<br>"transaction_id": "123456",<br>"round_id": 198909,<br>"game_type": "slots",<br>"game_name": "testgamename",<br>"note": "debit"<br>"bonus_id": "test2147hff"<br>}<br> |

| Parameters | Type | Description |
|---|---|---|
| token | String | Unique string assigned to each player |
| account_id | String | Account ID of the player |
| amount | Integer | Amount to be debited from player's wallet. |
| amount_type | String | Indicate type of amount type. It can be real / promo_freespin. |
| currency | String | Currency of the player |
| game_id | Integer | Unique ID of the game |
| transaction_id | String | Unique id of the transaction |
| round_id | Integer | Round ID of each play / spin |
| game_type | String | Type of the game. slots / table_games / scratch_cards / bingo |
| game_name | String | Name of the game |
| note | String | Extra information about the debit request |
| bonus_id | String | Provider will send the bonus / campaign ID which is related to the promotional freespins. It will be null when it is in real mode. |

## Debit() Response Example

Response would be in JSON format. Type: *dictionary*.

```
{
    "status": 0,
    "account_id": "123123",
    "country": "US",
    "token": "3c2ef39d02581e2db3ddf6c713e83e05",
    "balance": 12345,
    "currency": "USD",
    "transaction_id": "205354854449856",
    "bonus_amount": "10"
}
```

| Object / Key in response object | Type | Description |
|---|---|---|
| status | Integer | Response status |
| account_id | String | Account ID of the player |
| country | String | Country of the player |
| token | String | Unique string assigned to each player (session ID). |
| balance | Integer | Balance of the player |
| currency | String | Currency of the player |
| transaction_id | String | Debit transaction ID |
| bonus_amount | Integer | Bonus amount available in plaver's wallet. If this amount is not available please send O value |

# Credit()API.

This would be similar to the Debit API Request. This API request is made to credit currency amount to a player's account balance. This is made every time a player wins a monetary value from the games. Its response would contain the information of all credited amounts

## Credit() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/credit |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.:<br>{<br>"token": "cc918a3e4dea45ef5c31d6e3b9dce4afed3c02eb",<br>"account_id": "259823",<br>"amount": 100,<br>"amount_type": "real",<br>"currency": "GBP",<br>"game_id": 1,<br>"transaction_id": "123456",<br>"round_id": 198909,<br>"game_type": "slots",<br>"game_name": "testgamename",<br>"note": "credit"<br>"bonus_id": "test2147hff"<br>"isfreespin":"true"<br>}<br> |

| Parameters | Type | Description |
|---|---|---|
| token | String | Unique string assigned to each player |
| account_id | String | Account ID of the player |
| amount | Integer | Amount to be credited to player's wallet. |
| amount_type | String | Indicate type of amount type. It can be real / promo_freespin. |
| currency | String | Currency of the player |
| game_id | Integer | Unique ID of the game |
| transaction_id | String | Unique id of the transaction |
| round_id | Integer | Round ID of each play / spin |
| game_type | String | Type of the game. slots / table_games / scratch_cards / bingo |
| game_name | String | Name of the game |
| note | String | Extra information about the credit request |
| round_end_state | Boolean | true indicates round is closed<br>false indicates round is open |
| bonus_id | String | Provider will send the bonus / campaign ID which is related to the promotional freespins. It will be null when it is in real mode. |
| Isfreespin | Boolean | true indicates it is a freespin<br>false indicates it is not a freespin |

## Credit() Response Example

Response would be in JSON format. Type: *dictionary*.

```
{
    "status": 0,
    "account_id": "123123",
    "country": "US",
    "token": "3c2ef39d02581e2db3ddf6c713e83e05",
    "balance": 12345,
    "currency": "USD",
    "transaction_id": "205354854449856",
    "bonus_amount": "10"
}
```

| Object / Key in response object | Type | Description |
|---|---|---|
| status | Integer | Response status |
| account_id | String | Account ID of the player |
| country | String | Country of the player |
| token | String | Unique string assigned to each player (session ID). |
| balance | Integer | Balance of the player |
| currency | String | Currency of the player |
| transaction_id | String | Credit transaction ID |
| bonus_amount | Integer | Bonus amount available in player's wallet. If this amount is not available please send 0 value. |

# Refund()API.

This API request allows the credit of an amount to a player's balance. This request is made when something goes wrong at the server / game-engine side and a previous debit request needs to be refunded. Its response would contain the information of any credited amount.

## Refund() Request

| Type | Input |
|---|---|
| HTTP Method | POST |
| API URL | https://<API_BASE_URL>/refund |
| Headers | "Content-Type": "application/json" |
| POST Parameters | json object e.g.:<br>{<br>"token": "cc918a3e4dea45ef5c31d6e3b9dce4afed3c02eb",<br>"account_id": "259823",<br>"amount": 100,<br>"original_transaction_id": 1655,<br>"transaction_id": 2322<br>"amount_type": "real",<br>"currency": "GBP",<br>"game_id": 1,<br>"round_id": 198909,<br>"game_type": "slots",<br>"game_name": "testgamename",<br>"note": "refund",<br>}<br>} |

| Parameters | Type | Description |
|---|---|---|
| token | String | Unique string assigned to each player |
| account_id | String | Account ID of the player |
| amount | Integer | Amount to be credited from player's wallet. |
| original_transaction_id | String | A transaction ID which is related to respective debit transaction. |
| transaction_id | String | New refund transaction ID, should not be the same as the failed transaction. |
| amount_type | String | Indicate type of amount type. It can be real only. |
| currency | String | Currency of the player. |
| game_id | Integer | Unique ID of the game. |
| round_id | Integer | Round ID related to the Debit request in question. |
| game_type | String | Type of the game. slots / table_games / scratch_cards / bing. |
| game_name | String | Name of the game |
| note | String | Extra information about the cancel debit request |

### Refund() Response Example

Response would be in JSON format. Type: *dictionary*.

```
{
    "status": 0,
    "account_id": "123123",
    "country": "US",
    "token": "3c2ef39d02581e2db3ddf6c713e83e05",
    "balance": 12345,
    "currency": "USD",
    "bonus_amount": "10"
}
```

## Error Handling and Error Codes.

An error response has the following format and data.

### UnfinishedGames() Response Example

```
    "error": 1,
    "error_details": {
        "id": "1001",
        "code": "INVALID_API_KEY",
        "message": "Invalid API key."
    }
```

| Object / Key in response object | Type | Description |
|---|---|---|
| Error | Integer | Its value would be 1 incase error occurs. |
| error_details | JSON Object | It has 3 keys id, code and message. |
| Id | Integer | Optional. Assigned to each error. |
| Code | String | Error code. |
| message | String | Optional. Human readable error message. |

## List of common errors codes and messages for all POST parameters

Below are common error codes for POST parameters. These will validate if any required POST parameter is missing, if it is null, length and size of the parameters etc.
Same error code can be used for more than one POST parameter.

| Common ID | Code | Message |
|---|---|---|
| 2001 | INVALID_%field_name% | %field_name% is required |
| 2002 | INVALID_%field_name% | %field_name% should not be null |
| 2003 | INVALID_%field_name% | %field_name% should not be blank |
| 2004 | INVALID_%field_name% | %field_name% should be at least %error_value% characters long. |
| 2005 | INVALID_%field_name% | %field_name% should not be more than %error_value% characters. |
| 2006 | INVALID_%field_name% | String is invalid |
| 2007 | INVALID_%field_name% | Email is invalid |
| 2008 | INVALID_%field_name% | A valid integer is required |
| 2009 | INVALID_%field_name% | The value should be equal or less than %error_value% |
| 2010 | INVALID_%field_name% | The value should be equal or more than %error_value% |
| 2011 | INVALID_%field_name% | Too large string value |

In the above table "%field_name%" can be any expected parameter that must be sent in the POST data. For example, let us take below POST data.

```
{"api_key":"1pghjl4v5apt2bks","username":"test_user","password":"test1234","transaction_id":1234}
```

In case, API caller is not sending the 'transaction_id' in POST data, RGS API would send below error message to the caller.

```
{"error":1, "error_detail": {"id":2001,"code":"INVALID_TRANSACTION_ID",
"message":"Transactionid is required."}}
```

error 'code' would be 'INVALID_' followed by missing parameter name in capitals i.e. 'INVALID_TRANSACTION_ID' and 'message' would be 'parameter name without underscore, words separated by space' and followed by 'is required.' i.e. 'Transaction id is required'.

| When 'transaction_id' parameter is missing in POST Data | | |
|---|---|---|
| **"ID"** | **"Code"** Interpretation | **"Message"** Sample Value |
| 2001 | INVALID_%field_name% will be interpreted as INVALID_TRANSACTION_ID | "%field_name% is required." will be interpreted as "Transaction id is required." |
| 2002 | INVALID_%field_name% will be interpreted as INVALID_TRANSACTION_ID | "%field_name% should not be null." will be interpreted as "Transaction id should not be null." |
| 2003 | INVALID_%field_name% will be interpreted as INVALID_TRANSCTION_ID | "%field_name% should not be blank." will be interpreted as "Transaction id should not be blank." |

Same is the case with all the POST parameters that fall into the above validation criteria from id 2001 to 2011.

## List of Error codes and messages

*(Excluding common code and message)*

| ID | Code | Message |
|---|---|---|
| 2012 | INVALID_LOGIN_CREDENTIALS | Unable to login with credentials provided. |
| 2013 | INVALID_LOGIN_CREDENTIALS | Must include username and password. |
| 2014 | INVALID_LOGIN_CREDENTIALS | User does not belong to this website. |
| 2015 | INVALID_API_KEY | API Key is invalid |
| 2016 | INVALID_SESSION_ID | Invalid session ID. |
| 2017 | INVALID_PROVIDER | Invalid provider. |
| 2018 | INVALID_GAME_TYPE | Invalid game type. |

| ID | Code | Message |
|---|---|---|
| 2012 | INVALID_LOGIN_CREDENTIALS | Invalid platform. |
| 2013 | INVALID_LOGIN_CREDENTIALS | Invalid amount type. |
| 2014 | INVALID_LOGIN_CREDENTIALS | Invalid player ID. |
| 2015 | INVALID_API_KEY | Invalid game ID. |
| 2016 | INVALID_SESSION_ID | Invalid language. |
| 2017 | INVALID_PROVIDER | Invalid username. |
| 2018 | INVALID_GAME_TYPE | Current & new password should not be same. |
| 2019 | INVALID_PLATFORM | Invalid currency. |
| 2020 | INVALID_PLATFORM | Amount should be positive value. |
| 2021 | INVALID_PLAYER_ID | Invalid player ID |
| 2022 | INVALID_GAME_ID | Invalid game ID |
| 2023 | INVALID_LANGUAGE | Invalid language |
| 2024 | INVALID_USERNAME | Invalid username |
| 2025 | CURRENT_NEW_PASSWORD_SAME | Current and new password should not be same |
| 2026 | INVALID_CURRENCY | Invalid currency |
| 2027 | NEGATIVE_AMOUNT_VALUE | Amount should be positive value |
| 2028 | INSUFFICIENT_BALANCE | Insufficient balance |
| 2029 | INVALID_STARTDATE_ENDDATE | Start date should no be greater than end date |
| 2030 | INVALID_AMOUNT_VALUE | Amount should be less than balance |
| 2031 | DUPLICATE_SESSION_ID | Duplicate session ID |

| ID | Code | Message |
|---|---|---|
| 2032 | OPERATOR_AUTHENTICATION_FAIL | Authentication with operator failed. |
| 2033 | CURRENCY_MISMATCH | Player currency mismatch between operator and opus. |
| 2034 | ZERO_AMOUNT_VALUE | Amount should be more than zero. |
| 2035 | FALIED_TRANS | Transaction not found or failed. |
| 2036 | INVALID_FROM_DATE | From date is greater than to date. |
| 2037 | INVALID_TIME_INTERVAL | Time interval is more than the allowed one. |
| 2038 | INVALID_MONETARY_TYPE | Invalid monetary type. |
| 2039 | BALANCE_AMOUNT_EXPIRED | Balance amount is expired. |
| 2040 | INVALID_TRANSACTION_TYPE | Invalid transaction type. |
| 2041 | DUPLICATE_TRANSACTION_ID | Duplicate transaction ID. |
| 2042 | DUPLICATE_PROMO_BONUS_ID | Duplicate promo bonus ID. |
| 2043 | INVALID_ROUND_OR_ROUND _BET_VALUE | Invalid round or round_bet value. |
| 2049 | INVALID_CURRENCY_COUNT | Currency count should match with player count |
| 2050 | INVALID_NUM_ROUND_COUNT | Number of rounds count should match with game_ids count |
| 2051 | INVALID_COIN_VALUES_COUNT IN | Coin values count should match with game ids count |
| 2052 | INVALID_ROUND_TOTAL _BET_COUNT | Round total bet count should match with game_ids count |
| 2053 | INVALID_COIN_VALUE | Invalid coin values |
| 3003 | EXTERNAL_GATEWAY_FAULT | Invalid JSON response from operator. |