# SPRING Partner API 0.21

## Reference Guide

**Rev 1.02**

## Revision History

| Rev # | Description | Initiator | Version | Date | Rev Done | Date |
|---|---|---|---|---|---|---|
| 1.01 | Proofreading made | A. Soghomonyan | 0.12 | Jul 13, 2016 | I. Mukhsiyan | Jul 14, 2016 |
| | Updated frontend integration parts | A. Soghomonyan | 0.11 | Apr 20, 2016 | | |
| | Added request/response samples. | G. Melkonyan | 0.12 | Apr. 26, 2016 | | |
| | Made GetClientDetails ExternalId as Mandatory | G. Melkonyan | 0.13 | Aug 22 2016 | | |
| | Added some clarifications to BetResulted and Rollback API calls. | G. Melkonyan | 0.14 | Sep 30 2016 | | |
| | Bonus API changes | G. Melkonyan | 0.15 | Oct 07 2016 | | |
| | Removed Mobile integration parts | G. Melkonyan | 0.16 | Oct 10 2016 | | |
| | Added hash source concatenated field examples | G. Melkonyan | 0.17 | Nov 24 2016 | | |
| | Added fields Source, MatchInfo and IsOutright fields to BetPlaced call | G. Melkonyan | 0.18 | Dec 14 2016 | | |
| | Added IsTest field to GetClientDetails response | G. Amirkhanyan | 0.19 | Jan 16 2017 | | |
| | Added CalcDate to BetResulted call | G. Amirkhanyan | 0.20 | Feb 16 2017 | | |
| | Added IsLive to BetResulted call | G. Amirkhanyan | 0.21 | Apr. 13 2017 | | |
| 1.02 | Proofreading | E. Hakobyan | - | May 16 2017 | I. Mukhsiyan | May 16 2017 |

## Table of Contents

Revision History ........ 1

Introduction ........ 4

1. Basic Concept ........ 4

2. Participants / Terminology ........ 5

3. Process Flow ........ 5

  3.1 Opening Sportsbook page ........ 6

  3.2 Placing bets ........ 6

  3.3 Bets outcomes reporting ........ 6

  3.4 Time synchronization ........ 7

  3.5 Operator Backend connection ........ 7

  3.6 Operator Backend SharedKey ........ 7

  3.7 OperatorBE server requirements ........ 7

4. AGP to Operator API calls ........ 8

  4.1 Request and response formats ........ 8

    4.1.1 Mandatory request parameters ........ 8

    4.1.2 Mandatory response parameters ........ 8

  4.2 Security checks ........ 8

    4.2.1 TS parameter ........ 9

    4.2.2 Hash parameter ........ 9

    4.2.3 PHP functions ........ 9

    4.2.4 AuthToken parameter ........ 9

  4.3 API Calls ........ 11

    4.3.1 GetClientDetails ........ 11

    4.3.2 GetClientBalance ........ 13

    4.3.3 BetPlaced ........ 14

    4.3.4 Rollback ........ 21

    4.3.5 GetBonusDetails ........ 22

5. BonusSetStatus ........ 24

6. Bonus API ........ 26

  6.1 Free Bet Bonus ........ 26

  6.2 Wagering Bonus ........ 26

7. Language Codes ........ 27

8. Error Codes ........ 28
9. Currency Codes ........ 28

## Introduction

Spring platform AGP is designed to support integration with  the third-party partner systems (Operators). Operators, as a rule, have their own site (casino, poker, sport betting, etc.) with existing integrations to other game providers and wish to integrate an AGP into their own site.

Operators can manage their players and balances/wallets separately in their own systems and integrate seamlessly. For this purpose Operators have to implement the Partner API, which is called by AGP.

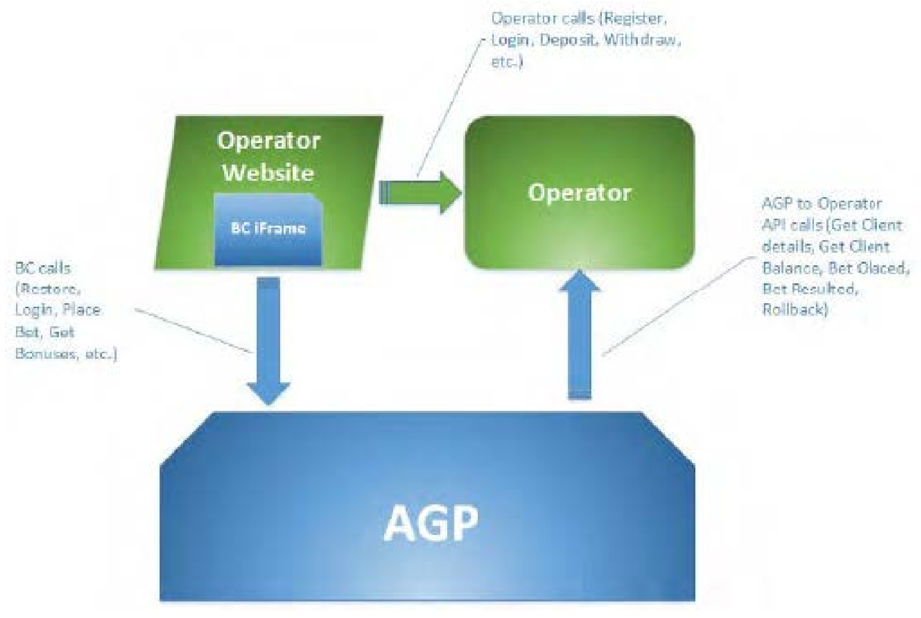
This API is a service contract with a collection of methods/calls with request and response messages. The format and details of the messages are described below in this document. The calls are accomplished during normal processing of AGP when exchanging or notifying some information with/to Operator is needed. The integration consists of two parts: web (iFrame) and back-end.

Web integration is used to provide UI for modules of upcoming and live games to place bets.

The back-end integration is implemented to send and receive messages between AGP and Operator (for example, placed bets, winning information, etc.).

## 1. Basic Concept

Concept of integrating AGP in common looks like below:


Operator calls (Register, Login, Deposit, Withdraw, etc.)
Operator Website
BC iFrame
Operator
AGP to Operator API calls (Get Client details, Get Client Balance, Bet Placed, Bet Resulted, Rollback)
BC calls (Restore, Login, Place Bet, Get Bonuses, etc.)
AGP

## 2. Participants / Terminology

| | | |
|---|---|---|
| **AGP** | – | “SPRING” Advanced Gaming Platform. |
| **AuthToken** | – | Unique string, identifying single user's single session, which is generated by Operator. |
| **BC** | – | BetConstruct |
| **BC iFrame** | – | Container supplied by BC and integrated into Operator's site. |
| **BetConstructBE** | – | BC's backend, providing information about odds and stake outcomes/results |
| **BetConstructSite** | – | BC's site, which generates iFrame content |
| **BettingSlip** | – | BC's betting slip integrated with Sportsbook. |
| **Operator** | – | Company/Partner, which intends to operate sport betting site. |
| **OperatorBE** | – | Backend system of the operator, holding user’s balance and doing all payment information processing. |
| **OperatorWebSite** | – | Operator's web site in which is integrated BC iFrame |
| **Sportsbook** | – | Module(s), supplied by BC and integrated into Operator's site. |
| **User** | – | Operator's customer, interacting with site and wanting to place bets. |

## 3. Process Flow

User interaction with the **OperatorWebSite** consists of several stages:

- loading site with **BC iFrame**
- placing bets

Backend to backend operations have following stages

- getting User’s information
- creating bets
- reporting bet outcomes

## 3.1 Opening Sportsbook page

- User opens web page on OperatorWebSite.
- OperatorWebSite includes a <script> tag, passing necessary parameters in URL.
- Included <script> tag creates an iFrame on page, loads Sportsbook and provides page-iFrame integration.

## 3.2 Placing bets

1. User selects event(s) from Sportsbook and places selected event into the BettingSlip.
2. After entering stake amount and checking all remaining information – user places stake by pressing corresponding button in BettingSlip.
3. Sportsbook calls corresponding handlers on BetConstructSite site, passing information about the user and the games/bet amount.
4. BetConstructSite calls BetConstructBE trying to put this bet with specified amount/odds/etc. in 'Hold' state, so that bet terms (odd and amount) are accepted, but bet is not approved yet. If bet detail changes occur and it could not be accepted – BetConstructSite returns corresponding error message which is shown to user.
5. BetConstructBE calls OperatorBE, trying to post the creating bet and receive Operators confirmation – **BetPlaced** call.
6. If user has no enough funds or communication error occurred
   a) BetConstructSite returns corresponding error message;
   b) it is shown to user;
7. Bet is canceled in BetConstructBE. In case of communication error BetConstructBE will tell OperatorBE to roll-back the bet - **Rollback** call.
8. If user has enough money and OperatorBE approves bet, then BetConstructBE confirms the bet.
9. BetConstructSite gives positive response to Sportsbook, which is shown to User.

## 3.3 Bets outcomes reporting

After a bet was accepted and if the status of the bet has changed (so the bet outcome is known), BetConstructBE reports the bet status change and winning amount to OperatorBE - **BetResulted** call. Bet status can be changed when bet won/lost/returned/cashed-out or rejected and with corresponding amount (zero
in case of lost stake, non-zero in remaining cases).

## 3.4 Time synchronization

Time on BetConstructWebSite is synchronized with global time using NTP protocol, the same should be done on Operator's side as well.

## 3.5 Operator Backend connection

To implement this API the Operator must provide address and port plus full URL of HTTP/HTTPS service. The OperatorBE service must accept only HTTP/HTTPS POST requests from BetConstruct addresses (request those from BetConstruct during the integration).

## 3.6 Operator Backend SharedKey

To sign and check requests the Operator must provide a SharedKey. The SharedKey is a 16-32 bytes long ASCII code (only printable characters, i.e. alphanumeric plus special symbols).

## 3.7 OperatorBE server requirements

The following factors may affect server's loading capability:

- Each user visiting your site and downloading Sportsbook does generate one GetClientDetails request – please calculate max number of simultaneous users at peak time.
- Every single bet typically generates 2 requests – one request at betting moment and another one when bet was placed. You need properly calculate a rate of requests. If you have N bets a day – expected figures are N x 2 requests,
- N/15000 requests per second in day peak times.
- Each site visitor generates GetClientBalance request every 20 seconds –so for 1000 simultaneous users it is expected having N/20 balance requests per second.

For major sporting events (Olympic Games, World Cup) the expected loading figures are about 5 times higher than normally.

Provide server's hardware and software capabilities to answer to all requests at peak time within 1-2 second time interval at least.
Actual server specifications are highly sensitive towards your back-end software, programming languages, implementation code quality and etc. and cannot be estimated from our side.

# 4. AGP to Operator API calls

The Operator has to implement an http endpoint for REST API calls from AGP. The calls are performed at certain AGP processing such as when the Client made a bet, the bet win is calculated, etc. Each callable method should have a unique URL to tailor to the base URL. For exact description of calls and request/response formats see below.

## 4.1 Request and response formats

Request and response messages are sent in JSON format using HTTP POST protocol. Each call has its own description for message details. All messages must meet the security requirements agreed between BC and Operator. All calls are implemented by request/response pattern: caller waits for an answer for the request.

### 4.1.1 Mandatory request parameters

| | | |
|---|---|---|
| **AuthToken (varchar 128)** | — | authentication token sent to client (JavaScript/Flex). This field is identified by a User. |
| **Hash (char 32, lowercase)** | — | checksum of all call parameters. See Security checks for more details. |
| **TS (long)** | — | request's timestamp. See Security checks for more details. |

### 4.1.2 Mandatory response parameters

**ErrorCode** — specifies an operation outcome. If operation succeeded, it is 0.

If error occurs, this is an OperatorBE internal error code. The Operator's own ErrorCodes should be within the range of 500-999, if documentation does not request to return some other error code for a call. For the full list of error codes that is recommended to visit the Error Codes section.

**ErrorText** – textual description of the error, can be empty in case of success.

## 4.2 Security checks

To prevent attacks and validate request sender and recipient identities, all request and response data has Hash and TS fields, which hold:

- Hash - checksum of all call parameters
- TS – request timestamp

### 4.2.1 TS parameter

Timestamp parameter is used to prevent request/response replay and to detect expired/old requests.

All the requests and responses holding TS (time stamp) parameter, which contains the UNIX time stamp (number of seconds since Unix Epoch - January 1 1970 00:00:00 GMT). Time on BetConstructBE is synchronized with global time using NTP protocol, the same should be done at Operator's side as well. While accepting packet, both sides must check the TS parameter to detect expired packets.
If time stamp is older than 20 seconds, this message is considered as expired or invalid one. The error code and message should be returned, no processing is done.

### 4.2.2 Hash parameter

Hash parameter is used to validate sender's identity and prevent data modification in transit and is calculated based on all parameters (except Hash parameter) in request/response message and shared key. Each OperatorBE has its own shared key.

Hash code is calculated as MD5 checksum of concatenation of all parameter names and values and shared key by the end.

1. Loop over all non-empty parameters in the message (do not include Hash parameter) and compose the string.
   **ATTENTION: Keep the order of parameters exactly as those are listed in this document.**
2. Concatenate it with Operator's shared key.
3. Calculate MD5 string.
4. Include MD5 checksum into response as Hash parameter.

**WARNING**: Hash signature **MUST** be all lowercase. Otherwise, it will be fail validation at BetConstruct side.

### 4.2.3 PHP functions

Use **`time()`** function to get Unix Timestamp. Use **`md5()`** function to calculate MD5 checksum of the string.

### 4.2.4 AuthToken parameter

### *AuthToken life time*

To prevent AuthToken reuse and to control expired sessions, AuthToken should have short expiration time – a few minutes, but not less than 1 minute.

Each call to OperatorBE should extend AuthToken lifetime for 1 minute at least. If user is idle, JavaScript/Flex client will use GetBalance call to keep the token fresh.

### *AuthToken and currency*

**IMPORTANT!**

During the user session OperatorBE should **NEVER change currency** for that AuthToken. So, if user needs, or has an ability to change his currency – the entire iFrame application should be reloaded and a **NEW** AuthToken should be generated.
So, each AuthToken is always bound to currency reported in GetClientDetails and this binding never changes.

**Breaking this point will mess all your financial transactions, so never ever do that**.

## 4.3 API Calls

### 4.3.1 GetClientDetails

This method is called when a player loads Sportsbook. Method receives AuthToken generated by OperatorBE server and returns players data in case player was logged into the OperatorWebSite and token was valid. If user is not logged in, method should return ErrorCode 1008. In case of other problems (for example, expired AuthToken, or similar) other ErrorCodes should be used.

Player currency should match the list of currencies, specified in Currency Codes section.

URL: /GetClientDetails

Example: http://operatorhostname:port/api/GetClientDetails where http://operatorhostname:port/api is the Base URL for all calls.

Request parameters:

| **Parameter Name** | **Type** | **Description** |
|---|---|---|
| **AuthToken** | String | See **AuthToken parameter** for details. |
| **TS** | Long | See **TS parameter** for details. |
| **Hash** | String | See **Hash parameter** for details. |

Response parameters:

| **Parameter Name** | **Type(max length)** | **Description** |
|---|---|---|
| Login | String(255) | Login of the User in OperatorBE used to log-in. **(Mandatory)** |
| CurrencyId | String(3) | User's currency id. See Currency Codes for valid values. **(Mandatory)** |
| LanguageId | String(2) | Two letter code of the User's language. For example "en" – for English, "ru" - for Russian, etc. See Language Codes for valid values. |
| Email | String(255) | Email address of User. |

| | | |
|---|---|---|
| `BirthDate` | Date | Birth date of User. |
| `Gender` | Int | 1 – for Male, 2 – for Female. |
| `ExternalId` | String(255) | Unique ID of User in OperatorBE. **(Mandatory)** |
| `CurrentIp` | String(50) | IP address of the User currently logged-in. |
| `Phone` | String(255) | Primary phone number of User. |
| `Address` | String(255) | Primary address of User. |
| `ErrorCode` | | See Error Codes for valid values. Mandatory in case of error. |
| `ErrorText` | | Error description |
| `ExternalBonusId` | Int | Unique bonus ID in OperatorBE currently assigned to User. Can be omitted if Operator does not use bonuses, or User has no current bonus. See Bonus API for more details. |
| `IsTest` | Boolean | true, false. Indicates if the player is a test player |

Mandatory fields are indicated as **(Mandatory)**. Otherwise, the parameters are optional. Optional fields could be omitted from the response.

Fields `Gender, BirthDate, CurrentIp, Phone, Address` are used for sport betting risk management and are advised to include them in the responses when possible.

In case of anonymous, non-authenticated user return ErrorCode 1008.

A request sample:

```
{
      "AuthToken":"your_client_security_token",
      "TS":1461682696,
      "Hash":"hash_code_generated_from_request_fields"
}
```

Where hash source string is:

**AuthToken**your_client_security_token**TS**1461682696**your_shared_key**

And a response sample:

```
{
	"Login":"player_login",
	"CurrencyId":"USD",
	"LanguageId":"en",
	"Email":"player@domain.com",
	"BirthDate":"1986-04-26",
	"Gender":1,
	"ExternalId":"partner_system_player_id",
	"CurrentIp":"123.123.123.123",
	"Phone":"234324",
	"Address":"player address",
	"ErrorCode":"0",
	"ErrorText":""
}
```

## 4.3.2 GetClientBalance

This method is used to retrieve current User's (player's) balance in player's currency.

URL: /GetClientBalance

Example: http://operatorhostname:port/api/GetClientBalance where http://operatorhostname:port/api is the Base URL for all calls.

Request parameters:

| **Parameter Name** | **Type** | **Description** |
|---|---|---|
| AuthToken | String | See **AuthToken parameter** for details. |
| TS | Long | See **TS parameter** for details. |
| Hash | String | See **Hash parameter** for details. |

Response parameters:

| **Parameter Name** | **Type(max length)** | **Description** |
|---|---|---|
| `Balance` | Decimal | User's current balance in User's currency. **(Mandatory)** |
| `ErrorCode` | | See Error Codes for valid values. Mandatory in case of error. |
| `ErrorText` | | Error description |

Mandatory fields are indicated as **(Mandatory)**. Otherwise, the parameters are optional. Optional fields could be omitted from the response.

In case of anonymous, non-authenticated user return ErrorCode 1008.

A request sample:

```
{
      "AuthToken":"your_client_security_token",
       "TS":1461682696,
       "Hash":"hash_code_generated_from_request_fields"
}
```

Where hash source string is:

**AuthToken**your_client_security_token**TS**1461682696**your_shared_key**

And a response sample:

```
{
      "Balance":123.00,
      "ErrorCode":"0",
      "ErrorText":""
}
```

### 4.3.3 BetPlaced

This method is called by BetConstructBE when User placed a bet. The bet is put into pending status and a call is performed. OperatorBE should validate the placed bet and take money off from the player's account. If there is not enough money, then OperatorBE should return insufficient funds error. In case of any error or communication failure, the placed bet is canceled and an error message displays to User.

URL: /BetPlaced
Example: http://operatorhostname:port/api/BetPlaced where http://operatorhostname:port/api is the Base URL for all calls.

Request parameters:

| Parameter Name | Type | Description |
|---|---|---|
| `AuthToken` | String | See **AuthToken parameter** for details. |
| `TS` | Long | See **TS parameter** for details. |
| `Hash` | String | See **Hash parameter** for details. |
| `TransactionId` | Long | Unique transaction ID in BetConstructBE to identify each call to OperatorBE regarding to bets. This is not a bet ID. This ID should be kept by OperatorBE to eliminate duplicate calls. It is also used in Rollback call to cancel previous transactions. **(Mandatory)** |
| `BetId` | Long | Unique ID of the placed bet in BetConstructBE to identify the bet. This value should be referenced when talking to LiveChat and other support services at BetConstruct. . **(Mandatory)** |
| `Amount` | Decimal | An amount (real money) in User's currency placed on the bet. **(Mandatory)** |
| `Created` | DateTime | A UTC time the bet was placed. **(Mandatory)** |
| `BetType` | Int | 1 – Single/Ordinary, 2 – Multiple/Express, 3 – System. **(Mandatory)** |
| `SystemMinCount` | Int | The count of sub-bet counts when System bet is placed. Only for 'system', how much events were placed in system 2 (of 3), for example. |
| `TotalPrice` | Decimal | Total price/odds/coefficient of the placed bet. |
| `Selections` | Array | The list of selections/events of the placed bet. Always 1 for Single bets, and more than 1 for Multiple and System bets. This field should not be included in Hash generation. **(Mandatory) - at least one** |
| `BonusBetAmount` | Decimal | Bonus amount in User's currency used to place the bet. This field may be greater than 0 when bonus is used on this bet, and some portion of the bet amount is taken from the bonus amount. The field can be ignored if Operator does not use Bonuses. See Bonus API for more details. |

| | | |
|---|---|---|
| BonusId | Int | User’s bonus ID used for this bet. If the User’s bonus was created by GetBonusDetails API call, the value is the ExternalBonusId received from GetClientDetails call. |
| Source | Int | 42 - HTML5, 4 - Mobile, 16 - Angroid, 17 - iOS, 98 - Betshop, 99 - Terminal. The field is not included in hash. |

Selection/Event parameters

| Parameter Name | Type | Description |
|---|---|---|
| SelectionId | Long | Unique ID of the bet’s selection/event in BetConstructBE to identify the selection. **(Mandatory)** |
| SelectionName | String(255) | The name of the selection/event. For example, P1. |
| MarketTypeId | Long | Unique ID of the selection group (market) in BetConstructBE to identify market type. **(Mandatory)** |
| MarketName | String(255) | The name of the market. For example, Match Result. |
| MatchId | Int | Unique ID of the match in BetConstructBE. **(Mandatory)** |
| MatchName | String(500) | The name of the match. For example, Barcelona – Real Madrid. |
| MatchStartDate | DateTime | UTC time of the start of the match. |
| RegionId | Int | Unique ID of the region the match is taking place in BetConstructBE. **(Mandatory)** |
| RegionName | String(255) | The name of the region. For example, Spain. |
| CompetitionId | Int | Unique ID of the competition of the match in BetConstructBE. **(Mandatory)** |
| CompetitionName | String(255) | The name of the competition. For example, La Liga. |
| SportId | Int | Unique ID of the sport in BetConstructBE. **(Mandatory)** |
| SportName | String(255) | The name of the sport. For example, Football. |
| Price | Decimal | The price/odd of the selection. **(Mandatory)** |
| IsLive | Boolean | True – when the selection is on live match, False – Pre-Match. |
| Basis | Decimal | Handicap value for handicap type markets. For example, -1, 2.5, etc. |

| | | |
|---|---|---|
| `MatchInfo` | `String` | A text describing the score and some live info when the bet is placed on this Live event. |
| `IsOutright` | `Boolean` | True - when the selection is an outright match/market, otherwise - False. |

Response parameters:

| **Parameter Name** | **Type(max length)** | **Description** |
|---|---|---|
| `ErrorCode` | | See Error Codes for valid values. Mandatory in case of error. |
| `ErrorText` | | Error description |

Mandatory fields are indicated as **(Mandatory)**. Otherwise, the parameters are optional. Optional fields could be omitted from the response.

In case of anonymous, non-authenticated user return ErrorCode 1008.

A request sample:

```
{
      "AuthToken":"your_client_security_token",
      "TS":1461670530,
      "TransactionId":34234324,
      "BetId":123456,
      "Amount":123.00,
      "Created":"2016-04-26T11:35:30.0543787Z",
      "BetType":1,
      "SystemMinCount":null,
      "TotalPrice":2.500,
      "Selections":[{
            "SelectionId":42343,
            "SelectionName":"P1",
            "MarketTypeId":333,
            "MarketName":"Match Result",
            "MatchId":55,
            "MatchName":"Barcelona - Real Madrid",
            "MatchStartDate":"2016-04-26T11:35:30.0543787Z",
            "RegionId":22,
            "RegionName":"Spain",
            "CompetitionId":44,
            "CompetitionName":"La Liga",
            "SportId":1,
            "SportName":"Football",
            "Price":2.5
            }],
    "Hash":"hash_code_generated_from_request_fields"
}
```

Where hash source string is:

AuthTokenyour_client_security_tokenTS1461670530TransactionId34234324BetId123456Amount123.00Created2016-04-26T11:35:30.0543787ZBetType1TotalPrice2.500your_shared_key

And a response sample:

```
{
"ErrorCode":"0",
"ErrorText":""
}
```

BetResulted

This method is called by BetConstructBE when an accepted bet was resulted or result was canceled (bet status or winning amount was changed). Operator may receive multiple BetResulted calls for one bet. This can occur when:

- User's bet was rejected or returned and amount is returned to user account (exactly same amount as was when placed the bet) - BetState:1, Amount:BetStakeAmount.
- User's bet won (amount is calculated by BetConstructBE and usually greater than the bet stake amount) - BetState:4, Amount:won amount.
- User's bet lost and BetConstructBE notifies that bet lost - BetState:3, Amount:0.
- User's bet is cashed-out (amount is calculated by BetConstructBE and usually it is less than the bet stake amount) - BetState:5, Amount:cashed-out amount.
- User's bet result is reverted/canceled and the bet was resulted as not settled/initial state - BetState:1, Amount:0.

*Retry Logic*

When this call fails due to communication or network, the BetConstructBE tries to resend the message. Each retry is performed after N minutes where N is the retry count. After 5 retries the message remains in the queue with "No Answer" status.

URL: /BetResulted

Example: http://operatorhostname:port/api/BetResulted where http://operatorhostname:port/api is the Base URL for all calls.

Request parameters:

| Parameter Name | Type | Description |
|---|---|---|
| `AuthToken` | String | See AuthToken parameter for details. AuthToken used in this call may be very old (several hours for live, several days or even months for prematch). This occurs because bet outcome is known after significant amount of time after the stake. You must distinguish these transactions and allow their processing. |
| `TS` | Long | See **TS parameter** for details. |
| `Hash` | String | See **Hash parameter** for details. |
| `TransactionId` | Long | Unique transaction ID in BetConstructBE to identify each call to OperatorBE regarding to bets. This is not a bet ID. This ID should be kept by OperatorBE to eliminate duplicate calls. It is also used in Rollback call to cancel previous transactions. **(Mandatory)** |
| `BetId` | Long | Unique ID of the bet in BetConstructBE to identify the bet. This value should be referenced when talking to LiveChat and other support services at BetConstruct. . **(Mandatory)** |
| `BetState` | Int | The resulted state of the bet. **(Mandatory)**<br>1 – Accepted (when resulted bet canceled due to match results are changed)<br>2 – Returned<br>3 – Lost<br>4 – Won<br>5 – Cashed-out |
| `Amount` | Decimal | Real Amount in User's currency. OperatorBE can compare this amount to the previously called BetResulted Amount and increase or decrease User's balance. OperatorBE can store this value on the bet's record as the final total amount returned to User's balance from this bet. **(Mandatory)** |
| `BonusAmount` | Decimal | Bonus amount in User's currency won from this bet. When the bet is a wagering bonus bet, the won amount goes into this field. See Bonus API for more details. |
| `BonusId` | Int | User's bonus ID used for this bet. If the User's bonus was created by GetBonusDetails API call, the value is the ExternalBonusId received from GetClientDetails call. |
| `CalcDate` | DateTime | The UTC time that the bet was settled. |
| `IsLive` | Boolean | True – when the selection is on live match, False – Pre-Match. |

Response parameters:

| Parameter Name | Type(max length) | Description |
|---|---|---|
| `ErrorCode` | | See Error Codes for valid values. Mandatory in case of error. |
| `ErrorText` | | Error description |

Mandatory fields are indicated as **(Mandatory)**. Otherwise, the parameters are optional. Optional fields could be omitted from the response.

In case of anonymous, non-authenticated user return ErrorCode 1008.

A request sample:

```
{
	"TransactionId":34234325,
	"Amount":307.50,
	"BetId":123456,
	"BetState":4,
	"AuthToken":"your_client_security_token",
	"TS":1461671373,
	"Hash":"hash_code_generated_from_request_fields"
}
```

Where hash source string is:

**AuthToken**your_client_security_token**TS**1461671373**TransactionId**34234325**BetId**123456**BetState**4**Amount**307.50**your_shared_key**

And a response sample is:

```
{
	"ErrorCode":"0",
	"ErrorText":""
}
```

### 4.3.4 Rollback

This method is called by BetConstructBE when a processed placed bet needs to be rolled-back in OperatorBE.

This can occur when user placed bet is waiting for Operator confirmation and the BetPlaced call fails, some communication error happens or bet placement transaction fails in BetConstructBE. If OperatorBE has not processed the original BetPlaced call with this TransactionId before, it can be ignored. If OperatorBE has already rolled-back the original BetPlaced transaction with this TransactionId, it can be ignored, too. In both cases, a valid response should be returned from the call, to eliminate retries from BetconstructBE.

***Retry Logic***

See Retry Logic for more details how BetConstructBE retries to send messages.

URL: /Rollback

Example: http://operatorhostname:port/api/Rollback where http://operatorhostname:port/api is the Base URL for all calls.

Request parameters:

| Parameter Name | Type | Description |
|---|---|---|
| `AuthToken` | `String` | See AuthToken parameter for details. AuthToken used in this call may be very old (several hours for live, several days or even months for prematch). This occurs because bet outcome is known after significant amount of time after the stake. You must distinguish these transactions and allow their processing. |
| `TS` | `Long` | See TS parameter for details. |
| `Hash` | `String` | See Hash parameter for details. |
| `TransactionId` | `Long` | Transaction ID in BetConstructBE to be rolled-back in OperatorBE. (Mandatory) |

Response parameters:

| Parameter Name | Type(max length) | Description |
|---|---|---|
| `ErrorCode` | | See Error Codes for valid values. Mandatory in case of error. |
| `ErrorText` | | Error description |

A request sample:

```
{
	"TransactionId":34234325,
	"AuthToken":"your_client_security_token",
	"TS":1461671373,
	"Hash":"hash_code_generated_from_request_fields"
}
```

Where hash source string is:

**AuthToken**your_client_security_token**TS**1461682696**TransactionId**34234325**your_shared_key**

And a response sample:

```
{
	"ErrorCode":"0",
	"ErrorText":""
}
```

### 4.3.5 GetBonusDetails

This method is called by BetConstructBE to fetch the bonus details assigned to the User in OperatorBE. The method is called when OperatorBE reports that new bonus is available for User and BetConsturctBE does not know about that bonus ID yet.

Usually this will be called once when bonus appears for a first time.

OperatorBE must always return same information for the same bonus ID.

See Bonus API for more details.

URL: /GetBonusDetails

Example: http://operatorhostname:port/api/GetBonusDetails where http://operatorhostname:port/api is the Base URL for all calls.

Request parameters:

| Parameter Name | Type | Description |
|---|---|---|
| `AuthToken` | String | See **AuthToken parameter** for details. |
| `TS` | Long | See **TS parameter** for details. |
| `Hash` | String | See **Hash parameter** for details. |
| `BonusId` | Int | Bonus ID in OperatorBE received through GetClientDetails call's ExternalBonusId field. **(Mandatory)** |

Response parameters:

| Parameter Name | Type(max length) | Description |
|---|---|---|
| `BonusId` | Int | Bonus ID in OperatorBE received through GetClientDetails call's ExternalBonusId field. **(Mandatory)** |
| `BonusDefId` | Int | Unique Bonus definition Id in BetContructBE. Bonus definitions are created in BetConstruct Bonus Engine. **(Mandatory)** |
| `BonusAmount` | Decimal | Total bonus amount in User's currency to give to User. **(Mandatory)** |
| `RealAmount` | Decimal | Total real money in wagering bonus to be used when placing bets. The field is ignored for Free Bet Bonus. |
| `ExpirationDate` | DateTime | UTC date and time in the future when bonus will expire if User does not use it. If this field is not specified, the BetConstructBE will calculate the ExpirationDate from the bonus definition. |
| `ErrorCode` | | See Error Codes for valid values. Mandatory in case of error. |
| `ErrorText` | | Error description |

A request sample is:

```
{
      "AuthToken":"your_client_security_token",
      "TS":1461671373,
      "Hash":"hash_code_generated_from_request_fields",
      “BonusId”:112233
}
```

Where hash source string is:

**AuthToken**your_client_security_token**TS**1461671373**BonusId**112233**your_shared_key**

And a response sample:

```
{
      “BonusId”:112233,
      “BonusDefId”:123,
      “BonusAmount”:100.00,
      “RealAmount”:100.00,
      “ExpirationDate”:"2016-10-30T11:35:30.0543787Z",
      "ErrorCode":"0",
      "ErrorText":""
}
```

## 5. BonusSetStatus

This method is called by BetConstructBE to report the bonus status change to Operator.

See Bonus API for more details.

URL: /BonusSetStatus

Example: http://operatorhostname:port/api/BonusSetStatus where http://operatorhostname:port/api is the Base URL for all calls.

Request parameters:

| Parameter Name | Type | Description |
|---|---|---|
| AuthToken | String | See **AuthToken parameter** for details. |
| TS | Long | See **TS parameter** for details. |

| | | |
|---|---|---|
| `Hash` | String | See **Hash parameter** for details. |
| `BonusId` | Int | Bonus ID in OperatorBE received through GetClientDetails call's ExternalBonusId field. **(Mandatory)** |
| `Status` | Int | 0 - Activated<br>1 - Succeeded<br>2 - Lost<br>3 - Canceled<br>4 - Expired<br>**(Mandatory)** |
| `BonusAmount` | Decimal | Remaining bonus amount in User's currency from user's bonus wallet (not a real money, just for informational purposes). **(Mandatory)** |
| `RealAmount` | Decimal | Remaining real amount in User's currency to be pushed back to User's balance. **(Mandatory)** |
| `UpdateTS` | Long | Epoch Date and Time of the bonus status change. **(Mandatory)** |

Response parameters:

| Parameter Name | Type(max length) | Description |
|---|---|---|
| `ErrorCode` | | See Error Codes for valid values. Mandatory in case of error. |
| `ErrorText` | | Error description |

A request sample is:

```
{
	"AuthToken":"your_client_security_token",
	"TS":1461671373,
	"Hash":"hash_code_generated_from_request_fields",
	“BonusId”:112233,
	“Status”:1,
	“BonusAmount”:100.00,
	“RealAmount”:50.00,
	“UpdateTS”:32231113
}
```

Where hash source string is:

**AuthToken**your_client_security_token**TS**1461671373**BonusId**112233**Status**1**BonusAmount**100.00**RealAmount**50.00**UpdateTS**32231113**your_shared_key**

And a response sample is:

```
{
     "ErrorCode":"0",
     "ErrorText":""
}
```

# 6. Bonus API

Operator is capable to use BetConstruct Bonus Engine to grant Users bonuses and accept bets using the assigned bonuses. Prior to starting using bonuses, Operator needs to create bonus definitions in BetConstruct Back-Office. After bonus definitions created, Operator can issue/assign bonuses to certain Users in OperatorBE by using the BonusDefId of the bonus definition. Each time the User logins into BetConstruct iFrame and the GetClientDetails call is performed, the Operator can pass bonus ID from OperatorBE in the response message ExternalBonusId field. When BetConstructBE receives a new ExternalBonusId for User, it calls API method GetBonusDetails to fetch the bonus information from Operator. If the call succeeded, the bonus details are used to create a User’s bonus in Bonus Engine, and User can use the bonus to place bets. When BetConstructBE fails to create User bonus, the User still can place bets, but without bonus.

## 6.1 Free Bet Bonus

Free Bet bonus is used for one bet and for one time. The entire bonus amount is placed on one bet. In this case, the BetPlaced call will have the Bonus ID of the free bet bonus, the BonusAmount will indicate the free bet bonus amount, and Amount field will be 0. After receiving this call, Operator can set the Free Bet Bonus as used and can issue another bonus for this User.

When the bet with Free Bet Bonus wins, the winning amount is calculated as usual and is reduced by free bet bonus amount. The won amount completely goes into Amount field of the BetResulted call.

## 6.2 Wagering Bonus

When a bet is placed and the wagering bonus is used, Operator receives BonusId in BetPlaced call. The bet stake amount is allocated in the following way: at first won amount of this bonus is considered, then real money, and then bonus money is used. Finally, the real money goes into Amount field of the BetPlaced call, and bonus money goes into BonusAmount field.

During the active wagering bonus period, any won bet with Wagering Bonus ID will have the won amount in BonusAmount field of the BetResulted call. Any won amount by this bonus is added to Bonus Win balance and future bets using this wagering bonus consider this balance at first.

For wagering bonus, Operator receives BonusSetStatus calls when bonus status changes. The first call should be when a wagering bonus is activated (Status=0). This call usually happens when a new Bonus ID is passed along with User details and bonus details are fetched from Operator.

When User completes with wagering requirements and wagered amount (total amount of bet stakes using the bonus) is greater than, or equals to the required wagering amount, the BonusSetStatus is called to report successful completion of the bonus (Status=1). In this case, BonusAmount along with RealAmount are added to User's balance. For the unsettled bets placed with this bonus, Operator will receive BetResulted calls when the bet is settled, and the Amount for winning bets will be real money.

User's bonus has expiration date and after that date an Operator receives a BonusSetStatus call with Expired status (Status=4) and the Bonus ID will not be valid anymore. BonusAmount and RealAmount show amounts in the corresponding wallets. RealAmount shows the remainder from real money and must be returned to User's balance. After receiving such a status Operator can issue another Bonus for this User.

# 7. Language Codes

| **Code** | **Language** |
|---|---|
| en | English |
| ru | Russian |
| lt | Lithuanian |
| ka | Georgian |
| zh | Chinese |
| tr | Turkish |
| fr | French |
| es | Spanish |
| sr | Serbian |
| et | Estonian |
| lv | Latvian |
| me | Montenegrian |

If a language code is not supported by Sportsbook, it will fall-back to English.

Sportsbook supports per skin translation changes, so that customer can have different names for interface elements on his site.

## 8. Error Codes

Operator’s own ErrorCodes should fall into range 500-999.

| Code of Error | Description |
|---|---|
| 1000 | Internal Error |
| 1003 | User is blocked |
| 1004 | User is dismissed |
| 1005 | Password/AuthToken error |
| 1008 | Logging in the page is not possible, since you have not activated your account from your e-mail. |
| 1117 | Wrong login or E-mail |
| 1550 | The sum exceeds maximum allowable limit |
| 1560 | The sum is less than minimum allowable limit |
| 1700 | API wrong access exception |
| 2200 | Client limit exception |
| 2300 | Office limit exception |
| 2400 | Client balance is less |

## 9. Currency Codes

**AMD**

**AZN**

**BYR**

**CHF**

**EUR**

**GBP**

**GEL**

**IRR**

**KGS**

**KRW**

**KZT**

**`LTL`**

**`LVL`**

**`MDL`**

**`NGN`**

**`RSD`**

**`RUB`**

**`TRY`**

**`UAH`**

**`USD`**

**`VND`**

**`XAF`**

See ISO 4217 standard for details.